Mario Montagud
Pablo Cesar
Fernando Boronat
Jack Jansen *Editors*

MediaSync

Handbook on Multimedia Synchronization

Springer

MediaSync

Mario Montagud · Pablo Cesar
Fernando Boronat · Jack Jansen
Editors

MediaSync

Handbook on Multimedia Synchronization

Springer

Editors
Mario Montagud
Departamento de Comunicaciones
Universitat Politècnica de València
(UPV) – Campus de Gandia
Valencia, Grao de Gandia
Spain

and

Centrum Wiskunde & Informatica (CWI)
Amsterdam
The Netherlands

Pablo Cesar
Centrum Wiskunde & Informatica (CWI)
Amsterdam
The Netherlands

Fernando Boronat
Departamento de Comunicaciones
Universitat Politècnica de València
(UPV) – Campus de Gandia
Valencia, Grao de Gandia
Spain

Jack Jansen
Centrum Wiskunde & Informatica (CWI)
Amsterdam
The Netherlands

ISBN 978-3-319-65839-1 ISBN 978-3-319-65840-7 (eBook)
https://doi.org/10.1007/978-3-319-65840-7

Library of Congress Control Number: 2017962556

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

Since the early years of multimedia research and development, synchronization (multimedia synchronization, media synchronization, or just "mediasync" as used in this book) has always been a key issue. At that time, media synchronization was introduced as a brave new topic in scientific workshops. For example, in 1993, the first mature papers on synchronization were presented at the first ACM Multimedia Conference. This ACM Multimedia '93 took place in Anaheim, California, USA, paving the way for special issues on media synchronization, such as "Synchronization Issues in Multimedia Communications" that appeared in the IEEE Journal on Selected Areas in Communications (JSAC) in 1996.

Media synchronization is all about the tight relationship between media data, the deficiencies of our technical world, and how humans perceive media and interact with correlated media data. In general, humans do not care for media data to be 100% accurate. The interaction of humans, of us, with media data as well as the perception has "to look like" being perfectly in sync; any imperfection shall not be noticeable.

In a nutshell: Related media data are always treated as one medium, i.e., there is no need to think explicitly about synchronization. In some cases, like the "antique" two-channel stereo, it is the best option, however, it is too restricted to be *the* overall approach.

In detail: Each individual media data item may be "handled" by itself; synchronization has to be enforced in time and space. In some cases, like the interaction over the Internet with haptic devices in a 3D virtual reality scene, it might be too complicated to be *the* general approach.

This outstanding *Handbook on Multimedia Synchronization* covers all important topics located "between" the nutshell and the detailed approaches.

Many of the recent presentations and papers on multimedia make use of machine learning methods that provide means to "handle" correlated media data "correctly". Hence, one could train any kind of synchronization to perform correctly, similar to training a pilot to react correctly in whatever flying situation that may occur. There is no guarantee: this approach performs its media synchronization tasks correctly under real-world real-time constraints. As appealing as this approach sounds, there

are still too many challenges ahead for bringing these learning approaches to every aspect of life.

I wish you—the reader of this handbook—to gain an in-depth knowledge into one of the most challenging topics of multimedia and human–computer interfaces.

Darmstadt
Germany

Ralf Steinmetz
Technische Universität Darmstadt

Contents

Contributors

Stephen Appleby British Telecommunications PLC, London, UK

Ingar M. Arntzen Norut Northern Research Institute, Tromsø, Norway

Rolf Bardeli Vodafone GmbH, Düsseldorf, Germany

Gunilla Berndtsson Digital Representation and Interaction, Ericsson Research, Stockholm, Sweden

Njål T. Borch Norut Northern Research Institute, Tromsø, Norway

Fernando Boronat Departamento de Comunicaciones, Universitat Politècnica de València (UPV) – Campus de Gandia, Valencia, Grao de Gandia, Spain

Dick C. A. Bulterman Vrije Universiteit Amsterdam and CWI, Amsterdam, The Netherlands

Pablo Cesar Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

Cyril Concolato Telecom ParisTech – LTCI, Paris, France

François Daoust World Wide Web Consortium (W3C), Paris, France

Romualdo M. de R. Costa Department of Computer Science, UFJF, Juiz de Fora, MG, Brazil

Frank den Hartog DoVes Research, Canberra, Australia

Ragnhild Eg Westerdals – Oslo School of Art, Communication and Technology, Oslo, Norway

Håvard Espeland ForzaSys AS, Fornebu, Norway

Marcelo F. Moreno Department of Computer Science, UFJF, Juiz de Fora, MG, Brazil

Marcio Ferreira Moreno IBM Research, Rio de Janeiro, RJ, Brazil

Vamsidhar R. Gaddam Simula Research Laboratory, Fornebu, Norway

David Geerts Mintlab, KU Leuven/imec, Leuven, Belgium

Carsten Griwodz Simula Research Laboratory, Fornebu, Norway

Pål Halvorsen Simula Research Laboratory, Fornebu, Norway

Pingguo Huang Seijoh University, Aichi, Japan

Zixia Huang Google Inc., Mountain View, USA

Peter Hughes British Telecom Research and Innovation, Martlesham Heath, Ipswich, UK

Yutaka Ishibashi Nagoya Institute of Technology, Nagoya, Japan

Jack Jansen Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

Dag Johansen UiT – The Artic University of Norway, Tromsø, Norway

Håvard D. Johansen UiT – The Artic University of Norway, Tromsø, Norway

Tomas Kupka ForzaSys AS, Fornebu, Norway

Ragnar Langseth ForzaSys AS, Fornebu, Norway

Brian Lee Department of Electronics & Informatics, Athlone Institute of Technology, Westmeath, Ireland

Jean Le Feuvre Telecom ParisTech – LTCI, Paris, France

Ketan Mayer-Patel University of North Carolina, Chapel Hill, NC, USA

Britta Meixner CWI, Science Park 123, Amsterdam, Netherlands

Hugh Melvin School of Computer Science, National University of Ireland, Galway, Republic of Ireland

Mario Montagud Departamento de Comunicaciones, Universitat Politècnica de València (UPV) – Campus de Gandia, Valencia, Grao de Gandia, Spain; Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

Mu Mu The University of Northampton, Northampton, United Kingdom

Gabriel-Miro Muntean Department of Electronics & Informatics, Athlone Institute of Technology, Westmeath, Ireland; Department of Electronic Engineering, Dublin City University, Dublin, Ireland

Niall Murray Department of Electronics & Informatics, Athlone Institute of Technology, Westmeath, Ireland

Klara Nahrstedt University of Illinois at Urbana-Champaign, Champaign, USA

Wei Tsang Ooi National University of Singapore, Singapore, Singapore

Stefan Petscharnig Institute of Information Technology, Klagenfurt, Austria

Michael Probst IRT, Munich, Germany

Yuansong Qiao Department of Electronics & Informatics, Athlone Institute of Technology, Westmeath, Ireland

Kjetil Raaen Westerdals – Oslo School of Art, Communication and Technology, Oslo, Norway

Alexander Raake Audiovisual Technology Group – Institute of Media Technology – Ilmenau University of Technology, Ilmenau, Germany

Benjamin Rainer Austrian Institute of Technology (AIT), Seibersdorf, Austria

Michael Riegler Simula Research Laboratory, Fornebu, Norway

Mukesh Kumar Saini Indian Institute of Technology Ropar, Rupnagar, India

Marwin Schmitt Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands

Katrin Schoenenberg Clinical Psychology and Psychotherapy, University of Wuppertal, Wuppertal, Germany

Jonathan Shannon School of Computer Science, National University of Ireland, Galway, Republic of Ireland

Mya Sithu Nagoya Institute of Technology, Nagoya, Japan

Janto Skowronek Audiovisual Technology Group – Institute of Media Technology – Ilmenau University of Technology, Ilmenau, Germany

Kevin Stanton Intel Corporation, Santa Clara, USA

Ralf Steinmetz Technische Universitat Darmstadt, Darmstadt, Germany

Håkon K. Stensland Simula Research Laboratory, Fornebu, Norway

Tim Stevens British Telecommunications PLC, London, UK

Hans Stokking TNO, The Hague, The Netherlands

Christian Timmerer Institute of Information Technology, Klagenfurt, Austria

Jeroen Vanattenhoven KU Leuven, Leuven, Belgium

M. Oskar van Deventer TNO, Den Haag, Netherlands

Lourdes Beloqui Yuste Ericsson, Athlone, Ireland

Christoph Ziegler IRT, Munich, Germany

List of Acronyms

3-D	Three Dimensions
A/V	Audio–Video
AAC	MPEG-4 Advanced Audio Coding
ACR	AGSolute Category Rating
ADAMS	Adaptive Mulsemedia Framework
ADC	Asset Delivery Characteristics
AI	Artificial intelligence
AIT	Application Information Table
AMP	Adaptive Media Playout
AP	Aggregation Point
API	Application Programming Interface
App	Application (e.g., running on an HbbTV terminal)
AppCatUI	Application Catalog User Interface
AQM	Active Queue Management
AR	Augmented reality
ASP	Adaptive Synchronization Protocol
ATCP	Aggregate Transmission Control Protocol
AU	Access Unit
AUduration	Access Unit duration
AV	Audio and/or video
AVB	Audio–Video Bridging
AVC	Advanced Video Coding
AVTP	Audio–Video Transport Protocol
BAT	Bouquet Association Table
BBC	British Broadcasting Corporation
BER	Bit Error Rate
B-frame	Bi-predictive frame
BIFS	Binary Format for Scene
BML	Broadcast Markup Language
CAexp	Clock Accuracy Exponent

CAfreq	Clock Accuracy Frequency
CAint	Clock Accuracy Integer
CAT	Conditional Access Table
CAVE	Caver Automatic Virtual Environment
CDN	Content Delivery Network
CENC	Common Encryption
CG	Computer Graphic
CI	Composition Information
CIDL	Control Information Description Language
CII	Content Identification and other Information (DVB-CSS)
CIKR	Critical Infrastructure and Key Resources
CL	Compression Layer (CL)
CM	Congestion Manager
CoD	Content on Demand
CP	Coordination Protocol
CPU	Central Processing Unit
CRI	Clock Relation Information
CS	Companion Screen (HbbTV)
CSA	Companion Screen Application (DVB-CSS)
CSCW	Computer-Supported Collaborative Work
CSS	Cascading Style Sheet (HTML)
CSS	Companion Streams and Screens (DVB)
CTS	Composition Time Stamp
CUDA	Compute Unified Device Architecture
CUduration	Composition Unit duration
DAB	Digital Audio Broadcasting
DAE	Declarative Application Environment (OIPF)
DASH	Dynamic Adaptive Streaming over HTTP
DCS	Distributed Control Scheme
DCT	Discrete cosine transform
DE	Development Environments
DIAL	Discovery And Launch
DL	Delivery Layer
DoF	Degree of freedom
DOM	Document Object Model (HTML)
DoS	Denial of Service
D-PLL	Digital Phase-Locked Loop
DRM	Digital Rights Management
DSL	Domain-Specific Language
DSMCC	Digital Storage Media Command and Control
DTS	Decoding Time Stamp
DTT	Digital Terrestrial Television
DTV	Digital Television
DVB	Digital Video Broadcasting

DVB SI	DVB Service Information
DVB-C/C2	DVB Cable
DVB-H	DVB Handheld
DVB-IPTV	DVB Internet Protocol TV
DVB-S/S2	DVB Satellite
DVB-T/T2	DVB Terrestrial
DWT	Discrete wavelet transform
ECMA	European Computer Manufacturers Association
EDGE	Enhanced Data rates for GSM Evolution, typically lowest level of Internet connectivity for smart phones
EIT	Event Information Table
EIT p/f	Event Information Table—present/following
EPG	Electronic Program Guide
ES	Elementary Stream
ESCR	Elementary Stream Clock Reference
ESCRbase	ESCR base
ESCRext	ESCR extension
ESD	Elementary Stream Descriptor
FB	Fullband (20–20000 Hz)
FCS	Finite Coordinate System
FEC	Forward Error Correction
FoV	Field of View
FPGA	Field-Programmable Gate Array
fps	Frames Per Second
FQTSS	Forwarding and Queuing Time-Sensitive Streams
FSP	Flow Synchronization Protocol
FTSP	Flooding Time Synchronization Protocol
GM	Grand Master
GNSS	Global Navigation Satellite System
GoP	Group of Pictures
GPS	Global Positioning System
GPU	Graphics Processing Unit
GWAP	Games with a purpose
HAS	HTTP Adaptive Streaming over TCP
HBB	Hybrid Broadcast & Broadband
HbbTV	Hybrid Broadcast Broadband Television
HbbTV	Hybrid Broadcast Broadband TV HTTP
HCI	Human–computer interactions
HD	High Definition
HDMI	High-Definition Multimedia Interface
HDR	High-Dynamic Range
HDTV	High-Definition Television
HEVC	High Efficiency Video Coding
HF	High Force
HFSC	Hierarchical Fair-Service Curve

HLS	HTTP Live Streaming
HMM	Hidden Markov Model
HTG	Hypermedia Temporal Graph
HTML	Hypertext Markup Language
HTML5	Hypertext markup language, version 5
HTTP	Hypertext Transfer Protocol
IBB	Integrated Broadcast Broadband
ICT	Information and Communications Technology
ID	Identifier
IDES	Inter-Device Synchronization
IDMS	Inter-Destination Multimedia Synchronization
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
I-frame	Intra-frame
IG	Interest Group
IGMP	Internet Group Management Protocol
IoE	Internet of Everything
IoT	Internet of Things
IP	Internet protocol
IPTV	Internet Protocol Television
IRT	Institut für Rundfunktechnik GmbH
ISAP	Index for the first SAP
ISAU	Index for the first AU
ISDB	Integrated Services Digital Broadcasting
ISO	Inter-Destination Multimedia Synchronization Session Object
ISO	International Standards Organization
ISOBMFF	ISO-based Media File Format (MPEG)
ISOs	Inter-Destination Multimedia Synchronization Session Objects
ISP	Internet Service Provider
ITU	International Telecommunications Union
IU	Information Unit
JS	JavaScript (ECMAScript): programming language of the Web
JSON	JavaScript object notation
LAN	Local Area Network
LASeR	Lightweight Application Scene Representation
LDU	Logical Data Unit
MCP	Mutliflow Conversation Protocol
MCU	Multipoint Control Unit
MDU	Media Data Unit
MFA	Mulsemedia Flow Adaptation
MFU	Media Fragment Unit

MHEG	Multimedia and Hypermedia Expert Group
MHP	Multimedia Home Platform (DVB)
MIME	Multipurpose Internet Mail Extension
MJD	Modified Julian Date
MMOG	Massive Multiplayer Online Game
MMT	MPEG Media Transport
MOG	Multiplayer Online Game
MOS	Mean Opinion Score
MP2T	MPEG-2 Transport Stream
MPD	Media Presentation Description (MPEG-DASH)
MPEG	Motion Picture Experts Group
MPEG-2 PSI	MPEG-2 Program Specific Information
MPEG2-TS	MPEG 2 Transport Stream
MPEG2TSPCRInfoBox	MPEG-2 TS PCR Information Box
MPEG-4 CL	MPEG-4 Compression Layer
MPEG-4 DL	MPEG-4 Delivery Layer
MPEG-4 SL	MPEG-4 Sync Layer
MPEG-DASH	Dynamic Adaptive Streaming over HTTP
MPEG-MMT	Moving Pictures Expert Group MultiMedia Transport
MPU	Media Processing Unit
MRS	Material Resolution Server (DVB-CSS)
MS	Master/Slave Scheme
MSAS	Media Synchronization Application Server (DVB-CSS)
MTP	Media Time Stamp or Media Time
MUs	Media Units
NAT	Network Address Translation
NB	Narrowband (300–3400 Hz)
NCL	Nested Context Language
NCM	Nested Context Model
NFV	Network Function Virtualisation
NG-MS	Next-generation multimedia systems
NID	Network Identifier
NIT	Network Information Table
NTP	Network Time Protocol
NVE	Networked Virtual Environment
OCR	Object Clock Reference
OCRlen	OCR field length
OCRres	OCR resolution
OCXO	Oven-Controlled Crystal Oscillator
OD	Object Descriptor
OD	Olfactory Display
OIPF	Open IPTV Forum
OPCR	Original Program Clock References
OPCRbase	OPCR extension
OPCRext	OPCR base

OS	Operating System
OSI	Open Systems Interconnection
OTB	Object Time Base
OTT	Over The Top
P2P	Peer to Peer
PAT	Program Association Table
PBDS	Private Base Data Structure
PCR	Program Clock Reference
PCR	Program Clock Reference (MPEG-2 Transport Stream)
PCRbase	Program Clock Reference Base
PCRext	Program Clock Reference Extension
PES	Packetized Elementary Stream
PES	Packetized Elementary Stream (MPEG-2 Transport Stream)
P-frame	Predictive frame
PID	Program ID
PLL	Phase-Locked Loop
PMT	Program Map Table
PMT	Program Map Table (MPEG-2 Transport Stream)
POMDP	Partially Observable Markov Decision Process
ppm	Parts per million
PPS	Packet Priority Scheduling
PR	Period Relative (MPEG-DASH)
PRs	Point Relations
PSB	Public Sector Broadcasting
PSN	Packet Switched Networks
PTM	Precision Time Measurement
PTP	Precision Time Protocol
PTS	Presentation Time Stamp
PTS	Presentation Time Stamp (MPEG-2 Transport Stream)
PTSD	Post-Traumatic Stress Disorder
PTZ	Pan–Tilt–Zoom
PU	Presentation Unit
QoE	Quality of Experience
QoS	Quality of Service
QUIC	Quick UDP Internet Connection
RAP	Random Access Point
RBS	Reference Broadcast Synchronization
REST	Representational State Transfer
RET	Retransmission
RFC	Request for Comments
RG	Residential Gateway
RGB	Color space using the dimensions red, green, and blue
RoI	Region of Interest
RR	RTCP Receiver Report

RST	Running Status Table
RTC	Real-Time Communication
RTCP	Real Time Control Procotol
RTCP	Real-Time Transport Control Protocol
RTCP	RTP Control Protocol
RTD	Round-Trip Delay
RTP	Real-Time Transport Protocol
RTP/RTCP	Real-Time Transport Protocol / RTP Control Protocol
RTS	Real-Time Systems
SAP	Stream Access Point
SAU	Stream Access Unit
SC	Synchronization Client (DVB-CSS)
SCD	System Clock Descriptor
SCF	System Clock Frequency
SCFMPEG-2	System Clock Frequency of MP2T streams
SCTE	Society of Cable Telecommunications Engineers
SD	Standard Definition
SDES RTCP	Source DEScription packet
SDI	Serial Digital Interface
SDL	Specification and Description Language
SDN	Software Defined Networking
SDT	Service Description Table
SEDL	Sensory Effect Description Language
SEM	Sensory Effect Metadata
SEV	Sensory Effect Vocabulary
SIDX	Segment Index Box
SIFT	Scale-Invariant Feature Transform
SL	Sync Layer
SL	Synchronization Layer
SLA	Service Level Agreement
SMIL	Synchronized Multimedia Integration Language
SMPTE	Society of Motion Picture and Television Engineers
SMS	Synchronization Maestro Scheme
SOTA	State of the Art
SQA	Subjective quality assessment
SR RTCP	Sender Report
SRD	Spatial Relationship Description
SRP	Stream Reservation Protocol
SSDP	Simple Service Discovery Protocol
SST	Structure Stream Transports
ST	Stuffing Table
STB	Set Top Box
STB	System Time Base
STC	System Time Clock
STD	System Target Decoder

STUN	Session Traversal Utilities for NAT
SVG	Scalable Vector Graphics
SYMM	Synchronized Multimedia working group
Synch	E Synchronous Ethernet
TAACCS	Time Aware Applications, Computers and Communications Systems
TAI	International Atomic Time
tC	Composition time
TCP	Transfer Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TCXO	Temperature Compensated Crystal Oscillator
tD	Decoding time
TDT	Time and Date Table
TE	Trigger Event (DVB-CSS)
TEMI	Timeline and External Media Information
TEMI	Timeline for External Data (MPEG)
TEMI_AU	TEMI Access Unit
TI	Tele-immersion
TLS	Transport Layer Security
TOT	Time Offset Table
Tp MPEG-DASH	Media Segments Presentation Time
TPSN	Timing-sync Protocol for Sensor Networks
TR	Transport Rate
TS MPEG-2	Transport Stream
TS	Timeline Synchronization (DVB-CSS)
TSC	Time Stamp Counter
TSlen	Time Stamps field length
TSN	Time-Sensitive Networking
TSres	Time Stamp resolution
T-STD	Transport-System Target Decoder
TTL	Time-to-live
TV	Television
UDP	User Datagram Protocol
UI	User interface
UK	United Kingdom
UML	Unified Modeling Language
UPNP	Universal Plug and Play
URL	Unified Resource Locator
UTC	Coordinated Universal Time
UTF-88-bit	Unicode Transformation Format
v/b	video broadcast object (HbbTV)
VANC	Vertical Ancillary Data
VBI	Vertical Blanking Interval
VCO	Voltage-Controlled Oscillator
VCR	Video Cassette Recorder

Vcxo	Voltage-Controlled Crystal Oscillator
VLAN	Virtual Local Area Network
VoD	Video On Demand
VR	Virtual Reality
VTR	Virtual-Time Rendering
W3C	World Wide Web Consortium
WB	Wideband (50–7000 Hz)
WC	Wall Clock (DVB-CSS)
WiFi	Wireless local area networking
WSN	Wireless Sensor Network
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XMT	Extensible MPEG-4 Textual Format
YUV	Color space using the dimensions luminance (Y) and chrominance (U, V)

List of Figures

Introduction to the MediaSync Book

Media synchronization, or mediasync in short, has been a key research area since the early development of (distributed) multimedia systems, mainly due to the best-effort nature of delivery networks and the imperfect behavior of the involved devices.

Many advances around mediasync have been devised and deployed up to date, but it still requires the attention of the research community. Core challenges need to be overcome to fulfill the requirements of the next-generation heterogeneous and ubiquitous media ecosystem.

Mediasync is a mature and broad research area, involving different disciplines: from media production to document engineering, from theoretical models to human factors, from networking issues to multimedia systems. Understanding the mediasync-related requirements and challenges in today's multisensory, multi-device and multi-protocol world is not an easy task. Although providing a full coverage of this research area is a big challenge, this book provides an approachable overview of mediasync from different perspectives, gathering contributions from the most representative and influential experts. It revisits the foundations of mediasync, including theoretical frameworks and models, compiles the most relevant advances, highlights ongoing research efforts, and paves the way for the future (e.g., toward the deployment of multisensory and ultra-realistic experiences).

The book has been structured in five parts. *Part I: Foundations* focuses on the fundamentals of mediasync. It introduces the research area, its relevance, its evolution, and key aspects. This initial part, as a whole, gives readers a global notion of the research area. *Part II: Applications, Use Cases, and Requirements* includes research studies and contributions for specific use cases in which mediasync becomes essential. Many other important application areas of mediasync exist, with different requirements and involving a variety of challenges. Considering all of them in the book would have been difficult, so representative use cases have been selected for their inclusion in the book: Networked Virtual Environments (NVEs) and games, novel video production paradigms like mashups, and Computer-Supported Cooperative Work (CSCW) systems. *Part III: User Experience and Evaluation Methodologies* discusses the relevance and the challenges of

taking into account human factors in mediasync and of subjectively evaluating mediasync solutions, with a focus on conferencing, TV, and multisensory services. *Part IV: Document Formats and Standards* reviews document formats and standards providing mediasync capabilities, like Moving Picture Experts Group (MPEG) and Hybrid Broadcast Broadband Television (HbbTV). Finally, *Part V: Protocols, Algorithms, and Techniques*, describes relevant standard and proprietary protocols, specific algorithms and control techniques that are relevant within the mediasync research area.

It is advisable to read the parts of the book from beginning to end, especially for *Part I*, to acquire a deeper and comprehensive understanding of the research area. However, the chapters are self-contained and can be read independently of the other chapters, based on specific interests.

The following paragraphs give more details about the specific contributions of each one of the chapters in each part of the book.

Part I: Foundations

Chapter 1 introduces key definitions and provides a categorization of the existing mediasync types, highlighting their relevance. It also identifies essential aspects and components of mediasync solutions.

Chapter 2 reflects how the evolution of multimedia systems has brought new challenges in terms of mediasync. It also reviews the most relevant contributions focused on overcoming/understanding these challenges. Finally, the chapter proposes a reference framework to address emerging requirements of next-generation multimedia systems.

Chapter 3 provides an overview, formal definitions, and an analysis of temporal models used to specify constraints and requirements for synchronization, scheduling, management, and presentation of interactive multimedia documents, composed of linked media elements.

Chapter 4 provides a comprehensive overview of timing, which is an essential aspect in most of the existing mediasync solutions. The chapter introduces key concepts, describes the most popular technologies and protocols related to timing, and reviews the most significant challenges to ensure precise timing in current systems and applications.

Part II: Applications, Use Cases, and Requirements

Chapter 5 highlights the relevance of providing synchronization between distributed participants in NVEs and games of both competitive and collaborative nature. It provides a review and analysis of existing techniques to achieve this goal

in these environments, comparing these techniques in terms of fairness, efficiency, consistency, and interactivity.

Chapter 6 introduces key challenges and building blocks to provide automated time-synchronous video mashups: concatenations of video segments from the same event recorded by different cameras, referred to a common timeline. It also reviews the state of the art and outlines a set of remaining challenges in this application area.

Chapter 7 describes and characterizes the mediasync-related challenges and requirements in CSCW systems. It also reviews a variety of streaming and coordination protocols, mechanisms, and techniques that have been devised up to date to overcome the associated challenges in these CSCW systems.

Part III: User Experience and Evaluation Methodologies

Chapter 8 focuses on the users' tolerance to delays in interactive computer systems, reviewing relevant studies and methodologies from three research disciplines: cognitive psychology, Human–Computer Interaction (HCI) and multimedia systems.

Chapter 9 gives an overview of the challenges in evaluating delays and mediasync in real-time audiovisual communications. It provides recommendations on how to subjectively test these systems and how the obtained results can be analyzed to deeply understand the impact of delays and of their variability.

Chapter 10 provides an in-depth discussion on the impact of mediasync and timing on the user experience within the TV consumption landscape, with a special focus on Social TV and multiscreen applications.

Chapter 11 reviews existing studies on mediasync for networked multisensory applications with haptics. The chapter highlights the relevance of human perception aspects in these scenarios, and proposes improvements to existing mediasync algorithms to take them into account. Finally, the chapter discusses future research directions on mediasync in these scenarios.

Chapter 12 discusses the wide applicability of olfaction-enhanced media services and focuses on two mediasync aspects in these services: the specification of temporal relationships and the implementation of end-to-end systems for a synchronized delivery and consumption of olfaction-enhanced media contents. The chapter also presents and discusses results of subjective studies, mainly focused on analyzing the users' tolerance to the lack of synchronization.

Part IV: Document Formats and Standards

Chapter 13 provides an overview of the timing and mediasync features of Synchronized Multimedia Integration Language (SMIL), which is a comprehensive

multimedia content integration language developed and maintained by the World Wide Web Consortium (W3C).

Chapter 14 presents the mediasync features of Nested Context Language (NCL) to support declarative specification of hypermedia applications. The chapter concentrates on describing the applicability of NCL within the fields of interactive TV and the Web.

Chapter 15 details how time-related information and mechanisms are conveyed and used in relevant MPEG and Digital Video Broadcasting (DVB) standards to achieve synchronized playout of media contents.

Chapter 16 focuses on explaining how the MPEG-4 standard manages the synchronization and playout of audiovisual streams and graphics animations and how multiple timelines can be used to provide rich interactive presentations, over both broadband and broadcast services.

Chapter 17 discusses the relevance of modularity, temporal composition, and inter-operability on the Web. It presents a programming model based on the use of external timing resources to enable these requirements and provide shared control, which has been proposed as a standard within the W3C. The chapter discusses relevant scenarios that can be provided by adopting that model, in both single- and multi-device settings, and includes evaluation results.

Chapter 18 provides an overview of the mediasync features provided by the HbbTV standard, some of them being adopted from the DVB Companion Screens and Streams (DVB-CSS) specification. These features enable the deployment of relevant scenarios, like Social TV, hybrid TV, and companion screen applications. The chapter also provides implementation guidelines.

Part V: Protocols, Algorithms, and Techniques

Chapter 19 analyzes the challenges, implications, and opportunities of the adoption of modern ubiquitous computing and IP-based technologies on the TV production and delivery industry, mainly focusing on the impact on delays.

Chapter 20 discusses the main mediasync-related challenges for the development of real-time video panorama recordings by using camera array systems. In particular, the chapter presents the main components of a sports analytics system developed in a soccer stadium and focuses on two challenges in the panorama generation process: shutter synchronization and exposure synchronization.

Chapter 21 presents a solution to provide synchronized playout between geographically distributed devices when using Dynamic Adaptive Streaming over HTTP (DASH) for media delivery. The chapter also introduces the use of crowdsourcing methodologies for subjective assessment in the area of mediasync.

Chapter 22 gives an overview and analyzes the applicability of two commonly used techniques for mediasync: *watermarking* and *fingerprinting*. These techniques

enable the identification of pieces of media content, and thus determining the current playout position of media streams, which are essentials tasks when performing mediasync.

Chapter 23 provides an overview and analyzes packet probing methods that can be used to measure different types of delays and the available network capacity in distributed multimedia systems. The chapter also shows how accurate delay measurements can result in an increased Quality of Experience (QoE) in immersive multi-device media applications.

Overall, *Mediasync: Handbook on Multimedia Synchronization* is the perfect companion for scholars and practitioners that want to acquire strong knowledge about theoretical, technological, and experimental aspects of this broad research area. It can also be used as a reference guide to approach the challenges behind ensuring the best mediated experiences, by providing the adequate synchronization between the media elements that constitute these experiences.

Extra materials about the book, including figures, slides, and updates are available at: https://sites.google.com/site/mediasynchronization/book.

Part I
Foundations

Chapter 1
Introduction to Media Synchronization (MediaSync)

Mario Montagud, Pablo Cesar, Fernando Boronat and Jack Jansen

Abstract Media synchronization is a core research area in multimedia systems. This chapter introduces the area by providing key definitions, classifications, and examples. It also discusses the relevance of different types of media synchronization to ensure satisfactory Quality of Experience (QoE) levels and highlights their necessity, by comparing the magnitudes of delay differences in real scenarios to the tolerable limits to the human perception. The chapter concludes with a brief description of the main aspects and components of media synchronization solutions, with the goal of providing a better understanding of this timely research area.

Keywords Intra-media synchronization · Inter-media synchronization Hybrid synchronization · Inter-device synchronization · Inter-destination media synchronization · Delays · Asynchrony · QoE

1.1 Introduction

Multimedia systems are characterized by the integration of various types of media components, like audio, video, text, and even multimodal data captured or generated by heterogeneous types of sensors. These media components can be captured or retrieved by one or more sources, delivered, and then consumed by one or more

M. Montagud (✉) · F. Boronat
Departamento de Comunicaciones, Universitat Politècnica de València (UPV) – Campus de Gandia, Valencia, Grao de Gandia, Spain
e-mail: mamontor@gmail.com

F. Boronat
e-mail: fboronat@dcom.upv.es

M. Montagud · P. Cesar · J. Jansen
Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
e-mail: P.S.Cesar@cwi.nl

J. Jansen
e-mail: jack.jansen@cwi.nl

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_1

receivers. Their processing and storage at both the source and receiver sides are also frequent.

There are two main categories of media: *continuous* (also known as time-dependent) and *static* (also known as noncontinuous, time-independent, or discrete). Typically, the information of each media component is modeled as a sequence of Media Units (MUs),[1] whose granularity is highly application-dependent. On the one hand, continuous media components, such as audio, video, or sensor data, are characterized by implicit and well-defined temporal relationships between subsequent MUs (e.g., audio samples and video frames). These temporal relationships are typically determined during the capturing, sampling, and/or encoding processes, and denote the original order and duration of MUs. On the other hand, static media components, such as text and images, have less strict temporal properties than continuous media components, and even their temporal attributes might not be implicitly known. For instance, a video stream (continuous media) is composed of a collection of subsequent frames or images that need to be presented in the same order and with the same duration in which they were captured. However, when presenting single and/or multiple independent images (static media), the capturing time could only be relevant to chronologically order them. Likewise, the duration of their presentation can be explicitly and flexibly set. However, the temporal attributes of static media components can be stricter if integration with other media components is required. For instance, in a multimedia presentation, static media components (e.g., images or text) need to be presented at the correct point of time, just before, after, or simultaneously with, other static or continuous media components (e.g., audio explanations), and during a specific period interval. The same applies for the integration of multisensory data with audiovisual contents. Therefore, within this perspective, a multimedia system can be defined as "*that system that supports the integrated processing of several media components, being at least one of them time-dependent (i.e. continuous)*" [1].

Apart from temporal relationships, the involved media components in a multimedia system can have spatial and semantic relationships. On the one hand, spatial relationships refer to the physical arrangement of the involved media capturing and presentation devices (e.g., arrays of microphones and loudspeakers, multiple TV cameras capturing different views of a scene, etc.) as well as the layout of media presentations. On the other hand, semantic relationships refer to the content dependency between media components (e.g., two graphics with different representations of the same data, data from an environment captured by heterogeneous multimodal sensors, etc.). Typically, media components with spatial and/or semantic relationships also have temporal dependences. For example, in a slide show, each slide can contain graphics, texts, animations, and even audio and video comments. In this case, the clarity of the presentation is highly influenced by the appropriate ordering, timing, layout, and semantics of the multimedia information

[1]Also known as Media Data Units (MDUs), Logical Data Units (LDUs), Information Units (IUs), and Access Units (AUs) in literature.

on the slides. If any of these dependences is not preserved, the slides can become difficult to understand, or can even convey incorrect information (e.g., audio comments referring to a graphic that has not been presented yet). Temporal and spatial synchronization is also essential in multiview video, where different cameras are capturing different point of views of an event or environment (e.g., a sports stadium), and in omnidirectional video, where the 360° area is typically divided and delivered in different Fields of Views (FoVs). Likewise, the multimodal or multisensory data collected by heterogeneous sensors in a Smart City, Internet of Things (IoT), or Virtual Reality (VR) scenario need to be presented on the proper devices, by taking into account their temporal, spatial, and semantic relationships. In all these examples, a precise and adaptive mechanism of integration, coordination, and organization is needed in order to ensure, during presentation, the proper dependences and evolution of the media components involved in a multimedia system. Such a process is referred to as *multimedia synchronization* (*mediasync, hereafter*).

Although mediasync can encompass the three above dimensions (content, space, and time), the temporal dependences need to be always taken into account. Continuous media components require fine-grained synchronization, while discrete media components are typically less restrictive, allowing more coarse-grained synchronization. Likewise, synchronization between continuous and discrete media components is also relevant.

Mediasync is required when media is captured/retrieved and consumed within specific devices (e.g., when playing out local media files), but its relevance increases in distributed multimedia systems, in which media sources and receivers are not co-located, especially when heterogeneous media contents, delivery technologies, and devices are involved in such systems. Typically, the temporal relationships for the involved media components are specified during the capturing/retrieval processes and inserted as metadata, as part of the MUs or of the delivery units. However, the required processes until their ultimate presentation, together with the imperfect performance of the involved networks, equipment and devices, will have an impact on such original relationships. For instance, when streaming continuous media over the Internet, the MUs of the involved media components will be typically captured at a specific MU rate (e.g., a video with 25 frames per second, *fps*), packetized and then conveyed into a single or multiple streams for transmission over the network. Such media streams will be sent (e.g., via unicast or multicast) by one or more sources to one or several receivers, which can (simultaneously) play out one or several media components. Each one of the packets of each involved media stream can follow the same or different paths to reach the receiver/s and can be differently affected by delays and jitter. Accordingly, apart from the temporal (possibly together with spatial and semantic) specification methods at the source side, adaptive mechanisms are needed to reconstruct the original timing for each of the incoming media streams at the receiver side, typically with the help of buffering and playout adjustment strategies. Even more, a proper collaboration between the involved entities in the end-to-end media distribution chain (e.g., sources, receivers, and even networking devices) can contribute to the preservation or reconstruction of these temporal dependences. All these

processes fall within the mediasync research space and are relevant to efficiently accomplish it. Mediasync must be addressed and supported by many system components, including hardware devices, Operating System (OS), protocol layers, multimedia formats and files, and even by applications. Hence, *mediasync is an end-to-end challenge* (i.e., from capturing/retrieval to consumption), which must be addressed at several levels in a multimedia system.

Mediasync has been a key research challenge since the early development of multimedia systems, due to the existence of variable delays and to the imperfect performance of the communication networks and the involved devices. On the one hand, mediasync is tightly coupled to the Quality of Service (QoS) of multimedia systems. It is due to the fact that the existence of different QoS factors, such as delays, jitter, bandwidth variability, and packet loss, will prevent from a proper presentation of media contents. In such cases, mediasync solutions can be employed to minimize the impact of such factors. On the other hand, the lack of mediasync has a significant impact on the user's perceived Quality of Experience (QoE). Examples of QoE factors related to mediasync are intelligibility, satisfaction, annoyance, and networked togetherness. The relevance of specific QoE factors for mediasync will be strongly influenced by many aspects, such as the specific type of mediasync (explained later), the targeted scenario, the specific use case (together with its context and evolution), as well as human perception factors. A list of QoS factors and QoE aspects that are relevant within the mediasync research area is provided in Fig. 1.1. Their relevance and relationship with mediasync will become much clearer after reading the different chapters of the book.

This chapter aims at introducing the mediasync research area. It has been structured as follows. In Sect. 1.2, different types and variants of mediasync are identified, by providing examples of relevant scenarios and highlighting the relevance of mediasync in each one of them. Next, Sect. 1.3 provides an overview of the different factors along the end-to-end media distribution chain that can contribute to

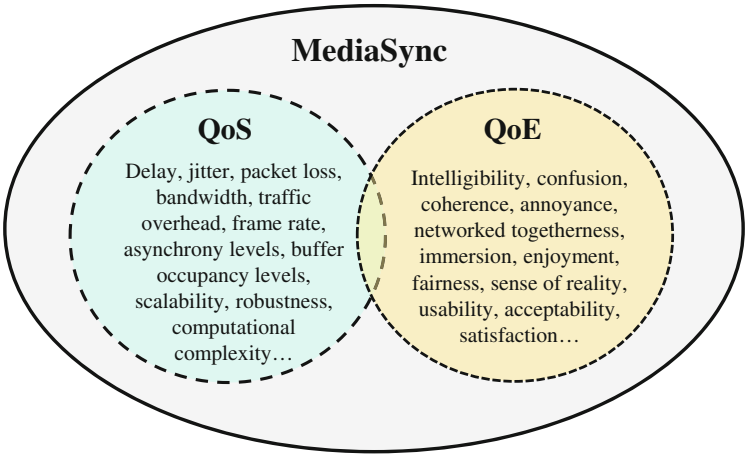


Fig. 1.1 Relevance of QoS factors and QoE aspects on MediaSync

the increase of delays and delay differences in current-day multimedia systems, thus having an impact on mediasync. The magnitudes of delays and delay differences in real scenarios are also reviewed, being compared to the tolerable limits to the human perception in different mediasync types and applications, with the goal of reflecting the necessity of mediasync. Section 1.4 identifies and briefly describes the main aspects and components of mediasync solutions. This will provide a deeper understanding of the existing mediasync solutions and also help comprehending why a variety of them exists. Finally, Sect. 1.5 provides a summary of the chapter.

1.2 Media Synchronization Types and Examples

Two main types of mediasync can be distinguished, based on the real-time nature of the media contents and on temporal aspects: live and synthetic synchronization. In live synchronization, the media contents are captured in real time by sensors (e.g., cameras, microphones, etc.). In synthetic synchronization, the media contents are retrieved from storage systems, although probably they were originally "live" captured and then stored. In the former case, the capturing and playout processes are performed during a continuous temporal process, with a delay as short as possible. In the latter case, MUs are captured, stored, and played out at a later point of time. Both types of synchronization attempt to preserve the original temporal dependences within and between media components. However, such temporal relationships can be altered in synthetic synchronization, according to the available resources or targeted requirements (e.g., by changing the transmission or playout rate). Synthetic synchronization is easier to achieve than live synchronization, because of the softer, or even inexistent, real-time requirements and higher flexibility (e.g., it is possible to adjust the transmission and playout rates, to schedule the initial playout times as desired, the media contents can be processed with lower delay constraints, etc.). Likewise, live synchronization can include both uni- and bidirectional communications, being the latter more challenging due to the non-symmetric performance of current networks and the involved interaction patterns. Internet Radio is an example of a use case requiring unidirectional live synchronization, while multiparty conferencing is an example of a use case requiring bidirectional live synchronization. Synthetic synchronization is often necessary in retrieval-based services, such as Content on Demand (CoD).

In both live and synthetic synchronization, different types and variants of mediasync techniques can be distinguished (see Fig. 1.2). Such classification is mainly based on the number of the involved media components, streams, sources, and receivers.

First, *intra-media synchronization*[2] is concerned with maintaining (and re-establishing, if necessary), during playout, the original temporal relationships among subsequent MUs composing each individual media component (e.g., video

[2]Also known as intra-stream and serial synchronization in literature.

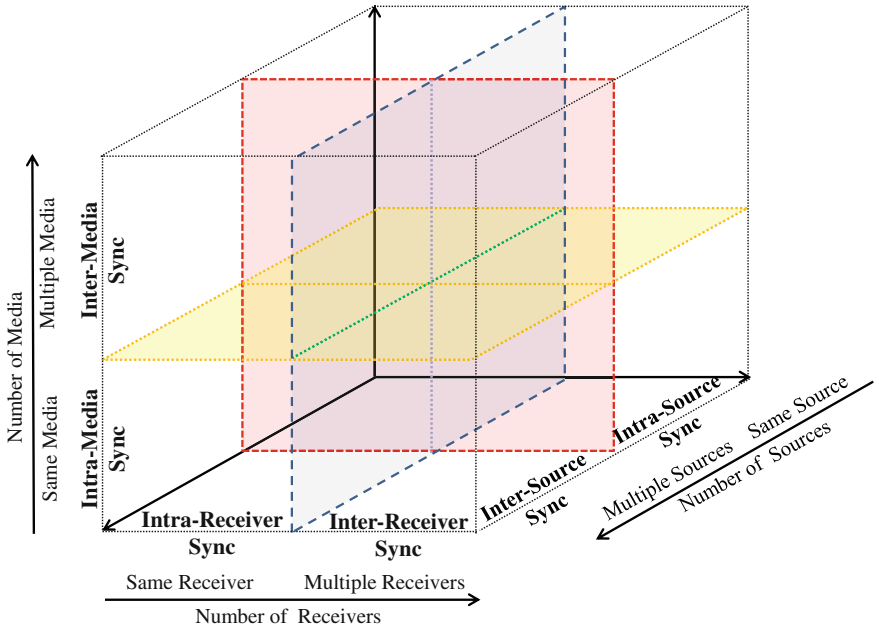


Fig. 1.2 Classification of MediaSync types

frames, audio samples, etc.). Its main goal is that the temporal relationships between MUs during presentation resemble as much as possible to the ones specified during the capturing/sampling/retrieval, or even encoding, processes at the source side, despite of the existence of delays and delay differences along the end-to-end distribution chain. To achieve this kind of synchronization in distributed multimedia systems, playout buffering strategies are typically employed at the receiver side to smooth out the effects of jitter or delay variability (and even of packet loss, by adopting error recovery or re-transmission techniques). On the one hand, the size of the playout buffer must be large enough to compensate for the effects of jitter. On the other hand, the buffering delays need to be as short as possible in order to minimize the latency of the multimedia service. Likewise, the playout buffer occupancy must be kept into stable and safe ranges, with the goal of minimizing the occurrence of buffer overflow and underflow situations. This way, a continuous and smooth playout for each media component can be guaranteed. Otherwise, the lack of intra-media synchronization can cause temporal distortions in the playout process of each involved media component. For example, it can result in playout discontinuities or interruptions (e.g., image jerkiness, images that will not be shown, audio distortion, etc.) and/or MUs presented out-of-order, which will not reflect a natural and consistent evolution of the media playout, resulting in confusion, unintelligibility, and/or users' annoyance (i.e., bad QoE). Therefore, adaptive mechanisms are necessary to guarantee intra-media synchronization, which is a fundamental aspect in every type of multimedia service involving playout of media contents.

The concept of intra-media synchronization, and of other mediasync types, when being applied to a unidirectional audiovisual service, is illustrated in Fig. 1.3. In the figure, video, data (e.g., text-based), and audio streams from an online football match are being played out by a group of receivers distributed over an IP network. In that scenario, it is first necessary to provide intra-media synchronization for each of the involved streams, ensuring proper and natural playout processes, such as the evolution of a video sequence showing a jumping ball. As an example, if the media source captures a video sequence at 25 MUs (video frames) per second, each MU must be played out (displayed) during 40 ms at the receiver side (see duration for each MU in the figure).

When several correlated media components are involved in a multimedia system, the original dependences between their MUs must be also preserved during playout. That is the goal of *inter-media synchronization.*[3] However, it is not easy to achieve when the different media components present heterogeneous characteristics (e.g., type of media, quality, etc.) and requirements (e.g., processing, bandwidth, etc.). Likewise, the inter-media synchronization process can have an impact on the intra-media synchronization processes for specific media components, so a proper tradeoff and coordination between them must be ensured to satisfy both necessities and provide adequate QoE levels.

Several use cases of inter-media synchronization can be mentioned. The most popular one is audio/video synchronization. In such a case, video frames and audio samples captured at the same time must be also simultaneously presented at the corresponding output devices (ideally with the minimum latency in time-sensitive services). For example, in remote user's speech, the synchronization between audible words and the associated movement of the lips is necessary. This is commonly known as *lip synchronization* or *lip-sync* (e.g., [2] and [3]). A similar inter-media synchronization scenario is when using a two-lens stereo camera system with an internal microphone. In such a case, if only 2D video streaming is required, the synchronization between the audio and the video content from one of the lens will be sufficient. However, if 3D video streaming is required, synchronization between the video content from the two lenses will be also necessary. A third use case is the synchronization of subtitles with the associated audiovisual contents (e.g., [4]). In addition, inter-media synchronization is not only limited to the synchronization between audiovisual contents, but the synchronization of audiovisual contents with different combinations of multisensory media contents (i.e., contents or effects that stimulate other senses beyond hearing and sight, such as olfaction, gustatory, and touch), and between multisensory contents, also becomes very relevant in many scenarios. The first example is the synchronization between audiovisual contents with haptic media, especially applicable in Networked Virtual Environments (NVEs) and games (e.g., [5]), as explained in Chap. 11. The second example is the synchronization between audiovisual contents with olfactory data (i.e., computer-generated scent) [6, 7], as explained in Chap. 12.

[3]Also known as inter-stream and parallel synchronization in literature.

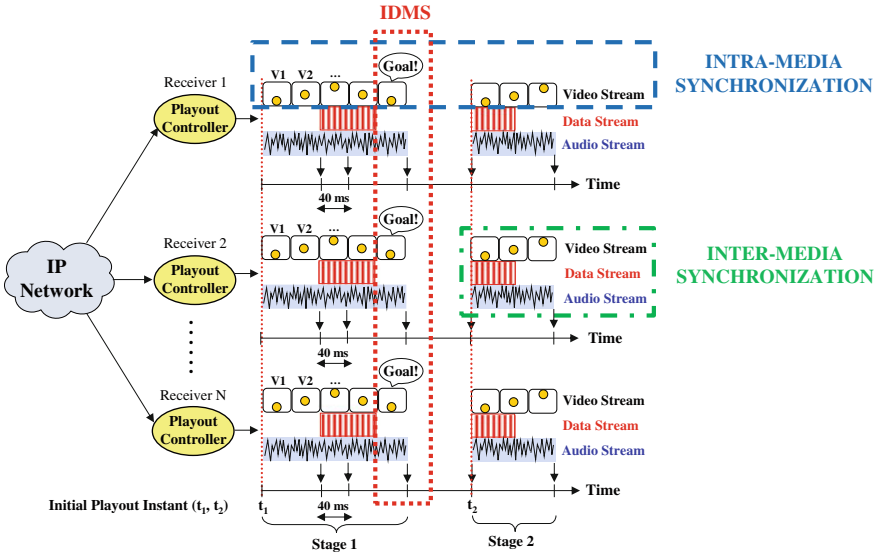


Fig. 1.3 Illustrative examples of MediaSync types

Similarly, synchronization becomes essential when capturing data from other types of heterogeneous sensors, such as environmental, motion, and physiological sensors, in a variety of scenarios, such as Smart Cities, IoT, and e-health. The lack of inter-media synchronization can originate confusion, inconsistent media presentations, and even incorrect information when trying to associate or correlate the data from the involved sensors. This will result in a bad QoE, or even in useless services. Examples of lack of inter-media synchronization are audio comments referring to incorrect video scenes or graphics, subtitles that do not match with the audio and scenes being presented, olfactory/haptic stimuli presented in combination with unrelated audiovisual contents, inconsistency and un-correlation between the data from different physiological sensors, etc.

In all these inter-media synchronization use cases, the involved media components can be of the same or different modalities. For instance, synchronization of media components of the same modality is required when arrays of capturing and output devices are involved. Such arrays of sensors and output devices typically capture and present, respectively, media content from different positions or angles, with the goal of providing enriched and immersive media experiences. As these two cases may present different requirements in terms of implementation and user's perception, related works use different terms to refer to them. For example, in [8] (see Chap. 2), the term intra-media synchronization is used to refer to the synchronization between media components of the same modality, while a new term, *intra-bundle synchronization*, is introduced to refer to the synchronization between media components of different modalities.

Likewise, two main approaches for inter-media synchronization can be distinguished. The first one is when the (elementary streams for the different) media components are multiplexed into a single transport stream[4] by the media source (see Fig. 1.4a.1). The second one is when the different media components are delivered in different streams (see Fig. 1.4a.2). Inter-media synchronization is easier to be provided in the former case than in the latter one. It is due to the fact that the involved media components are delivered using a single transport stream, thus experiencing the same delay to reach the receiver(s), and can be more easily identified and linked. However, the latter approach provides higher flexibility and efficiency with respect to bandwidth usage. For example, a user can choose to receive only the media components of interest, or to momentarily stop the reception of a specific media component (considered less important) in case of congestion. The delivery of media components as aggregated transport streams is commonly employed in Moving Picture Experts Group (MPEG) technologies (e.g., MPEG 2 Transport Stream (MPEG2-TS)), while the delivery of individual streams for each media component is more common when using Real-Time Transport Protocol/RTP Control Protocol (RTP/RTCP) [9].

Inter-media synchronization can also involve the synchronization of media components originated from different sources, or delivered by different senders. This specific subtype of inter-media synchronization is typically referred to as *inter-source synchronization*[5] (see Fig. 1.4b). A typical example is the synchronization between video streams sent by different video servers (e.g., in surveillance systems). Moreover, it can also be possible that the different media components, from either the same or different sources/senders, are delivered using different protocols, or even via different (e.g., broadcast and/or broadband) technologies or networks, of the same or different characteristics. In such scenarios, the term *hybrid synchronization* is commonly used in literature (e.g., [10–13]). Hybrid synchronization allows leveraging the particular advantages of broadcast and broadcast networks for media delivery and consumption, by using them in a coordinated manner. For example, the contents provided via broadcast networks (mainly characterized by scalability and ubiquity) can be augmented by additional related contents provided via broadband networks (mainly characterized by bidirectionality, adaptability, flexibility, and low-cost bandwidth), providing interactive, personalized, and enriched services. This is especially relevant within the TV consumption area, from which two representative examples can be cited. The first one is the simultaneous playout of the video streams from a sports event being transmitted by broadcast (e.g., via Digital Video Broadcasting or DVB) and broadband TV operators (e.g., via Internet Protocol Television (IPTV) or via Dynamic Adaptive Streaming over HTTP (DASH)). The second one is the synchronization of real-time statistics or subtitles from an Internet server with audiovisual contents from a broadcast TV program.

[4]Also known as aggregated stream or bundle in literature.

[5]Also known as multisource or inter-sender synchronization in literature.

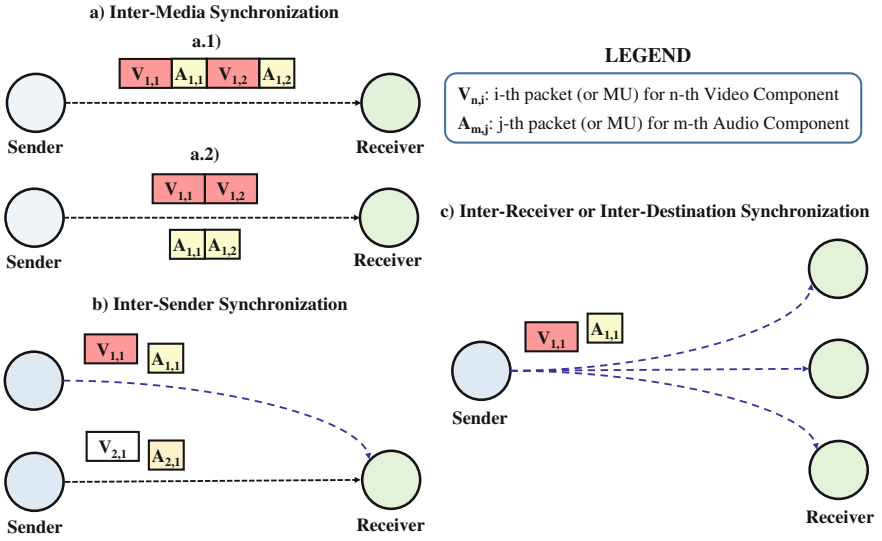


Fig. 1.4 Different types and variants of MediaSync

The involved media components, regardless of their modality, number, and combinations, can be played out in a synchronized manner by single or multiple receivers/devices (see Figs. 1.2, 1.3 and 1.4). When multiple receivers/devices are involved, the terms *inter-receiver* and *Inter-Device Synchronization* (IDES) are typically employed. Likewise, the involved receivers/devices playing out the same or related contents can be physically close-by (e.g., in the same local network) or geographically distributed (e.g., in different buildings, cities, or countries). Examples of the former situation are multiscreen scenarios (e.g., [14]) and buildings with multiple distributed loudspeakers. In such cases, the absence of IDES will result in incoherent multiscreen experiences and in the perception of annoying echo effects, respectively. When the involved devices are far apart, the term *Inter-Destination Media Synchronization* (IDMS)[6] is typically employed to refer the synchronization between their playout processes. IDMS is essential to enable coherent shared media experiences between remote users [15], such as multiparty conferencing, Social TV, and networked Multiplayer Online Games (MOGs). Such type of synchronization aims to guarantee that all the involved users in a shared session perceive the same events (i.e., play out the same pieces of media contents) at the same time, despite the existence of delay differences between them. As an illustrative example of IDMS, it can be noticed in Fig. 1.3 that all the receivers are playing the same MU of each media component at any moment during the multimedia session. In the presented online football match or "*watching apart together*" scenario, IDMS aims to guarantee that all the involved friends perceive the important events and actions

[6]Also known as group, multipoint and inter-participant synchronization in literature.

almost simultaneously. The absence of IDMS in that scenario will result in inconsistent interactions (e.g., via text/audio/video chat) and frustrating situations, such as being aware of a goal through the cheering of a friend via the chat channel, before the goal sequence is displayed on the local screen, spoiling the shared experience [16]. In other use cases, like MOGs and NVEs, the absence of IDMS can also result in unfairness, for competitive scenarios, or lower efficiency, for collaborative scenarios.

Likewise, a proof of evidence of the increasing relevance of hybrid synchronization and IDES is the recent Hybrid Broadcast Broadband TV (HbbTV) standard [17], which harmonizes the delivery and consumption of related hybrid broadcast (DVB) and broadband (DASH and HTML5) contents, both on single devices and on multiple devices in multiscreen scenarios. Chapter 18 will describe the mediasync features provided by HbbTV.

1.3 Delays and MediaSync

As mentioned, different QoS factors have an impact on mediasync, from which delays, and especially their variability, must be emphasized. Processing and delivery of media contents in current multimedia systems are not instantaneous but take some time. This is especially noteworthy in digital communications. Moreover, present-day IP-based networks provide no guarantees, either on delay or on delay variability bounds. Delays are not serious constraints when isolated users are consuming non-time-sensitive contents (e.g., in CoD services). However, delays and their variability for each individual media stream (i.e., jitter), between different related media streams, between different sources, and between different receivers, become serious barriers when tight real-time requirements must be met, and when interactivity between the users and the media content, as well as between users, are pursued.

This section first identifies various system components belonging to different steps of the end-to-end media distribution chain that can contribute to the increase of delays and of their variability, thus having a significant impact on mediasync. Then, it provides insights about the magnitudes of the delays and delay differences in real scenarios, both for single streams and between different streams sent to a single or to multiple receivers, according to measurements conducted in different studies. After that, it is shown that these magnitudes significantly exceed the levels of delay differences (i.e., asynchrony) tolerable to the human perception for each mediasync type, in different applications. This clearly reflects the need for adaptive and accurate mediasync solutions to compensate for these delay differences, with the goal of ensuring satisfactory QoE levels.

1.3.1 Delay and Delay Variability Factors

When delivering media, different factors and system components can contribute to the increase of delays and of their variability. Such sources of delay and of delay variability can be originated at the source, distribution, and receiver sides, and they mainly depend on (i) the features of the involved devices in the distribution chain (sources, receivers, and networking equipment); (ii) the types of media components and their encoding settings; (iii) the network infrastructure through which media is distributed and its current conditions (e.g., available bandwidth, traffic load, number of active receivers, etc.); (iv) the network and Central Processing Unit (CPU) load of the involved devices; and (v) clock imperfections.

Figure 1.5 shows the impact of many relevant systems components and factors on the end-to-end delays and on the delay differences when delivering media. Apart from introducing (undesired and larger than expected) delays, some of these system components may present an unpredictable and uncontrollable behavior, contributing to the increase of the delay variability. Accordingly, a proper understanding of

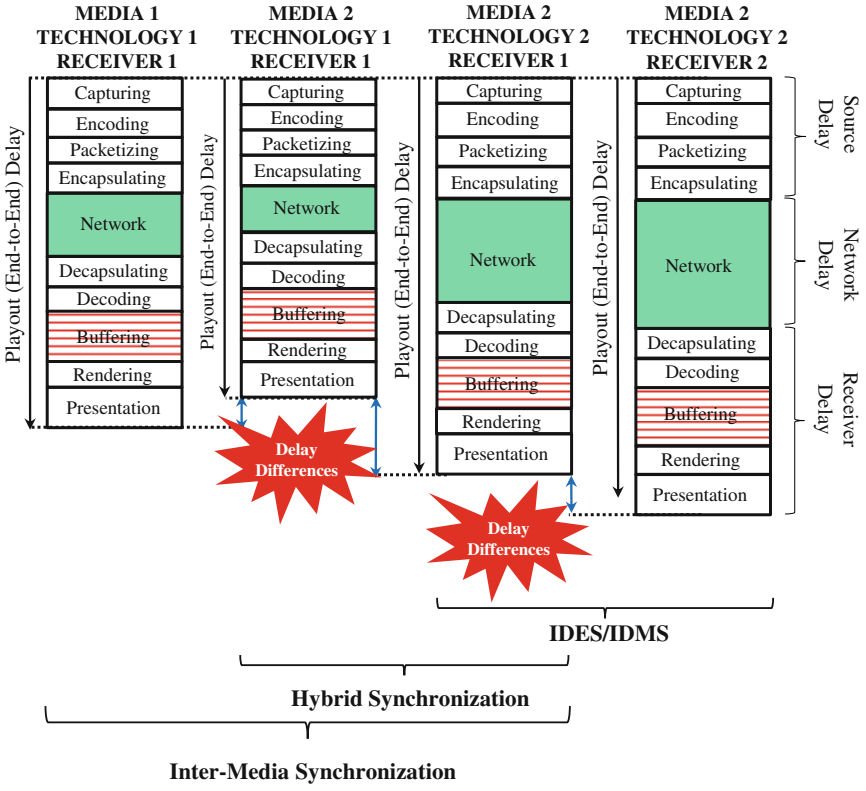


Fig. 1.5 End-to-end delay variability: need for MediaSync

the origin of these sources of delay and of delay differences, as well as their integrated treatment, is essential to efficiently provide mediasync. Next, the most relevant sources of delay and delay differences in each stage of the end-to-end media distribution chain are briefly introduced.

Delays at the source side: The computation heterogeneity of the different involved sources, servers, and senders, as well as the temporal CPU load variation in each one of them, can cause variable delays at the source/sender side. At this stage of the delivery chain, various sources of delay and delay variability can be cited, such as capturing, sampling, encoding, encryption, packetization, protocol layer processing, and transmission buffering. For instance, the capturing of different types of media components can take a different amount of time, as different computational resources are required for each one of them (e.g., audio and 3D video), mainly due to the amount of captured data, to the encoding mechanisms being used, and their settings. For instance, it is shown in [8] that the capturing and encoding delays for audio and video streams can significantly vary, in the order of tens of milliseconds, especially when using different processors and depending on their processing load. Likewise, the use of different delivery technologies can also incur in different processing and transmission overheads. For example, when using DASH, the media contents are commonly encoded in different qualities and stored in web servers for being pulled by clients, which results in an increase of delays.

Delays at the network side: The network delay is the delay experienced by the MUs of a specific media stream sent from a source to reach a specific receiver. It highly varies depending on the followed path, the available resources (e.g., bandwidth, capacity, etc.) and conditions (e.g., traffic load, link failures, number of active receivers, etc.). This is mostly relevant in broadband communications, especially in wireless scenarios. The network delay includes the propagation and serialization delays through the involved links, as well as the processing (e.g., routing decisions, queuing policies, etc.) at the intermediate networking equipment (e.g., routers, switches, multiplexers, etc.). Likewise, advanced procedures, such as fragmentation and reassembly of packets, trans-coding, and format conversion, can also occur at this stage of the distribution chain and have an impact on the delays and on their variability. The variation of inter-arrival times of MUs at the receiver side is commonly referred to as *network jitter*. It also strongly depends on the variable network path, resources, and conditions, as for network delays. For example, as a result of an increase of the network load, the intermediate links and routers can become congested, with a consequent increase of the propagation and processing delays, respectively. Even, data packets may be lost during distribution and may not arrive to the targeted receiver/s. Due to the network jitter, the original temporal relationships between MUs of each individual media component/stream and of different related components/streams will not be kept at the receiver side, so they will need to be reconstructed.

Delays at the receiver side: Delays and delay variability at the receiver side are mainly originated due to buffering, depacketization, protocol layer processing,

decoding, decryption, and rendering processes. In addition, the adoption of any type of error and/or loss concealment techniques will incur in extra and variable delays. Likewise, the total delay at the receiver side can fluctuate over time, according to the instantaneous CPU load of the receiver, the real-time responsiveness of its OS, the protocol layer processing, together with the current network resources and conditions (as it determines the rate at which data packets are received). This variable delay is usually known as *end-system jitter*. At this stage of the distribution chain, two sources of delays should be emphasized: decoding and buffering delays. First, decoding delays become significant when using encoding mechanisms with spatial/temporal interpolation, compensation, and/or prediction techniques, as well as large Group of Pictures (GOP) sizes. Second, elastic playout (also known as reception and de-jitter) buffers are regularly allocated at the receiver side to compensate for the effect of any source of delay variability along the end-to-end distribution chain. Ideally, the delay variability must be compensated for as close as possible to the point of playout. Likewise, it must be also highlighted that the delays and their variability can differ at different receivers, mainly due to their heterogeneous software and hardware resources, to possible different settings and to their variable conditions and states (e.g., other applications being simultaneously executed). These issues must be especially taken into account when performing IDES/IDMS.

An additional factor to take into account when using digital screens is the *display lag*, which is the time difference between the instant at which a video signal is input into a display and the instant at which it is shown on the screen. It may be originated by image processing routines, such as scaling and enhancement. The display lag may also cause a noticeable delay difference (i.e., asynchrony or offset) between the audio and the image signals, thus having an impact on inter-media synchronization. Display lags in High Definition (HD) TVs can vary between 30 and 90 ms, depending on the TV model and on the type of input signal [18]. Besides, it is shown in [19] that the setup of different display modes on the same TV can result in different and variable end-to-end video delays. Accordingly, the features and settings of the specific consumption devices also have a significant impact on the delay and delay variability.

Clock imperfections: The availability of precise, high-resolution, and reliable timing mechanisms is a key aspect in multimedia systems, especially in distributed settings. Media capturing, encoding, transmission, decoding, rendering, and many other processes along the end-to-end distribution chain are executed and scheduled according to the timing provided by end-system and/or external clocks. An incorrect/inaccurate timing can result in incorrect/inaccurate time/delay measurements and in extra delays and delay variability. Therefore, mediasync issues will arise if clock errors or inaccuracies exist, and if the clocks of the involved sources and receivers are not in perfect agreement or do not run at exactly the same rate. Regarding intra-media synchronization, buffer overflow (i.e., flooding) or buffer underflow (i.e., starvation) situations can occur if the receiver's clock is slower or faster than the sender's clock, respectively. Regarding inter-media synchronization,

asynchrony situations will occur if the clock rates of the involved media sources, senders, and receivers do not match. Regarding IDES/IDMS, asynchrony situations will occur if the receivers' clocks are not time-aligned, even if some mechanisms to compensate for the delay differences between them are adopted.

Within this context, the time difference between the clock instances of two entities is referred to as *clock skew*, while the rate of change of the clock skew, due to a non-homogeneous advance of the clock rates (i.e., the clock frequency varies over time), is known as *clock drift*. This fluctuation is very close related to the resolution of the crystal clocks, oscillator stability, voltage changes, aging, surrounding temperature, and other environmental variables (e.g., noise) [20]. The magnitudes of the clock offset and drift are commonly expressed in parts per million (ppm). For example, the existence of a clock skew of 10 ppm with respect to a reference (ideal) clock will cause an error in time/delay measurement, and potentially a subsequent asynchrony, of over 100 ms after a period of 3 h, if it is not corrected. Therefore, it does not suffice with adopting mechanisms to synchronize the involved clocks at the beginning of a multimedia session, but this process must be controlled and repeated throughout its duration, if accurate mediasync needs to be provided. The problem of clock synchronization, including mechanisms to compensate for clock imperfections, can be overcome by efficiently adopting clock synchronization protocols, such as Network Time Protocol (NTP) [21] or Precision Time Protocol (PTP) [22], and other related algorithms. These clock and timing issues are more thoroughly described in Chap. 4 (and are also covered in Chap. 15).

Figure 1.5 shows that end-to-end delay differences can occur when simultaneously delivering different media components via the same technology to the same receiver (e.g., an audio and a video stream sent in individual RTP/RTCP streams), when simultaneously delivering the same media component(s) via different technologies (e.g., the same audiovisual contents in MPEG2-TS format sent in a broadcast DVB stream and in a broadband IPTV or DASH stream), and when simultaneously delivering the same media content(s) via the same technologies to different receivers, regardless of their location. In these conditions, MUs of different media components generated at the same instant will not be simultaneously played out at a specific receiver. Likewise, the same MUs of specific media components will not be simultaneously played out by the involved receivers in a shared media session. All these delay differences, which can even increase if the involved media components are sent by different media sources/senders, clearly reflect the need for different types of mediasync.

1.3.2 Magnitudes of Delays and Delay Differences in Real Scenarios

Previous studies have measured and reported on the magnitudes of delays and delay differences in different real scenarios.

The Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T) G.1050 standard [23] reports on typical values of delays and jitter in Internet. It is indicated that network delays typically range between 20 and 500 ms, while jitter values range between 0 and 500 ms. Likewise, the ITU-T G.114 standard [24] specifies that delays lower than 150 ms are acceptable for Internet conferencing, while delays larger than 400 ms are typically unacceptable. In [25], it is reported that end-to-end delays when using different videoconferencing systems can range between a few tens of ms to more than 300 ms. In particular, it is shown that the delays in a Local Area Network (LAN) scenario accumulate up to 99 ms for Google Talk (Gtalk), and up to 312 ms (with a standard deviation of 67 ms) for Skype. Likewise, end-to-end delays of approximately 350 ms (with a standard deviation of 67 ms) were measured when using an ad-hoc video conferencing system in a distributed scenario between Belgium and United Kingdom (UK).

Likewise, different studies have revealed that delays differences of several seconds occur when delivering media contents using different variants of broadcast and broadband technologies. On the one hand, different TV broadcast technologies make use of different distribution channels and techniques, having a clear influence on the resulting end-to-end delays. In this context, the measurements presented in [18] and [26] reveal that the end-to-end delay differences when delivering the same media content (in the same or different formats) via different TV broadcast technologies and analog cable can accumulate up to 5 s. It is also reported in [16] that satellite communications (DVB-S) involve an extra delay of at least some hundreds of milliseconds compared to the terrestrial variants of DVB. It is mainly due to the delay introduced by the signal propagation process to and from the satellite at the speed of light. Similarly, the study in [27] provides measurements about the magnitudes of delay differences for different TV setups in specific receivers. It is shown that delay differences between different TV broadcasts in a national scenario (in the Netherlands) can accumulate up to almost 5 s, while in an international scenario (between the Netherlands and UK) can accumulate up to 6 s. These measurements also show that analog broadcasted contents are typically delivered faster than digital broadcasted ones (the latter involve extra processes, such as encoding or trans-coding, which contribute to increase the delay) and that, in general, the broadcast delivery of High Definition (HD) contents is slightly slower than the broadcast delivery of Standard Definition (SD) contents. These works have also reported on the magnitudes of delay differences when delivering media contents using broadcast technologies compared to when using broadband technologies. In particular, the measurements in [18] reflect that delays when using web-based (broadband) technologies can be up to 8 s higher when using broadcast technologies, while the ones in [27] indicate that these delay differences can accumulate up to 72 s.

Finally, end-to-end delay differences when delivering media contents to different receivers have also been found in previous studies. In [28], it is stated that these differences can be larger than 6 s in an IPTV scenario. That study additionally analyzes the delay ranges for the different factors along the end-to-end distribution chain that contribute to the increase of delays in such scenarios. The results of such an

Table 1.1 Sources of delay in an IPTV scenario [28]

Delay contributing factor		Delay range (ms)
Source	Video capturing	17–40
	Video encoding	50–2000
	Encryption	0–50
	Error protection	0–100
	Transmission buffer	50–500
Network	Uplink transmission	10–300
	Trans-coding	0–2000
	Downlink transmission	10–300
Receiver	Jitter buffer	50–500
	Error protection	0–100
	Decryption	0–50
	Video decoding	50–500
	Display buffer	0–50
Total		250–6500

analysis are summarized in Table 1.1, which reflects that the end-to-end delays in such scenarios can range between 250 and 6500 ms. The measurements in [27] also revealed that significant delay differences between receivers occur, even when all of them are using exactly the same TV delivery technology, subscription type, setup combination, and equipment. However, no numbers are provided in that study due to the lack of insufficient measurements (e.g., considering different combinations of locations, subscribers, technologies, and setups) and, therefore, of concluding results.

1.3.3 Human Perception of Delays and on Delay Differences

The perceived QoE is a crucial aspect in multimedia systems and applications. In this context, many studies have focused on determining the levels of delay differences that can be tolerable to the human perception. If these levels are exceeded in multimedia systems and applications, the QoE will drop, with the consequent impact in their success.

Since the early development of multimedia communications, jitter has been considered as a key problem to solve, as it can negatively affect the fidelity and intelligibility of media. In fact, studies conducted in the 70 s already analyzed the impact of jitter on intra-media synchronization and on the perceived QoE. For example, it is indicated in [29] that the maximum tolerable jitter for 16-bit high-quality audio is 200 ns. In general, if the value of jitter exceeds the one of the durations of MUs (e.g., audio samples or video frames), they could be received, and thus played out, out-of-order, with a consequent impact on the continuity and intelligibility of the media playout. However, the design of proper buffering strategies can relax these requirements.

As for intra-media synchronization and jitter, many studies have been conducted with the goal of determining the acceptable asynchrony limits for different inter-media synchronization use cases, with a special focus on lip-sync. The subjective studies conducted in [30] for lip-sync indicate that asynchrony levels below 80 ms are unnoticeable, but asynchrony levels above 160 ms become inacceptable. In [1], it is indicated that asynchrony levels between +80 ms (audio ahead video) and −80 ms (audio behind video) are noticeable, but still tolerable, for most users, while asynchrony levels exceeding +160 and −240 ms become unacceptable. In [31], three ranges for the lip-sync asynchrony limits are recommended: undetectability (+25 ms, −95 ms); detectability (+45 ms, −125 ms); and acceptability (+90 ms, −85 ms). More restrictive asynchrony limits are specified in [32], bounding them between +15 and −45 ms.

In [1], the allowable asynchrony limits for other inter-media synchronization use cases, apart from lip-sync, are analyzed and classified, according to the types of media components, their parameters, and the specific application. For example, strict synchronization requirements are specified for audio audio/audio synchronization (±11 μs), while more relaxed synchronization requirements are specified for audio/pointer synchronization (−500 to 750 ms). In [33], the allowable asynchrony limits between (stereo) acoustic signals from a microphone array are analyzed. The results indicate that asynchrony levels of 17 ms are noticeable, but it is preferable to keep them below 11 ms. In [34], it is concluded that asynchrony values lower than 80 ms still guarantee a good perceived visual quality in 3D stereoscopic video, but if they exceed 200 ms, the resulting QoE becomes inacceptable.

Other subjective studies have analyzed the allowable asynchrony limits between audiovisual contents and haptic media. For example, the results in [35] indicate that asynchrony levels around 40–80 ms are hardly noticeable, but they become annoying when reaching 300 ms. Likewise, the allowable asynchrony limits between audiovisual contents and (computer-generated) olfactory data, assuming a perfect lip-sync, have also been analyzed in recent works. For example, it is concluded in [36] that the allowable asynchrony limits in such a use case are less strict than for the other media types, ranging from 30 s when the olfactory data are presented before the audiovisual contents to 20 s when they are presented later. Chapter 12 provides relevant results and findings regarding this use case. In [37], the allowable asynchrony limits between olfactory and haptic data are analyzed. In that work, a virtual scenario in which users have to pick fruits from a tree by using a haptic device is created. When picking the fruit, the users perceive its associated smell and a feedback force from the haptic device. It is concluded that asynchrony levels until 1500 ms are generally tolerable to users in that scenario. In [38], audiovisual contents are enriched with the inclusion of airflow effects from a fan and haptic (concretely, pressure) stimuli from sensors in a gaming vest, and the allowable asynchrony limits between these media types are assessed. It is concluded that the haptic feedback should not be presented before the audiovisual contents, but that delays of up to 1 s are tolerable to users. In contrast, the tolerable asynchrony limits for airflow effects range from −5 s (airflow effects ahead of audiovisual contents) to +3 s (airflow effects behind of audiovisual contents).

The tolerable asynchrony limits in different IDES and IDMS use cases have been also investigated in many studies. Regarding IDES, allowable asynchrony limits for different use cases are recommended in [39]: ±10 μs for tightly coupled audio; 2 ms for real-time audio; 15 ms for advanced audio and 45 ms for lagged audio in audio/video synchronization; and ±80 ms for video animation. Likewise, the tolerable asynchrony limits between audiovisual contents and subtitles in multiscreen scenarios are analyzed in [40]. In the scenario under test, the audiovisual contents from a theater play are presented on a main device, while the associated transcripts (i.e., subtitles) are presented on a companion device. It is concluded that asynchrony levels between −500 and 1000 ms are unlikely to be noticed in this type of multiscreen scenarios.

Regarding IDMS, up to 20 use cases are analyzed and qualitatively ranked according to their synchronization requirements in [15]. The established synchronization levels in that study are very high (10 μs–10 ms); high (10–100 ms); medium (100–500 ms); and low (500–2000 ms). For instance, networked stereo loudspeakers require very high synchronization levels; multiparty conferencing demands high synchronization levels; synchronous e-learning needs medium synchronization levels; and Social TV requires low synchronization levels. Similarly, it is stated in [28] that the allowable asynchrony limits for interactive services requiring IDMS may vary between 15 and 500 ms, depending on the type of service. In some use cases, differences around 100 ms may already cause annoyance to users. In this context, some subjective studies have been conducted to determine the tolerable asynchrony limits in different IDMS scenarios. The study in [27] was focused on the Social TV use case, in which remote users can interact via text and/or voice chat. In these tests, different asynchrony levels, ranging from 0 to 4 s (in steps of 500 ms), were forced and presented to participants in a randomized order, by enabling one of the two interaction channels (voice and text) in each test. The results indicate that asynchrony levels up to 1 s might not be perceptible by users while communicating using audio conferencing services. However, asynchrony levels above 2 s generally become annoying, regardless of the chat modality. When using voice chat, users notice asynchrony situations sooner, and thus get annoyed sooner, but they feel more together than when using text chat. Similar results are provided in [16], in which a shared football watching experience was recreated. More recently, a subjective quality assessment was conducted in [41], with the goal of also determining the tolerable asynchrony levels when geographically distributed users are simultaneously consuming the same media content and interacting. It is argued that asynchrony levels of 400 ms do not have an impact on the QoE, but levels of 750 ms are already noticeable and can degrade the QoE in these scenarios.

From the previous review, it can be first remarked that the tolerable asynchrony limits strongly depend on the mediasync type and on the specific use case. Most importantly, these limits are generally much lower that the magnitudes of delay differences in real network scenarios, as discussed in the previous subsection. This fact clearly reflects the need for designing and adopting adaptive and accurate mediasync solutions to compensate for existing delay differences. Otherwise, the success and mission of the media services/applications being implemented will be

compromised, due to an unsatisfactory QoE (e.g., caused by unintelligibility, incoherent situations and interactions, unfairness, frustration, etc.).

As a summary, a categorization of the magnitudes of delay differences and the tolerable limits for them in different mediasync types and use cases is provided in Table 1.2.

Table 1.2 Magnitudes of and tolerable delay differences in relevant scenarios

Mediasync type	(Approximate) Magnitudes of delay differences	(Approximate) Tolerable delay differences (i.e., asynchrony levels)
Intra-media synchronization	Jitter: 0–500 ms in Internet (ITU-T G.1050) [23]	–Audio streaming: <200 ns [29] –Video streaming: <MU duration (~20–70 ms for typical frame rates)
Inter-media synchronization (including hybrid synchronization)	Up to 6 s when using (different variants of) the same delivery technology [18], [27] >1 min when using broadcast and HTTP-based delivery [27]	–Lip-sync: ~80–100 ms [1, 30–32] –Video/Animation: ~120 ms [1] –Video/Image: ~500 ms [1] –Video/Text: ~500 ms [1] –Audio/Audio: tightly coupled (stereo), ~11us; loosely coupled (dialogue mode), ~500 ms [1] –Array of microphones: <10 ms [33] –Audio/pointer: ~750 ms [1] –3D stereoscopic and multiview video: ~80 ms [34] –Audiovisual contents—haptics: 80 ms (haptic joystick) [35], ~1 s (haptic vest) [38] –Haptics—olfactory data: ~1.5 s [37] –Olfactory data —video: >10 s [36]
IDES/IDMS	Up to 6 s [28]	IDES (local scenarios) –Networked stereo loudspeakers: 10 μs–10 ms [15] –Multiscreen scenarios: video–video: frame level alignment (~20–70 ms for typical frame rates) video subtitles: >1 s [40] IDMS (distributed scenarios) –Multiparty conferencing: 10–100 ms [15] –Networked games: 10–100 ms [15], <750 ms [41] –Social TV: ~1 s [26]

1.4 Essential Aspects and Components of MediaSync Solutions

Given the need for the different types of mediasync, many proprietary and standard solutions have been devised up to date, and remaining research challenges still need to be overcome. Different considerations, implications, and criteria have to be taken into account before proceeding with the design of a mediasync solution. Besides, the existing mediasync solutions are generally comprised of different components, from which different alternatives can be adopted. The design principles/criteria, together with the applicability and suitability of the components of mediasync solutions, will strongly depend on the targeted mediasync type, requirements, scenarios, and applications, as well as on the available resources. Likewise, the high variety of scenarios, media delivery technologies, protocols, contents, and devices have also implied the design of diverse mediasync solutions, as universal solutions are neither efficient nor feasible in this heterogeneous context.

This section identifies key aspects and components of mediasync solutions. The goal is not to provide an exhaustive and complete list, but to help readers in:

- Conceiving the mediasync research problem in a modular and structured manner, as a set of related subproblems. This will contribute to better understand the publications presenting specific mediasync solutions (by identifying the different components they are composed of) or proposing specific components for an improved performance.
- Comprehending why a wide variety of mediasync solutions have been proposed up to date.
- Identifying the essential steps, considerations, and criteria to follow when proceeding with the design of a complete mediasync solution.

1.4.1 Essential Aspects and Design Criteria

The design of a mediasync solution must be preceded by an analysis of essential aspects. Next, **key considerations** to take into account are enumerated:

- The amount and types of the involved media contents, mainly due to their characteristics and requirements (e.g., in terms of processing, bandwidth, latency, etc.).
- The underlying network environment (e.g., broadcast networks, controlled IP scenarios, over-the-top Internet, etc.), delivery technologies/protocols (e.g., RTP, HTTP, etc.), and delivery strategy (e.g., unicast, multicast, broadcast, push-based vs pull-based streaming, etc.) to be used.
- The available resources (e.g., delivery and feedback channels, bandwidth, processing resources, alternative servers, session managers, etc.).

- The involved entities in the end-to-end distribution chain (e.g., media capturing and consumption devices, gateways, trans-coders, multiplexers, etc.).
- The requirements of the targeted application scenarios (e.g., in terms of delays, synchronization accuracy, robustness, scalability, etc.).
- Human perception aspects as well as potential diversities in terms of users' preferences and needs.
- The existence of other types of mediasync solutions for the targeted contents, applications and scenarios. In such a case, the mediasync solution to be designed would ideally have to be able to inter-operate with them, probably sharing resources (e.g., bandwidth, computation, and memory resources). An example is the design of an IDMS solution for a multiparty conferencing service, which already implements an inter-media synchronization (e.g., lip-sync) solution.

The previous factors and considerations, among others, will have a high influence on the mediasync solution to be designed, and on the selection of key **design criteria**, such as

- Mechanisms to specify, convey, and correlate the relationships within media streams, between contents, sources, and/or receivers.
- Most proper locations and entities where mediasync needs to be performed.
- Coordination and prioritization strategies between mediasync types, media contents, sources, and receivers.
- Reusability of existing components and resources or deployment of new ones.

1.4.2 Components of MediaSync Solutions

Based on the considered aspects, the adopted design criteria, and the targeted type of mediasync, the mediasync solutions can be comprised of different components, from which different alternatives can be adopted. Next, **key components** of mediasync solutions are listed:

- Protocols and mechanisms to signalize, describe, associate, and discover the available related media contents, streams, sources, and receivers.
- Protocols for media delivery and for exchanging useful control information (e.g., metadata) to provide mediasync. It also includes the methods and mechanisms to specify and convey the relationships within and between media streams (e.g., use of timestamps, markers, notifications, sequencing, etc.), generally in terms of metadata. This information can be provided independently of or in a multiplexed manner with the delivered media streams. The first approach is used in Synchronized Multimedia Integration Language (SMIL), see Chap. 13, and in Nested Context Language (NCL), see Chap. 14. The second approach is employed when using (some variants of) MPEG technologies, see Chap. 15. In addition, the multiplexed information within the delivered streams can be augmented with extra metadata provided via a feedback channel, as when using RTP/RTCP protocols.

- Involved entities in the end-to-end distribution and synchronization processes. It can include media sources, senders, receivers, gateways, trans-coders, multiplexers, session and synchronization managers, etc. Likewise, different priority levels can be assigned to specific entities (e.g., a reference media source or receiver, backup servers, etc.).
- Protocols and mechanisms for session management and negotiation of key features for mediasync (e.g., encoding settings, media formats, clock references, groups' establishment, priority levels, QoS levels, involved entities, etc.).
- When the mediasync solution relies on the availability of a coherent notion of time in the media session, a key component of it will be the technology or protocol used for clock synchronization. Relevant examples are NTP and PTP protocols, commonly used by media sources and receivers. Chapters 4 and 15 are focused on these types of technologies and protocols. However, mediasync solutions could not rely on these time-related components to achieve the synchronization goal, but instead on other type of information, such as markers, audiovisual features (e.g., by means of watermarking or fingerprinting techniques, as explained in Chap. 22), or even on other features of the media streams, such as their bit rate, as in [42].
- Architectural schemes between the involved entities to achieve mediasync, according to their location and distribution, as well as on the communication processes(es) between them. It includes centralized and distributed schemes, and even hybrid and hierarchical schemes. For example, the applicability and suitability of different schemes for IDMS are analyzed in [15] and in Chap. 2.
- Algorithms to monitor and control the synchronization process (e.g., monitoring and calculation of asynchronies, selection of a temporal reference to synchronize with, dynamic creation of groups, fault tolerance, etc.). Different algorithms can be implemented in different entities involved in the mediasync process.
- Control or adjustment techniques to maintain and/or restore mediasync. In [43] and in [44], the adjustment techniques are classified into four main categories, according to their purpose: basic, preventive, reactive, and common adjustment techniques. *Basic adjustment techniques* are needed in most of the mediasync solutions and are essential to preserve the temporal relationships within or between streams. *Preventive adjustment techniques* attempt to avoid asynchrony situations, before they occur. *Reactive adjustment techniques* are used to recover synchronization after the detection of asynchrony situations. Finally, other *common adjustment techniques* can be used as a means to prevent (preventive) or correct (reactive) asynchrony situations. Likewise, all these categories can be divided into two main groups, depending on if they are executed at the source or at the receiver sides, although they could also be executed at the network side. If the adjustment techniques are performed at the source side, feedback from the receivers will be typically required. Likewise, the use of source-based adjustment techniques requires a coordination with receiver-based techniques, as each individual receiver has to perform the required adjustments to achieve mediasync. Table 1.3 provides a classification and a brief description of relevant adjustment techniques in each category and location. Note that the table is not meant to be exhaustive, but it aims to provide an overview of the most commonly adopted adjustment techniques for mediasync.

Table 1.3 Control or adjustment techniques for MediaSync (adapted from [43] and [44])

Type of technique	Location	Technique
Basic	Server	Addition of useful metadata for synchronization (timestamps, sequence numbers, source and group identifiers, markers, event information, etc.)
	Receiver	Buffering techniques
Preventive	Server	Initial playout instant calculation
		Deadline-based transmission scheduling in stored media (i.e., scheduling the transmission of MU to meet the targeted requirements)
		Interleave MUs of different media streams in a single transport stream
	Receiver	Preventive skips of MUs (eliminations or discardings) and/or preventive pauses of MUs (repetitions, insertions, or stops)
		Adjustment of the buffering time of MUs
Reactive	Server	Adjustment of the transmission timing
		Decrease the number of delivered media streams
		Drop low-priority MUs
	Receiver	Reactive skips (eliminations or discardings) and/or reactive pauses (repetitions, insertions, or stops)
		Playout duration extensions or reductions
		Use of a virtual time axis with contractions or expansions
		Master/Slave switching (e.g., for streams or receivers)
		Late events discarding (i.e., preventing that late media events/contents have an impact on the interactivity of the service)
		Rollback techniques (i.e., to reestablish the state of a shared session to an older one when an inconsistency is detected)
Common	Server	Skip or pause MUs in the transmission process
		Advance the transmission timing in stored media
		Adjustment of the input rate
		Media scaling
	Receiver	Adjustment of the playout rate (i.e., Adaptive media playout or AMP)
		Data interpolation

In general, several adjustment techniques, of different categories, can be employed in mediasync solutions. For example, since preventive adjustment techniques cannot completely avoid asynchrony situations, the combination with reactive adjustment techniques becomes necessary. Similarly, source-based and receiver-based techniques are also typically used in a coordinated manner, as sources and receivers need to cooperate to achieve the synchronization goal.

A typical example is the combination of the addition of useful information for synchronization in the headers of the media data packets (e.g., sequence numbers, timestamps, markers, etc.) at the source side, with buffering and playout adjustment techniques at the receiver side.

Different mediasync solutions can be comprised of different components, and even make use of different variants of these components, mainly according to the targeted application, scenario, requirements, and available resources. Likewise, a proper integration and interaction between these components are necessary for a proper performance of the designed mediasync solutions. In this context, it is also noteworthy to remark that most of these components can be used in a modular manner, thus being adopted, or adapted to be reused, in different mediasync solutions.

Previous studies have assessed the applicability and suitability of some variants of these components for different mediasync types and applications. For example, the study in [15] qualitatively analyzed the applicability and suitability of the most common control schemes for IDMS in terms of key QoS and QoE factors. In [45], the performance of these schemes was compared, and the appropriateness of two reactive playout adjustment techniques (aggressive skips/pauses vs linear Adaptive Media Playout or AMP) was objectively evaluated, in terms of synchronization accuracy and delays. More recently, the appropriateness of aggressive playout adjustment techniques and of three AMP strategies (linear, quadratic, and cubic) was objectively and subjectively assessed, also focusing on IDMS, in [46].

As an example, the main involved components and their interactions in a non-specific IDMS solution (derived from the one presented in [15]) are illustrated in Fig. 1.6. In this solution, the source delivers the media contents to the receivers. The receivers make use of a feedback channel (e.g., via the RTCP protocol) to periodically send reports informing about their playout timing to a centralized synchronization manager.[7] Then, the synchronization manager compares the playout timings included in these reports and, if necessary, makes use of the feedback channel to send a new control message to the receivers, including the required information to achieve synchronization. Therefore, the involved entities are structured in a centralized architectural or control scheme, which is commonly known as Synchronization Maestro Scheme (SMS), and is explained with more details in Chap. 2. The synchronization manager can be an independent entity, but it can also be co-located with the media source or with any of the receivers. It will typically implement some monitoring and control algorithms (e.g., to calculate the asynchrony, to detect and react to faults, etc.) for the synchronization process. Likewise, the same entity or an additional one can be used to act as a session manager (e.g., to dynamically create and maintain different groups of receivers in a shared session). In addition, both the media sources and receivers can implement

[7] Also known as Media Synchronization Application Server (MSAS), maestro, and synchronizer in literature.

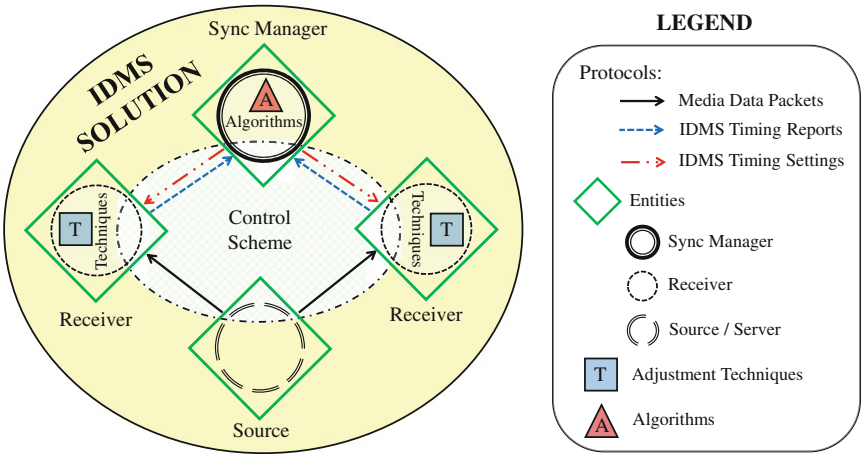


Fig. 1.6 Components of an IDMS solution and their relationships

different control or adjustment techniques to maintain and/or restore mediasync. The selection of these techniques can be independent of the adopted synchronization protocol and architectural scheme. Finally, in case that clock synchronization is required for mediasync, the involved entities will maintain periodic communication with a (possibly external) clock source server (e.g., an NTP server).

It is important to remark that the previous example and figure are not meant to include all the components required for every IDMS solution, but to provide a general idea about the components of a specific one, and about their possible interactions. Other IDMS solutions, and especially solutions for other mediasync types, can be comprised of other components or make use of different variants of the same components.

1.5 Summary

Mediasync is a fundamental aspect for the successful deployment of multimedia systems. Due to its high relevance, many research efforts have been, and are being, devoted to overcoming existing challenges. This chapter has introduced the broad mediasync research area, by providing key definitions, classifications, and examples. The chapter has also discussed the relevance of the different types of mediasync, and reflected their necessity, by comparing the magnitudes of delay differences in real scenarios to the asynchrony limits that can be tolerated by users. Finally, the chapter has introduced key considerations and criteria to take into account when designing a mediasync solution. The most common components of mediasync solutions have been also identified and briefly described, and some

examples have been provided. Overall, the chapter can help readers in acquiring a deeper understanding of the mediasync research area and comprehending the subsequent chapters of the book.

References

1. Steinmetz, R.: Human perception of jitter and media synchronization. IEEE J. Sel. Areas Commun. **14**(1), 61–72 (1996)
2. Chen, M.: A low-latency lip-synchronized videoconferencing system. In: SIGCHI Conference on Human Factors in Computing Systems, ACM CHI'03, pp. 464–471. New York, April 2003
3. Bartoli, I., Iacovoni, G., Ubaldi, F.: A synchronization control scheme for videoconferencing services. J. Multimedia **2**(4), 1–9 (2007)
4. Rodriguez, A., Talavera, G., Orero, P., Carrabina, J.: Subtitle synchronization across multiple screens and devices. Sensors **12**(7), 8710–8731 (2012)
5. Huang, P., Ishibashi, Y.: QoE assessment of will transmission using vision and haptics in networked virtual environment. Int. J. Commun. Netw. Syst. Sci. (IJCNS) **7**(8), 265–278 (2014)
6. Ademoye, O.A., Ghinea, G.: Synchronization of olfaction-enhanced multimedia. IEEE Trans. Multimedia **11**(3), 561–565 (2009)
7. Murray, N., Qiao, Y., Lee, B., Muntean, G.-M.: User-profile-based perceived olfactory and visual media synchronization. ACM Trans. Multimedia Comput. Commun. Appl. (TOMCCAP) **10**(1s), 11, 24, Jan 2014
8. Huang, Z., Nahrstedt, K., Steinmetz, R.: Evolution of temporal multimedia synchronization principles: a historical viewpoint. ACM Trans. Multimedia Comput. Commun. Appl. (TOMCCAP) **9**(1s), 34, 23, Oct 2013
9. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: a transport protocol for real-time applications. IETF Internet Standard, RFC 3550, July 2003
10. Concolato, C., Thomas, S., Bouqueau, R., Le Feuvre, J.: Synchronized delivery of multimedia content over uncoordinated broadcast broadband networks. In: ACM MMSys'12. Chapel Hill, North Carolina, USA, Feb 2012
11. Veenhuizen, A.: Frame accurate media synchronization of heterogeneous media sources in an HBB context. In: MediaSync Workshop 2012. Berlin (Germany), Oct 2012
12. Le Feuvre, J., Viet-Than-Trung, N., Hammidouche, W., Marchal, P., Monnier, R., Dupain, P.: A test bed for hybrid broadcast broadband services. In: MediaSync Workshop 2015. Brussels (Belgium), 3 June 2015
13. Boronat, F., Montagud, M., Marfil, D., Luzón, C.: Hybrid broadcast/broadband TV services and media synchronization. Demands, preferences and expectations of Spanish consumers. IEEE Trans. Broadcast. **PP**(99), 1–18, Aug 2017
14. van Brandenburg, R., Veenhuizen, A.: Immersive second-screen experiences using hybrid media synchronization. In: MediaSync Workshop 2013. Nantes (France), Oct 2013
15. Montagud, M., Boronat, F., Stokking, H., van Brandenburg, R.: Inter-destination multimedia synchronization: schemes, use cases and standardization. Multimedia Syst. **18**(6), 459–482 (2012)
16. Mekuria, R., Cesar, P., Bulterman, D.: Digital TV: the effect of delay when watching football. 10th European Conference on Interactive TV and Video (EuroiTV '12). Berlin (Germany), July 2012
17. Hybrid Broadcast Broadband TV (HbbTV) 2.0.1 Specification, HbbTV Association Resource Library, July 2016. https://www.hbbtv.org/resource-library

18. Mekuria, R.N.: Inter-destination media synchronization for TV broadcasts. In: Master Thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, Department of Network architecture and Services. Delft University of Technology, Apr 2011
19. Jansen, J., Bulterman, D.C.A.: User-centric video delay measurements. In: 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '13), pp. 37–42. Oslo (Norway), Feb 2013
20. Ridoux, J., Veitch, R.: Principles of robust timing over the internet. Com. ACM **53**(5), May 2010
21. Mills, D., Delaware, U., Martin, J., Burbank, J., Kasch, W.: Network time protocol version 4: protocol and algorithms specification. In: IETF Internet Standard, RFC 5905, June 2010
22. IEEE 1588-2008: IEEE standard for a precision clock synchronization protocol for networked measurement and control systems (2008)
23. ITU-T Rec. G.1050: Network model for evaluating multimedia transmission performance over internet protocol (2007)
24. ITU-T Rec. G. 114: One-way transmission time (2003)
25. Jansen, J., Cesar, P., Bulterman, D.C.A., Stevens, T., Kegel, I., Issing, J.: Enabling composition-based video-conferencing for the home. IEEE Trans. Multimedia (TMM) **13**(5), 869–881 (2011)
26. Geerts, D., Vaishnavi, I., Mekuria, R., Van Deventer, M.O., Cesar, P.: Are we in sync?: synchronization requirements for watching on-line video together. ACM CHI '11, New York, USA (2011)
27. Kooij, W., Stokking, H., van Brandenburg, R., de Boer, P.-T.: Playout delay of TV signals: measurement system design, validation and results. In: ACM TVX 2014, Newcastle (UK), June 2014
28. Van Deventer, M.O., Stokking, H., Niamut, O., Walraven, A., Klos, V.B.: Advanced interactive television service require synchronization. In: 15th International Conference on Systems, Signals and Image Processing, IWSSIP 2008. Bratislava (Slovak Republic), June 2008
29. Blesser, B.: Digitization of audio: a comprehensive examination of theory, implementation, and current practice. AES J. Audio Eng. Soc. **26**(10), 739–771 (1978)
30. Steinmetz, R., Engler, C.: Human perception of media synchronization. In: Technical Report 43.9310. IBM European Networking Center, Heidelberg (1993)
31. ITU-R BT.1359: International telecommunication union radio communication sector relative timing of sound and vision for broadcasting
32. ATSC Implementation Subcommittee Finding: Relative Timing of Sound and Vision for Broadcast Operations, Doc. ID-191, June 2003
33. Dannenberg, R., Stern, R.: Experiments concerning the allowable skew of two audio channels operating in the stereo mode. Personal Commun. (1993)
34. Goldmann, L., Jong-Seok Lee, L., Ebrahimi, T.: Temporal synchronization in stereoscopic video: Influence on quality of experience and automatic asynchrony detection. IEEE ICIP 2010, Hong Kong, Sept 2010
35. Fujimoto, T., Ishibashi, Y., Sugawara, S.: Influences of inter-stream synchronization error on collaborative work in haptic and visual environments. In: Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Mar 2008
36. Ghinea, G., Ademoye, O.A.: Perceived synchronization of olfactory multimedia. IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans **40**(4), 657–663 (2010)
37. Hoshino, S., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE assessment in olfactory and haptic media transmission: influence of inter-stream synchronization error. In: IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR) Workshop, May 2011
38. Yuan, Z., Bi, T., Muntean, G.-M., Ghinea, G.: Perceived synchronization of mulsemedia services. IEEE Trans. Multimedia **17**(7), 957–966 (2015)

39. Bang, Y., Han, J., Lee, K., Yoon, J., Joung, J., Yang, S., Rhee, J.: Wireless network synchronization for multichannel multimedia services. IEEE ICAT 2009, Sarajevo (Bosnia and Herzegovina), Feb 2009
40. Ziegler, C., Keimel, C., Ramdhany, R., Vinayagamoorthy, V.: On Time or Not on Time: A user study on delays in a synchronised companion-screen experience. In: ACM International Conference on Interactive Experiences for TV and Online Video (TVX '17). Hilversum (Netherlands), June 2017
41. Rainer, B., Petscharnig, S., Timmerer, C., Hellwagner, H.: Is one second enough? evaluating QoE for inter-destination multimedia synchronization using human computation and crowdsourcing. In: IEEE Seventh International Workshop in Quality of Multimedia Experience (QoMEX), pp. 1–6. Costa Navarino, Messinia, Greece, May 2015
42. Pereira, I., Silveira, L.F., Gonçalves, L.: Video synchronization with bit-rate signals and correntropy function. Sensors **17**(9), 2021 (2017)
43. Ishibashi, Y., Tasaka, S.: A comparative survey of synchronization algorithms for continuous media in network environments. In: 25th Annual IEEE Conference on Local Computer Networks, LCN 2000, pp. 337–348. Florida (USA), Nov 2000
44. Boronat, F., Lloret, J., García, M.: Multimedia group and inter-stream synchronization techniques: a comparative study. Inf. Syst. **34**(1), 108–131 (2009)
45. Montagud, M., Boronat, F., Stokking, H., Cesar, P.: Design, development and assessment of control schemes for IDMS in a standardized RTCP-based solution. Comput. Netw. **70**(9), 240–259 (2014)
46. Montagud, M., Boronat, F., Roig, B., Sapena, A.: How to perform AMP? cubic adjustments for improving the QoE. Comput. Commun. **103**, 61–73 (2017)

Chapter 2
Evolution of Temporal Multimedia Synchronization Principles

Zixia Huang, Klara Nahrstedt and Ralf Steinmetz

Abstract Ever since the invention of the world's first telephone in the nineteenth century, the evolution of multimedia applications has drastically changed human life and behaviors, and has introduced new demands for multimedia synchronization. In this chapter, we present a historical view of temporal synchronization efforts with a focus on continuous multimedia (i.e., sequences of time-correlated multimedia data). We demonstrate how the development of multimedia systems has advanced the research on synchronization, and what additional challenges have been imposed by next-generation multimedia technologies. We conclude with a new application-dependent multilocation multi-demand synchronization framework to address these new challenges.

Keywords Continuous multimedia · Temporal synchronization · Evolution

2.1 Introduction

The past century has witnessed generations of multimedia applications. The maturity of storage and transmission technologies enables transition from analog modulation to digital media. The emergence of low-cost sensory devices contributes to growing popularity of multimodal multichannel data. The advancement of high-speed wireline and wireless Internet allows transmission and sharing of these multimedia information at a large scale.

Z. Huang (✉)
Google Inc., Mountain View, USA
e-mail: zixia@google.com

K. Nahrstedt
University of Illinois at Urbana-Champaign, Champaign, USA
e-mail: klara@illinois.edu

R. Steinmetz
Technische Universitat Darmstadt, Darmstadt, Germany
e-mail: ralf.steinmetz@kom.tu-darmstadt.de

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_2

Multimedia synchronization is needed to preserve original correlations among diverse multimedia data, so that they are **synchronous** (or **in-sync**) before their final presentation. There are two major synchronization categories in multimedia systems. On the one hand, **temporal synchronization** [13, 55] requires presentation of multimedia data based on their original time attributes. For example, a motion picture and an audio sample which are captured by a camera and microphone at the same time must be presented at the corresponding output devices synchronously. On the other hand, **spatial synchronization** [51] demands layout alignment of media data based on their contextual correlations at each time point. For instance, two images must show up on the left and right side of a presentation slide in the first two seconds. Existing synchronization studies mostly focus on temporal synchronization because of the time-sensitive nature of multimedia applications, which demand strict time correlations among multimedia data.

Temporal synchronization is demanded for both **continuous** and **discrete** multimedia data [13, 50]. Continuous multimedia are defined as sequences of time-correlated media packets, which are generated by one or multiple sensory devices over time. Video, audio, and haptic data are all continuous multimedia. On the contrary, discrete multimedia are the set of static media data like single images and texts, or standalone media **events** (e.g., pop-up of an image, or movement of a text). Synchronization of discrete multimedia may come with a coarse granularity where only their temporal order needs to be preserved. Hence, it is also called **event synchronization**. There have been numerous synchronization research works for both continuous and discrete multimedia. This chapter only focuses on continuous multimedia.

The configuration of a continuous multimedia system can be represented in multiple forms of media components (Sect. 2.2), where each component demands individual temporal synchronization. However, their original time dependencies at the source sensory devices can lose track in multiple locations during media computation and distribution, because of variations of Internet latencies and computation demands. The problem is called **mis-synchronization**. A mis-synchronization in one location of a multimedia system can be propagated to future locations, and multimedia data become **asynchronous** (or **out-of-sync**) during their final presentation.

Mis-synchronization of multimedia data can negatively impact human perception. Depending on application functionalities, a single multimedia system may exhibit heterogeneous demands in terms of synchronization and affects user experience at different levels.

Multimedia synchronization has already been a well-known issue in traditional multimedia applications (e.g., 2D video conferencing, on-demand video, video broadcast, etc.). **Next-generation multimedia systems** (NG-MS), like 3D tele-immersion, virtual reality, and Internet of Things (IoT), utilize multimodal multi-channel sensors to provide users an immersive and realistic experience. They are becoming more complex in terms of hardware configurations, more diverse in terms of application functionalities, and more expensive in terms of consumptions of computation and network resources. Hence, preserving the time correlations of media data in these systems becomes an even larger challenge. A systematic framework

is needed to integrate application-dependent multilocation multi-demand synchronization problems, in order to achieve in-sync multimedia presentation at their final outputs. We will show that such a framework is unfortunately missing in existing studies.

In this chapter, we present a historical view of the temporal synchronization studies for continuous multimedia over the past 30 years. Based on synchronization definitions and formulations (Sect. 2.2), we demonstrate how the development of multimedia technologies has advanced the research on multimedia synchronization, and what additional issues have been imposed by NG-MS (Sect. 2.3). We conclude with a multidimensional synchronization framework to address these issues at the end of the chapter (Sect. 2.4).

2.2 Synchronization Formulation

Before we discuss existing research studies on multimedia synchronization, we formulate the term "synchronization". We present a mathematical model which will be used throughout this chapter. The mathematical symbols and their denotations are also listed in Table 2.1 in Appendix I.

2.2.1 Continuous Multimedia Data Model

The overall architecture of continuous multimedia data can be described in five categories in a hierarchical fashion.

- **Session**. A session indicates a status of multimedia communications between two or more *sites* (end systems). In this chapter, we use $\{n^1, \ldots, n^N\}$ to denote N sites within the same session.
- **Bundle**. A bundle is a set of time-correlated multimedia data outputted from heterogeneous sensors of the same sender site. We denote the bundle of site n^x as u^x.
- **Media Modality**. To provide users with a realistic and immersive experience, each site may be equipped with multiple multimedia sensory devices with different modalities: 2D/3D videos, audios, haptics, etc. We let m_i^x be the i-th media modality of site n^x, i.e., $\{m_1^x, m_2^x, \ldots\}$. For example, we can use $i = 1$ or "V" to represent the video modality, $i = 2$ or "A" for the audio modality, $i = 3$ or "H" for the haptic modality, etc.
- **Sensory Stream**. To preserve directionality and spatiality of the physical room environment, multiple media sensors of the same modality (e.g., a microphone array or a multi-camera array) can capture a scene at the same time, but from different angles. Each sensor produces a sensory stream $s_{i,j}^x$ (j is the stream index).

- **Media Frame**. A sensory stream is composed of a sequence of media frames (i.e., motion images and audio samples), captured by the same sensor over time. We denote the k-th media frame of $s^x_{i,j}$ as $f^x_{i,j}(k)$.

Hence, a site n^x outputs a multimedia bundle data that can include multiple media modalities, i.e., $u^x = \{m^x_1, m^x_2, \dots\}$. Each media modality can produce multiple media streams: $m^x_i = \{s^x_{i,1}, s^x_{i,2}, \dots\}$. For example, $m^x_{\mathrm{A}} = \{s^x_{\mathrm{A},1}, s^x_{\mathrm{A},2}, s^x_{\mathrm{A},3}\}$ represents the audio modality at site n^x with three audio streams captured by a microphone array. Each sensory stream is further composed of a sequence of media frames: $s^x_{i,j} = \{f^x_{i,j}(1), f^x_{i,j}(2), \dots\}$. For example, $s^x_{\mathrm{V},2} = \{f^x_{\mathrm{V},2}(1), f^x_{\mathrm{V},2}(2), \dots, f^x_{\mathrm{V},2}(k)\}$ represents the second video stream at site n^x with k media frames.

2.2.2 *Layers of Synchronization Demands*

Due to the hierarchical multisite multisensory nature of multimedia data, four layers of synchronization relations are demanded, where each *synchronization layer* from bottom-up is depicted in Fig. 2.1.

Intra-stream synchronization prescribes synchronous presentation of media frames within each sensory stream at receivers, according to their original captured timeline at the multimedia sensors. A mis-synchronization in this layer can cause temporal media distortion (e.g., image jerkiness or audio pitch).

Intra-media synchronization refers to synchronization of sensory streams from multiple media devices of the same media modality within a media bundle. Mis-synchronization in this layer can violate their spatial and temporal correlations during media presentation (e.g., a visual mismatch between two multi-view images).

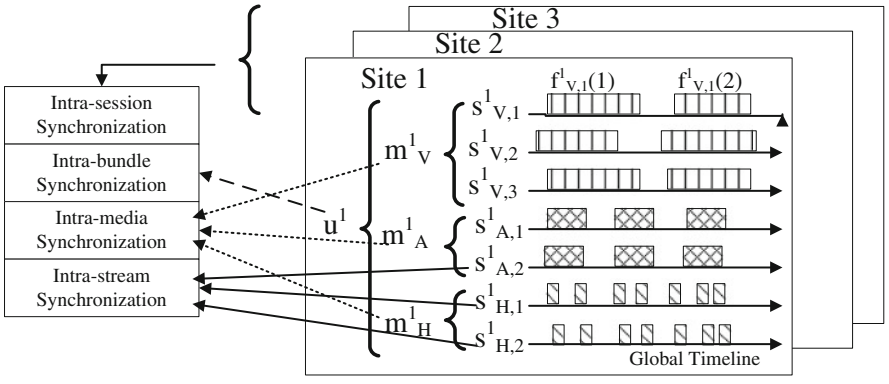


Fig. 2.1 Four layers of synchronization relations. $f^x_{i,j}(k)$ denotes the frame k in stream $s^x_{i,j}$; $s^x_{i,j}$ denotes the j-th sensory stream in media modality i="V", "A", and "H"; and m^x_i denotes the media modality in bundle u^x at site n^x ($x = 1, 2, 3$)

Intra-bundle synchronization is defined as synchronization of multiple media modalities within a bundle. This layer evaluates timing consistency across different media modalities. The most studied example would be audiovisual synchronization (e.g., lip synchronization). Note that previous studies (e.g., [13, 51, 55]) usually combine intra-media and intra-bundle synchronization demands into a single layer called **inter-stream synchronization** or **inter-media synchronization**, i.e., synchronization of multiple multimodal streams within a media bundle. In NG-MS, the scalability of media devices of each media modality and the diversity of new media devices of heterogeneous media modalities are increasing, posing very different requirements on intra-media and intra-bundle synchronization demands. Therefore, these two synchronization relations should be addressed separately.

Intra-session synchronization represents ***inter-receiver*** and/or ***inter-sender*** synchronization within a multimedia session. The inter-receiver synchronization, also named **group synchronization** or **inter-destination synchronization**, has been extensively studied by the community (e.g., [13, 17]). It refers to synchronization of media bundles from the same sender site (or media server) to multiple receivers (e.g., synchronous video playback during TV broadcast). An out-of-sync presentation can cause inconsistent interactions when multiple users at different receiver sites get a timing privilege to conduct an activity. The inter-sender synchronization, also named **inter-source synchronization**, is a new demand imposed by interactive and immersive activities. It represents in-sync presentation of media bundles from multiple senders at the same receiver (e.g., synchronous playout of 3D video streams of multiple scenes in a joint virtual space). A mis-synchronization may lead to confusion of the users when they are watching senders conducting a highly collaborative activity.

2.2.3 Definition of Synchronization Skews

The **synchronization skew** in a continuous multimedia setting is defined as the delay difference of two time-correlated *media objects* (media frame, sensory stream, media modality, and participating site), traveling from the media sources to the current location. One of the objects is usually the **synchronization reference**, i.e., the (most important) media object that other objects need to be synchronized against. Because of the multilayer synchronization hierarchy, a media object can be represented in multiple forms, meaning that the synchronization references must change accordingly at different layers. Thus, it is not possible to use a single skew to describe the whole multimedia session, but rather, multiple skew definitions for different layers will be more reasonable. Please note that a synchronization reference at each layer can be dynamically changed throughout a media session because of possible activity changes in a multimedia application.

Intra-stream synchronization skew. The skew within a sensory stream $s^x_{i,j}$ is evaluated by computing the delay difference of a media frame $f^x_{i,j}(k)$ w.r.t. the *reference frame* $f^x_{i,j}(*)$. We denote $D(f^x_{i,j}(k), n^y)$ as the experienced latency of $f^x_{i,j}(k)$ from its capturing time, when it is being delivered to the receiver site n^y. Thus, the intra-stream synchronization skew is defined by Eq. 2.1 as

$$\forall x, y, i, j,\ f^x_{i,j}(k) \in s^x_{i,j} : \quad \Delta D(f^x_{i,j}(k), n^y) = D(f^x_{i,j}(k), n^y) - D(f^x_{i,j}(*), n^y) \tag{2.1}$$

Intra-media synchronization skew. We denote $D(s^x_{i,j}, n^y)$ as the experienced latency of $s^x_{i,j}$ when delivered to n^y. Note that due to potential computation and Internet jitter across media frames within the sensory stream, we use the latency of the reference frame to represent that of the stream, i.e., $D(s^x_{i,j}, n^y) = D(f^x_{i,j}(*), n^y)$. Hence, the intra-media synchronization skew $\Delta D(s^x_{i,j}, n^y)$ w.r.t. the *reference stream* $s^x_{i,*}$ is defined by Eq. 2.2 as

$$\forall x, y, i, j,\ s^x_{i,j} \in m^x_i : \quad \Delta D(s^x_{i,j}, n^y) = D(s^x_{i,j}, n^y) - D(s^x_{i,*}, n^y) \tag{2.2}$$

Intra-bundle synchronization skew. Because sensory streams within a media modality can experience heterogeneous latencies, we prescribe that the latency of a media modality is equivalent to that of the intra-media synchronization reference (i.e., reference stream) within this modality, in order to best match human perceptual interests, i.e., $D(m^x_i, n^y) = D(s^x_{i,*}, n^y)$. Thus, the intra-bundle synchronization skew of m^x_i w.r.t. the *reference modality* m^x_* is defined by Eq. 2.3 as

$$\forall x, y, i,\ m^x_i \in u^x : \quad \Delta D(m^x_i, n^y) = D(m^x_i, n^y) - D(m^x_*, n^y) \tag{2.3}$$

Note that the **inter-stream synchronization skew** studied in multiple studies (e.g., [13, 51, 55]) is defined regardless of media modalities. In other words, it uses a single reference stream (denoted as s^x_*) for all other streams of different media modalities within the same bundle. The skew in these studies can be defined by Eq. 2.4 as

$$\forall x, y, i, j : \quad \Delta D(s^x_{i,j}, n^y) = D(s^x_{i,j}, n^y) - D(s^x_*, n^y) \tag{2.4}$$

There is no skew constraint between two non-reference streams in inter-stream synchronization. For example, we are unable to bound the skew between two video streams (from a multi-camera system) which use the same audio stream as the reference. This is why we propose the intra-media and intra-bundle synchronization layers separately. The issue has been neglected even in the work finished when camera/microphone arrays were being deployed [16], mainly because of the community's stereotyped view of synchronizing a single video and a single audio stream in the most common on-demand or conferencing multimedia systems.

Intra-session synchronization skew. Similar to the intra-bundle layer, we prescribe that the latency of a bundle is equivalent to that of the intra-bundle synchronization reference within the bundle, i.e., $D(u^x, n^y) = D(m^x_*, n^y)$. Given the *reference*

site n^*, the inter-sender synchronization skew as to a receiver site n^{y_0} is defined by Eq. 2.5 as

$$\forall x : \quad \Delta D(u^x, n^{y_0}) = D(u^x, n^{y_0}) - D(u^*, n^{y_0}) \tag{2.5}$$

Accordingly, the inter-receiver synchronization (group synchronization) skew as to a sender site n^{x_0} is defined by Eq. 2.6 as

$$\forall y : \quad \Delta D(u^{x_0}, n^y) = D(u^{x_0}, n^y) - D(u^{x_0}, n^*) \tag{2.6}$$

In continuous multimedia, the synchronization skews are usually evaluated at specific time points, called ***synchronization points***. Multiple studies utilize the concept of synchronization points to evaluate synchronization skews and perform synchronization controls [76].

Based on the above formulation, we will review existing research works on multimedia synchronization. Each work addresses one or multiple layers of synchronization demands. For consistency, we will use the same set of mathematical symbols throughout the chapter.

2.3 A Historical View of Multimedia Synchronization Studies

The multimedia technologies have experienced multiple generations of evolution, with different synchronization focuses in each generation. In this chapter, we roughly divide them into four stages. In each stage, we discuss the advancement of multimedia technologies and its impact on synchronization. Figure 2.2 shows a timeline of the four stages.

2.3.1 Years of Birth: In and Before 1980s

The rise of electronic technologies had given birth to a number of analog and digital multimedia applications in early years. The rapid deployment of digital computing and communication technologies, such as PCs and Internet, and their unreliable characteristics brought people's attention to the problem of digital multimedia synchronization. However, the synchronization concept was mainly concerned with the fidelity or intelligibility of multimedia signals.

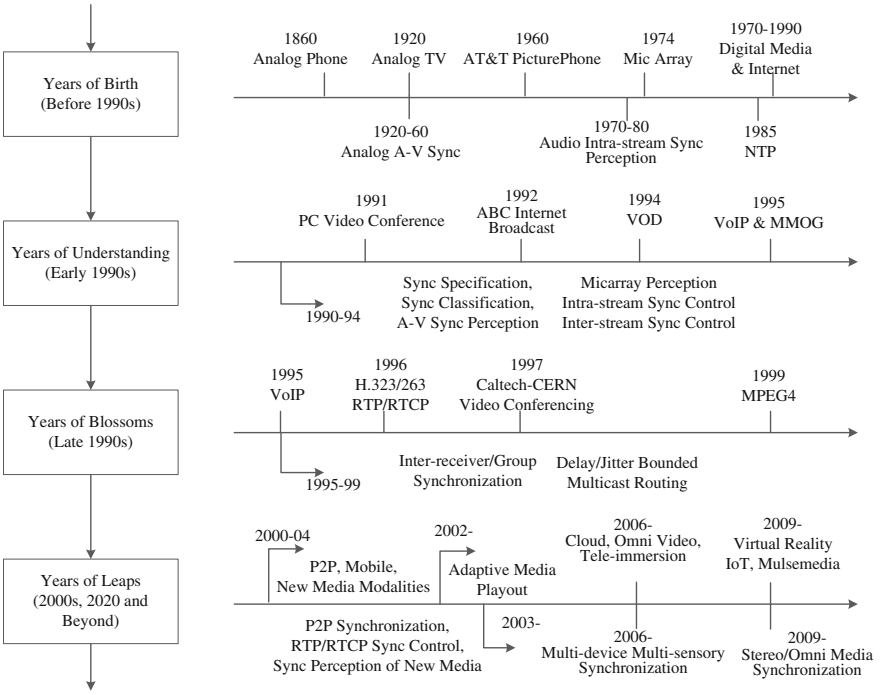


Fig. 2.2 Advancement timeline of multimedia and synchronization technologies

2.3.1.1 Historical Background

Back to the years of 1860s and 1870s, the telephone device was invented to allow the analog speech transmissions over wired circuits [11], thus starting a new era of multimedia innovations. In the late 1920s, the idea of television set was proven practical, and the broadcast analog TV service rapidly developed ever since. Later in 1960s, AT&T Bell Labs demonstrated its own analog picturephone which supported a video frame rate of up to 30 fps [1]. In 1974, the microphone array (or microphone antenna) technique was invented by Billingsley [56].

Analog multimedia synchronization between audio and video had been an issue, but it was solved early on by approaches such as taking analog audio and video signals, multiplexing them and transmitting them over a controlled communication channel [1]. In addition, the quality of analog audio and video signals is not reliable; hence, it became the priority problem to solve. The concept of analog multimedia applications (radio and TV) was being accepted by people, who demonstrated more of a curiosity than an everyday demand.

2.3.1.2 Start of Synchronization Perception Studies

It was not until the 1970s and 1980s that the digital multimedia synchronization was realized as a problem. The invention of the computing machines fostered the development of digital media, while the introduction of the best-effort Internet protocols brought people's attention to the concept of delay variations (i.e., *jitter*). People became interested in how the Internet jitter affected digital media fidelity and human perception, and multiple preliminary studies were conducted to discuss the impact of jitter on intra-media synchronization of digital audios. For example, Blesser [14] offers several experimental results demonstrating that the maximum tolerable jitter for 16-bit high-quality audio is 200 ns in one sampling period. Similar work was also done in [79], which recommends a maximum allowable jitter of no more than 10 ns.

2.3.1.3 NTP: A Clock Synchronization Protocol

In 1985, David Mills proposed the first version of Network Time Protocol (NTP) in RFC 958 [2], a protocol designed for synchronizing the clocks of distributed computers connected by the Internet. NTP has gone through four iterations so far, and the latest version is published in RFC 5905 [7].

To synchronize one computing machine (called *slave*) against other (called *master*), the NTP slave computes the round-trip delays by sending a set of User Datagram Protocol (UDP) packets to the remote master. We assume that a packet leaves the slave at t_1 and arrives at the remote master at t_2 (Fig. 2.3a). We also denote that the packet leaves the master at t_3 and returns to the slave at t_4. All times are measured based on the local clocks. Hence, the clock offset between machines is defined by Eq. 2.7 as

$$\delta = \frac{(t_2 - t_1) + (t_3 - t_4)}{2} \tag{2.7}$$

Equation 2.7 implies that the synchronization approach assumes symmetrical round-trip delay. But in reality, the unequal bidirectional latency and jitter can degrade the clock synchronization accuracy. In addition, time measurement is at

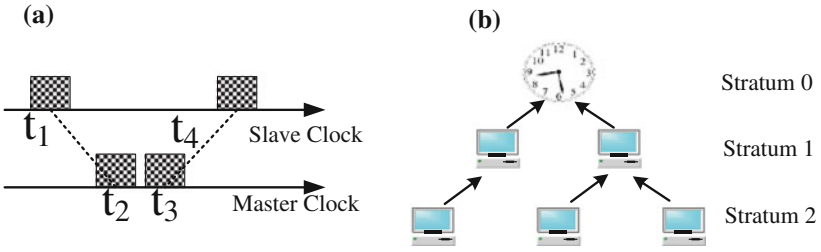


Fig. 2.3 **a** NTP clock offset computation, **b** NTP multi-stratum hierarchy

the application layer, whose accuracy depends on the underlying operating system. In general, NTP can only lead to a synchronization accuracy up to the range of 10 ms [36]. To minimize the impact of jitter and address the issue of computing machine scalability, NTP adopts a multi-stratum hierarchy (Fig. 2.3b), where machines in a *stratum layer l* are synchronized to the corresponding masters in the higher stratum layer $l-1$ based on Eq. 2.7. A stratum layer in NTP represents the synchronization distance from the reference clock source.

NTP is important in multimedia applications, because it provides a solution to have a coherent notion of time (by accessing the same or a related global clock) across distributed machines. It allows us to identify the temporal correlation between two media objects, which are produced or are operating at different physical systems. We will show that existing studies heavily rely on this global timing state in order to achieve multimedia synchronization throughout the chapter.

2.3.2 Years of Understanding: Early 1990s

Owing to the technological advances of the Internet protocols, many Internet-based digital multimedia systems emerged and were commercialized in late 1980s and early 1990s. Multimedia synchronization became a known and important topic to the research community, and extensive amounts of research were done to understand the synchronization problem. These studies covered a broad synchronization area, including classification and specification modeling, subjective perception evaluation, and synchronization control algorithms.

2.3.2.1 Historical Background

In 1991, IBM and PictureTel introduced the first PC-based black-and-white video conferencing system [63]. In 1992, the teleorchestration service was invented as a stream-oriented interface for continuous media presentation across multiple distributed systems [18], while a real-time virtual multichannel acoustic environment was invented by Gardner based on microphone arrays [27]. The first commercial television program, ABC World News, was broadcasted over the Internet in the same year [3]. The Video On Demand (VOD) service was also started under the Cambridge project, offering video streaming at a bandwidth up to 25 Mbps [4].

The proliferation of new Internet-based multimedia systems and the improvement of digital audio-visual fidelity promoted researchers to address the synchronization problems. The Internet delay variations between the (single) audio and (single) video streams in both live and on-demand video systems exhibited a need for intra-bundle synchronization. The delay variations between multiple audio streams in the microphone array setup showed that intra-media synchronization was also important. The development of the teleorchestration service brought people's attention to inter-

receiver/group synchronization. Multimedia synchronization studies thus became a hot topic during this period.

2.3.2.2 Synchronization Classification

To understand the heterogeneous demands of multimedia synchronization, a classification model was needed to identify the structure of synchronization mechanisms. Many models were proposed, with views from different aspects of the synchronization problem [61, 78].

Little et al. Model [51]. Proposed by Little, Thomas, and Ghafoor in 1991, the classification model spans over both spatial and temporal synchronization. The temporal synchronization includes intra-stream synchronization and inter-stream synchronization (i.e., stream synchronization regardless of media modalities), in spite of random network delays. Discrete timed media objects, like still images and texts, are also included in this category. However, neither spatial synchronization nor discrete media is within the scope of this paper.

Steinmetz et al. Model [13, 55]. Meyer, Effelsberg, and Steinmetz presented a more sophisticated synchronization model in 1993, based on the type of synchronization demands. The model is divided into four synchronization layers: (1) *media layer*, i.e., intra-stream synchronization; (2) *stream layer*, including inter-stream synchronization and inter-receiver/group synchronization; (3) *object layer*, describing synchronization of both continuous and discrete media objects; and (4) *specification layer*, prescribing applications and tools for synchronization specification.

Ehley et al. Model [25]. Proposed in 1994, Ehley, Furth, and Ilyas classified the synchronization technologies based upon synchronization locations, i.e., places where the synchronization control schemes are performed. However, only inter-stream synchronization was investigated in each location.

As one can see, the above three synchronization classification models are, in nature, either aligned with each other or mutually orthogonal. There is no single model which is able to cover all orthogonal dimensions.

2.3.2.3 Synchronization Specification

A further understanding of the multimedia synchronization topics requires more systematic specification methods to describe the synchronization problems. This promotes a number of specification models which can generally be grouped into four categories (Fig. 2.4), according to [13]. Here, we focus on their roles in real continuous multimedia implementations and provide a comparison in Table 2.2 in Appendix II.

Axis-based specification. First proposed by Hodges et al. in 1989 [31] and later used by [44, 59], the axis-based specification method aligns media objects in either a real or virtual global timeline axis, based on the start and finish time of each object. The accessibility of this global timeline axis is owed largely by the wide deploy-

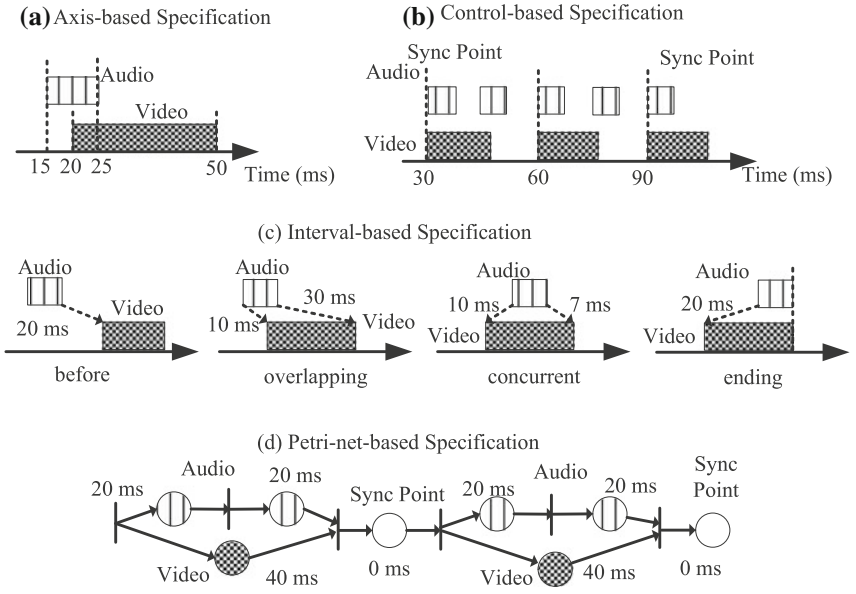


Fig. 2.4 Four synchronization specification models. Each box or circle represents a media frame

ment of NTP, and a virtual axis can be obtained by referencing the clock skews across distributed machines. The duration of each media object must be described in the specification. For example in Fig. 2.4a, we can specify that a video frame is presented between the 20th and the 50th ms, while another audio frame is played between the 15th and the 25th ms. The axis-based specification offers a direct view of temporal relations and synchronization skews of media objects in a global setting, thus facilitating its implementation in real multimedia systems. Media objects in the specification can be added and removed easily due to their mutual independence. However, media data with unknown start and finish time cannot be integrated into the axis-based method, and cannot take advantage of the benefits that this specification method provides.

Control-based specification. Developed by Steinmetz in 1990 [76] and later used by [36, 59], multimedia data are synchronized over a set of synchronization points, based on which multimedia systems can detect synchronization skews and realign the presentation of multimedia data. Oftentimes, these time points are placed periodically in order to allow consistent media re-synchronization. Figure 2.4b shows a sequence of synchronization points every 30 ms. The major advantage of this method is that it can explicitly inform users when synchronization should be performed. It also allows the integration of new media objects without major efforts. Its drawbacks are that additional mechanisms are required to specify the synchronization skews and that a timer is required to realize the periodic synchronization points.

Interval-based specification. Proposed by Wahl and Rothernel in 1994 [82] and later used by [20, 47], this specification method presents the logical tempo-

ral relations between media objects (e.g., a media object is before, after, or overlapping with another object). The exact start and finish time of each media object are unspecified. Figure 2.4c shows an example of four relations ("before", "overlapping", "concurrent", and "ending") with different delay parameter inputs. Similar to the axis-based approach, the interval-based specification is easy to understand, and adding/removing media objects is relatively simple. However, because it does not demand a knowledge of the duration of each media object and cannot describe synchronization skews, the real specification implementation can be difficult.

Petri-net-based specification. Developed by Little and Ghafoor in 1991 [51] and later used by [19, 33], this type of specification is based on Petri networks. For continuous multimedia, the specification can be described by points and arrows, where a point represents a media frame and an arrow indicates a transition state from one media frame to the other. Synchronization is achieved at each intersection of arrows. For example in Fig. 2.4d, two audio frames must be synchronized against one video frame with 0 ms synchronization skew. The Petri-net-based specification requires complex procedures to build the whole network topology. Adding and removing new media objects may also restructure the existing topology.

2.3.2.4 Synchronization Perception

As users noticed more and more audiovisual synchronization skews (i.e., perceivable lip mismatch between voices and videos) in VOD and conferencing systems over the Internet, researchers became interested in understanding the magnitude of the audiovisual skews that could be noticed by humans. A subjective study conducted by Steinmetz [77] recommends an in-sync region of a maximum 80-ms lip-sync skew when people will not perceive a lip mismatch, and an out-of-sync region of more than 160 ms when human perception can be significantly degraded. In addition, it also concludes that people are less tolerable to a skew when the video signal is behind the audio, than a skew when the audio is behind. The findings can be explained by the fact that the speed of light is much faster than the speed of sound, so people are getting accustomed to late audio signals.

In the same year, the skews between multiple acoustic streams within a microphone array were also studied in [23]. Based on subjective evaluations, it concludes that a skew of 17 ms between the stereo audio signals can be perceivable, and that a maximum skew of 11 ms is preferable.

2.3.2.5 Intra-stream and Inter-stream Synchronization Controls

To preserve temporal correlations at media presentation, a **synchronization control** scheme is often used in a multimedia system. It statically or dynamically adapts one or multiple system components, in order to mitigate synchronization skews during computation and/or distribution. Researchers began to investigate the synchronization control schemes in early 1990s, exclusively focusing on intra-stream and

inter-stream synchronization for video conferencing or on-demand systems, owing to the rapid commercialization of these Internet-based applications. Most studies in those early years focused on synchronization of a single audio and a single video stream, where the audio stream was always selected as the reference stream in the master (audio)–slave (video) synchronization prototype, mainly due to the fact that the human perception is more sensitive to the degradation of audio signals. A global time was also assumed to be available between the two signals.

In this chapter, we group different studies based on both location and functionality of the synchronization control mechanisms. For synchronization location, we investigate control algorithms located and executed at both sender and receiver sides. In terms of functionality, we classify synchronization approaches that can either be shared generically by any media modality or be applied only to specific modalities.

Receiver-based Synchronization

The buffering compensation has always been the most common approach to accommodate the jitter and to minimize the inter-stream skew. To facilitate our description, we prescribe that the sender site is n^x, and the receiver site is n^y. The network delay of a media frame $f^x_{i,j}(k)$ (within the sensory stream $s^x_{i,j}$) is $D_{\text{net}}(f^x_{i,j}(k), n^y)$, the buffering delay $D_{\text{buf}}(f^x_{i,j}(k), n^y)$, and the resulting end-to-end latency is approximately $D_e(f^x_{i,j}(k), n^y) = D_{\text{net}}(f^x_{i,j}(k), n^y) + D_{\text{buf}}(f^x_{i,j}(k), n^y)$. Hence, between two buffer status updates, the following two requirements must be satisfied:

1. Intra-stream synchronization: $\forall\ k, \quad D_e(f^x_{i,j}(k), n^y)$ must remain equal, that is, $D_e(f^x_{i,j}(k), n^y) = D_e(f^x_{i,j}(*), n^y)$.
2. Inter-stream synchronization: $\forall\ i, j, \ |D_e(s^x_{i,j}, n^y) - D_e(s^x_*, n^y)| < \delta_s$ must be satisfied, where s^x_* is the inter-stream reference, and δ_s is the **synchronization threshold** of the inter-stream skew.

A synchronization threshold is defined as the maximum allowable value of the synchronization skew. It is determined by specific synchronization demands of multimedia applications.

When D_e is decided, the buffering delay of each media frame D_{buf} can be computed by subtracting the network latency D_{net} from D_e. Please note that the computation heterogeneity was usually not considered in those early years.

The abrupt change of the buffering delay D_{buf} before and right after a synchronization control update can introduce discontinuities in the media presentation processes. Most studies address this issue and mitigate the degradations of intra-stream synchronization quality based on the following methods (e.g., [9, 49, 68, 73, 83, 84]).

- **Increasing buffering latency**. There have been multiple generic approaches that can be shared among all media modalities. For example, a cost-effective way is to replicate past media frames. A more expensive approach is to interpolate media information by data prediction based on neighboring or past media frames. There are also a number of approaches that can be applied to specific media modalities. For video buffer, we can increase the video inter-frame period. For audio buffer,

we may perform timescale modification without pith change (i.e., expanding the playout duration of each audio sample). We may also insert silence packets during silence periods between two utterances.

- **Decreasing buffering latency**. There have also been multiple generic or media-specific approaches. The most common generic approach is to simply skip the presentation of several media frames. For video buffer, we can decrease the video inter-frame period. With respect to audio buffer, we may also perform timescale modification without pitch change (i.e., reducing the playout duration of each audio sample) or remove silence packets during silence periods between two utterances.

Sender-based Synchronization

There are two key components in sender-based synchronization: the network bandwidth estimation and the resulting control scheme. An insufficient bandwidth can exert Internet congestion jitter and losses which can affect both intra-stream and inter-stream synchronization. Bandwidth estimation can be achieved either by packet pair probing [34] or by monitoring the receiver jitter and loss statistics via feedback control loop [67]. Based on the estimated bandwidth, the sender can perform one or multiple options of the synchronization control schemes [66, 68, 73, 83] which include (1) reducing the media sampling rate (e.g., changing audio sampling frequency from 16000 to 8000 Hz or video frame rate from 20 to 10 fps), (2) downgrading the media encoding quality (e.g., reducing video/audio encoded data rate), (3) skipping media data of low priority (e.g., only sending the I-frames during video streaming), and (4) discarding media frames that cannot meet the receiver presentation deadline (based on feedback messages from the receiver that indicate the current playout buffer status).

The sender and receiver synchronization control schemes can be employed jointly. Each scheme can be performed either reactively in response to Internet quality changes or preventively so as to reduce the chance of possible Internet quality degradations [16, 42].

2.3.3 Years of Blossoms: Late 1990s

Multimedia synchronization continued to be a hot topic due to the revolutionary change of the Internet quality and the development of sophisticated multimedia technologies. The study of inter-receiver/group synchronization with the design of media multicast overlay was the main topic in late 1990s.

2.3.3.1 Historical Background

The accessibility of broadband Internet became popular in late 1990s. This fostered the blossoms of multiple real-time applications, including the world's first commer-

cial Voice over Internet Protocol (VoIP) service by VocalTec in 1995 [80], the first 3D massive multiplayer online game (MMOG) by 3DO Company in 1995 [22], and the Caltech-CERN project in 1997 which built a virtual room videoconferencing system that was able to connect the research centers over the world [5].

In parallel with the development of new multimedia applications, the year of 1996 gave birth to many well-known Internet Telecommunication Union (ITU) standards on multimedia codec specifications and streaming protocols, including ITU-T H.263 [45] for low-bandwidth video codec and ITU-T H.323 [46] on packet-based multimedia communication systems. The Internet Engineering Task Force (IETF), on the other hand, proposed RFC 1889 [69], the Real-time Transport Protocol (RTP), and the RTP Control Protocol (RTCP). RTP specifies a standardized packet format for delivering streaming media over the Internet, while RTCP defines the control information for RTP data. Both ITU and IETF standards have experienced several revisions since then, and RFC 1889 has been deprecated and replaced by RFC 3550 [70].

The studies of intra-stream and inter-stream synchronization continued to prevail, due to more sophisticated multimedia applications and new multimedia standards. The evolution of multi-party conferencing systems and MMOG applications, owing to tremendously enhanced Internet bandwidth availability, had sparked massive interests in providing inter-receiver/group synchronization, for the purpose of preserving the fairness and the temporal relations among the participants.

2.3.3.2 Inter-receiver/Group Synchronization Control

Similar to intra-stream and inter-stream synchronization, inter-receiver synchronization control schemes [16, 42, 58, 59] can also be classified based on synchronization locations and synchronization control functionalities. To facilitate the description, we describe that the sender site is n^{x_0}, and the list of receiver sites is $\{n^1, n^2, \ldots\}$. We also denote that the network delay between the sender n^{x_0} and any receiver n^y is $D_{\text{net}}(u^{x_0}, n^y)$ (where u^{x_0} is the media bundle sourced at n^{x_0}), the buffering delay $D_{\text{buf}}(u^{x_0}, n^y)$, and the resulting end-to-end latency is approximately $D_e(u^{x_0}, n^y) = D_{\text{net}}(u^{x_0}, n^y) + D_{\text{buf}}(u^{x_0}, n^y)$. Here, we simplify the problem and assume negligible computation overhead at sender sites.

By denoting the synchronization reference site as n^*, the synchronization goal can be formulated by Eq. 2.8 as

$$\forall y : \quad |D_e(u^{x_0}, n^y) - D_e(u^{x_0}, n^*)| < \delta_{\text{rcv}}, \tag{2.8}$$

where δ_{rcv} is the synchronization threshold of the inter-receiver synchronization skew. To further simplify the problem, we assume zero Internet jitter in our discussion.

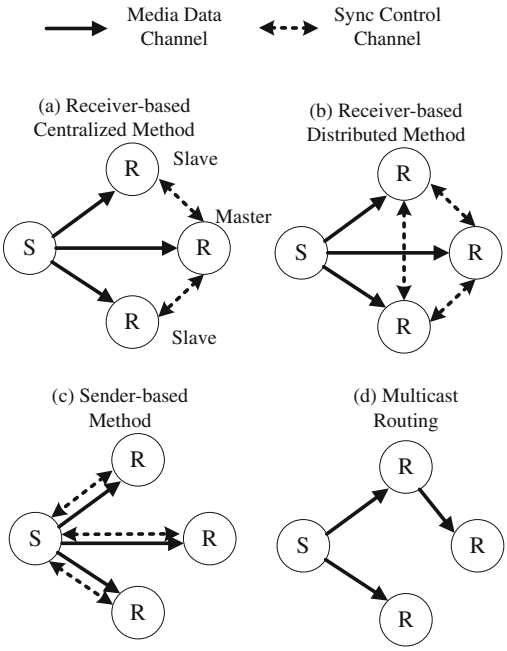


Fig. 2.5 Group synchronization control algorithms. S: sender site, R: receiver site

Receiver-based Synchronization

One or multiple receivers need to calculate the buffering delay D_{buf} without information from the sender site. Based on the synchronization functionalities, the receiver-based approaches can be further divided into two categories:

Centralized (master–slave) method (Fig. 2.5a). In this method, one master receiver is selected as the synchronization reference site n^*, and all other receiver sites are the slaves [8, 43]. Usually, n^* is chosen as the receiver with the longest D_{net} from the sender, i.e., $n^* = \arg\max_y D_{\text{net}}(u^{x_0}, n^y)$. The detailed procedure can be divided into the following four steps.

1. n^* first decides the one-way latency $D_e(u^{x_0}, n^*) = D_{\text{net}}(u^{x_0}, n^*)$, assuming that $D_{\text{buf}}(u^{x_0}, n^*) = 0$.
2. n^* multicasts $D_e(u^{x_0}, n^*)$ value to all other slave receivers.
3. Each slave receiver n^y measures individual $D_{\text{net}}(u^{x_0}, n^y)$ and decides its own target latency $D_e(u^{x_0}, n^y) = \max\left\{D_{\text{net}}(u^{x_0}, n^y),\ D_e(u^{x_0}, n^*) - \delta_{\text{rcv}}\right\}$.
4. n^y updates its buffering delay D_{buf}, which is approximately $D_{\text{buf}}(u^{x_0}, n^y) = D_e(u^{x_0}, n^y) - D_{\text{net}}(u^{x_0}, n^y)$.

While it is simple to implement the centralized method in real multimedia systems, there are multiple serious drawbacks that may hinder its efficient operation. First, the connectivity between the master and slave receivers cannot be guaranteed due to potential poor Internet conditions and firewall blocking issues. Second, a timely synchronization adaptation in response to sudden Internet changes is not

possible. Third, scalability is a common problem in the centralized method, where the computation and network resources may be a bottleneck at the master receiver. Fourth, receiver sites can easily join and leave the session in multimedia applications like MMOG. When the master site suddenly leaves without announcement, the group synchronization will fail immediately.

Distributed method (Fig. 2.5b). In this method, each receiver site decides its own buffering delay D_{buf} in a distributed fashion [41], by periodically multicasting its D_e value to each other. The overall procedure can also be divided into four steps.

1. Each receiver site n^y multicasts its current $D_e(u^{x_0}, n^y)$ value to all other receivers.
2. A specific receiver site (denoted as n^{y_1}) waits until it receives messages from all other sites. It picks the site (denoted as n^*), which usually has the largest D_e value, i.e., $n^* = \arg\max_y D_e(u^{x_0}, n^y)$.
3. n^{y_1} measures its $D_{\text{net}}(u^{x_0}, n^{y_1})$ and decides its own target latency $D_e(u^{x_0}, n^{y_1}) = \max\left\{D_{\text{net}}(u^{x_0}, n^{y_1}),\ D_e(u^{x_0}, n^*) - \delta_{\text{rcv}}\right\}$.
4. n^{y_1} updates its buffering delay D_{buf}, which is approximately $D_{\text{buf}}(u^{x_0}, n^{y_1}) = D_e(u^{x_0}, n^{y_1}) - D_{\text{net}}(u^{x_0}, n^{y_1})$.

Compared to the centralized method, frequent message exchanges among the receivers due to full-mesh communication can bring about tremendous communication overhead. In addition, because each site performs synchronization adaptations without a collaboration, the state of playout buffers of all receiver sites in the overall session may never converge under Internet dynamics (e.g., changing latency). These drawbacks prevent the adoption of the distributed method in real systems.

Sender-based (Maestro) Synchronization

Sender-based (maestro) synchronization is demonstrated in Fig. 2.5c. The receiver sites unicast individual D_{net} information to the sender site (denoted as $n^{x_0} = n^*$, which is then responsible for deciding the receiver buffering delay D_{buf} and the target end-to-end latency D_e of each receiver. The detailed procedure can be listed in five steps.

1. Each receiver site n^y measures its own latency $D_{\text{net}}(u^{x_0}, n^y)$.
2. All receiver sites unicast individual $D_{\text{net}}(u^{x_0}, n^y)$ value to the sender site n^{x_0}.
3. The sender site n^{x_0} selects the largest D_e latency among all receiver sites, i.e., $D_e^{\max}(u^{x_0}) = \max\left\{D_{\text{net}}(u^{x_0}, n^y)\right\}$. For each receiver site n^y, n^{x_0} decides its target latency $D_e(u^{x_0}, n^y) = \max\left\{D_{\text{net}}(u^{x_0}, n^y),\ D_e^{\max}(u^{x_0}) - \delta_{\text{rcv}}\right\}$.
4. n^{x_0} sends $D_e(u^{x_0}, n^y)$ value to the receiver site n^y either by unicast or multicast.
5. n^y updates its buffering delay D_{buf}, which is approximately $D_{\text{buf}}(u^{x_0}, n^y) = D_e(u^{x_0}, n^y) - D_{\text{net}}(u^{x_0}, n^y)$.

The values of D_{net}, D_{buf}, and D_e can be piggybacked in the media packet header during bidirectional media data transmission between the sender and receivers. The resulting message exchanges can be effectively minimized. In addition, the reliability is no longer a problem when receiver sites are joining and leaving a session, as long as the sender is consistently sending media data to the receivers. The sender-based

synchronization is, by far, the best method to realize the inter-receiver/group synchronization in the real systems, due to its flexibility, reliability, and the implementation easiness. However, timely synchronization adaptation is still not possible due to the round-trip latency incurred during the synchronization information exchanges.

Multicast Routing with Bounded Delay and Delay Variation

Multicast routing with bounded delay and delay variation is shown in Fig. 2.5d. It is used to control D_{net} for bounding the inter-receiver synchronization skews incurred over the Internet, rather than introducing additional buffering latencies to compensate for the skews. In multisite applications, the distribution of multimedia data from the sender to each receiver may be routed through some intermediate sites. We call it a *multicast overlay*. In designing such a topology, there can be multiple path options from the same sender to the same receiver, but via different intermediate sites. Multiple path options may feature unequal network latencies that will lead to heterogeneous inter-receiver synchronization skews. Multiple synchronization control schemes (e.g., [74, 75, 86]) have been developed to decide a multicast overlay topology with bounded inter-receiver synchronization skews. In general, the overlay design can be formulated as an optimization problem in the following form:

- **Goal**: minimizing the average D_{net} for all sender–receiver pairs.
- **Synchronization constraint** (optional): bounding the resulting delay (i.e., D_{net}) and/or delay variation (i.e., inter-receiver synchronization skew).
- **Bandwidth constraint** (optional): the inbound/outbound bandwidth utilization of each site is also a constraint.

The above problem has been proven NP-hard [86]. The optimization goal can be achieved by combining the shortest bounded path options based on the Dijkstra's algorithm as discussed in [86]. Synchronization and bandwidth constraints are realized by iterating over k-shortest path options between sender and receiver sites in order to find the one which can bound synchronization skews and/or bandwidth utilization [74, 75].

Note that if these multicast studies are employed, one must assume that multimodal multi-stream data from the same sender site follow the same distribution path to the same receiver.

Table 2.3 in Appendix II summarizes the differences of existing group control methods.

2.3.4 Years of Leaps: 2000 to Date

Modern multimedia systems are becoming more powerful in terms of accessibility of computation and network resources, more complex in terms of both hardware and software configurations, and more versatile in terms of application functionalities that can be performed. The leap of modern multimedia and networking technologies

and their integration into a single platform has led to many open synchronization problems that await researchers to investigate.

2.3.4.1 Historical Background

Due to the rapid development of wireline and wireless technologies with much better network quality, modern multimedia technological advances are mainly defined by three characteristics:

Scalability. In traditional server–client or multicast systems, bandwidth becomes a bottleneck at media servers when there is a growing number of users requesting media contents. Peer-to-peer technologies are designed to address this scalability issue, so increasing number of end clients can share the bandwidth resources by allowing end clients to request media data directly from other end clients. Applications include the prototypes of MMOG [40] and peer-to-peer TV systems [85]. The cloud infrastructure, on the other hand, offers scalable Central Processing Unit (CPU) resources by outsourcing computation tasks to remote server farms.

Mobility. Mobile devices, including cell phones and tablets, became vital parts of people's everyday lives. Multimedia data are consumed on a variety of mobile terminals with different display sizes [24].

Diversity. The wide acceptance of haptics, accelerometers, body sensors, and many other sensory media (called **Mulsemedia**) in a variety of multimedia applications offers users a completely new experience. New applications can be seen in haptic desktop [53], interactive haptic painting [12], wireless body sensory network for health monitoring [52], accelerometer-based motion analysis systems [54], and many more.

The scalable multimedia applications are composed of new system components that have new demands for multimedia synchronization [81]. The mobile backhaul has very tight temporal and frequency synchronization constraints that NTP is unable to resolve. The diversity of low-cost sensory devices also requires data synchronization and affects human perception in new use applications. Multimedia synchronization becomes an even more challenging problem because of technological developments.

2.3.4.2 Precision Time Protocol (PTP)

The wireless industry demands a more precise clock source and temporal synchronization in the range of microseconds, which NTP cannot achieve. Hence, the Institute of Electrical and Electronics Engineers (IEEE) presents the new IEEE-1588 standard, named the Precision Time Protocol (PTP) [6]. Similar to NTP, PTP also uses a *master* (i.e., the device that is synchronized against) and *slave* (i.e., the device that needs synchronization) architecture to distribute synchronization packets. But PTP is able to achieve a clock synchronization accuracy up to the range of submicroseconds in a *PTP-compliant network* (i.e., all networking devices between a

PTP master and a PTP slave need to support PTP). Compared to NTP, PTP provides the following improvements.

First, PTP packet timestamping is at the dedicated PTP chip close to the physical transmission medium, providing much better precision than NTP's application-layer measurement. The accuracy and reliability of the hardware timestamp depend on the quality of the crystal oscillator in the *PTP-compliant device* (i.e., the device that has PTP chip support). The crystal oscillator generates high-frequency pulse signals, which serve as a frequency reference to the PTP chip for high-precision timestamping. Oscillators used in PTP-compliant devices usually include Rubidium oscillator, oven-controlled crystal oscillator (OCXO), and temperature compensated crystal oscillator (TCXO). When selecting an oscillator, three factors need to be considered: (1) the time accuracy when the PTP device is freely running without a time source, (2) the short-term clock stability, and (3) the temperature-dependent clock drift. In general, rubidium provides the best quality but is also the most expensive, while TCXO is an affordable solution but with the worst stability among the three.

Second, the asymmetrical bidirectional latency introduced between an NTP master and an NTP slave is mainly caused by the delays incurred at the intermediate network devices (e.g., router or switch). To compensate for this asymmetry and provide a better clock accuracy, PTP has the notion of *transparent clock*. If a device is a transparent clock, the time that a PTP packet enters and leaves the device is recorded, and the *residence time* incurred at this device is added to the *correction field* of the PTP packet. When a PTP client decides its time drift from the PTP master, the value inside the correction field will be used, in order to compensate for the bidirectional asymmetry.

Third, for scalability, a PTP-compliant device can also be a *boundary clock*. A boundary clock can have multiple networking interfaces, where one or multiple interfaces behave as the PTP slaves of other master clocks, and the rest behave as the PTP master clocks for other slaves. When a boundary clock sees multiple master clocks from different interfaces, it uses the best master clock algorithm to select the best synchronization master, where multiple candidate master clocks are prioritized by user predefined configurations as well as clock traceability, accuracy, variance, and unique identifier.

2.3.4.3 RTP/RTCP-Based Synchronization Control Implementation

It is not until 2000s and beyond that RTP and RTCP become extensively used for real-time multimedia streaming and synchronization [15, 48, 57]. RTP defines the distribution format of media data. Three main fields are included in the RTP header that are directly related to synchronization: (1) payload type, indicating the media modality of the payload; (2) sequence number, representing the index of the RTP packet in each sensory stream for the intra-stream synchronization; and (3) timestamp, describing the local (relative) timestamp of media data units within each sensory stream, a must field for satisfying various synchronization demands. Note that RTP itself does not specify a global time status. In other words, we are unable to

identify the temporal correlation across different sensory streams, without the help of other clock synchronization algorithms or protocols.

On the other hand, RTCP provides a communication channel for synchronization control support between the streams and sites. There are mainly three types of packets supported in RTCP:

1. Receiver report (RR). The receivers send RR messages to the senders specifying the packet loss rate and the jitter statistics. The RR packets may be further extended to specify the receiver buffer status [71]. This allows the sender to dynamically perform various synchronization control adaptations based on real-time streaming quality feedback, including the bandwidth allocation, and sending media data that only meet the receiver buffer deadline (as discussed in Sect. 2.3.2).
2. Sender report (SR). The senders send SR messages periodically to the receivers. An NTP/PTP (global) timestamp field is included in the SR message in order to compute the one-way latency between each sender and receiver, and to meet various synchronization demands.
3. Source description (SDES) RTCP packet. The canonical (CNAME) identifier in SDES packet is used to associate multiple media streams from a participant in multiple correlated sessions [70]. This will be useful for inter-stream synchronization.

RFC 7272 [72] investigates the use of RTCP to achieve inter-receiver synchronization. Note that both RTP and RTCP do not natively provide support for the specification of synchronization references. Hence, the reference information must be tackled in the application implementation itself.

2.3.4.4 Synchronization Perception of New Media

There are also a number of subjective studies that have investigated the impact on the human perception of synchronization skews.

Curcio and Lundan [21] evaluated the synchronization in mobile terminals with a maximum image size of 176×144 pixels. They show that in the mobile setting with a video frame rate below 15 fps, people are more tolerant to a synchronization error when the video spatial resolution is reduced. They also conclude that the annoying threshold of lip synchronization skew can be as large as 200–300 ms due to a degraded motion smoothness.

Ghinea and Ademoye [29] conducted perceptual measurements on the impact of synchronization skews between smell sensory data (i.e., olfaction) and audiovisual content, assuming the audiovisual lip synchronization skew is zero. Their results show a synchronization threshold of -30 s when olfaction is ahead of audiovisual data, and of $+20$ s when olfaction is behind. A skew within the synchronization threshold will not be perceived by humans. The paper also evaluates the impact of synchronization skew on the acceptability of the olfactory media. Participants are

asked if "the olfactory smell was distracting" or "annoying" when the synchronization skews between olfactory and audiovisual data are introduced for different video clips. The results demonstrate that a mis-synchronization has minimal impact on the olfactory perception. Similar works have also been done in [62] which shows that people enjoy a synchronization skew of less than 5 s between olfaction and video data.

Hoshino et al. [32] measured the quality of olfactory–haptic synchronization skew. The authors conclude that the threshold of annoyance is in the range of 1–3 s.

Fujimoto et al. [26] subjectively evaluated the synchronization skew between haptics and video data. They show that a skew below 40–80 ms is hardly perceptible, and that skews greater than 300 ms are annoying.

For the (intra-media) synchronization quality of the 3D stereoscopic videos, Goldmann et al. [30] argue that a synchronization skew of 120 ms between the left and right views is satisfactory, and a skew of 280 ms can lead to poor synchronization perception. The authors also show that the human perception impact of the same synchronization skew can vary depending on heterogeneous activities.

Multiple studies have evaluated human perceptual quality of inter-receiver (or inter-destination) synchronization [28, 58, 65]. In general, people will not feel annoyance at an inter-receiver synchronization skew of less than 2 s in a social TV scenario. The number drops to 400 ms in an interactive competition, when the fairness of the game becomes a primary consideration.

There are also a number of perception-driven *adaptive media playout* (AMP) schemes for synchronization control, based on extensive subjective evaluations and feedbacks. The goal is to adapt media buffer in a way that allows people to perceive minimal noticeable differences during media presentation. These AMP schemes have been deployed in multiple applications, including video conferencing, 3D teleimmersion, and live TV multicast [35, 60, 64].

For further details on synchronization perception, readers can refer to Part 3 of the book (Chap. 10–14).

2.3.5 Remarks

Several remarks can be made from the above discussions. First, there is no classification model that can capture both multi-demand and multilocation synchronization requirements. Second, the synchronization reference is usually chosen statically (e.g., the audio for inter-stream synchronization). However, new multimedia systems are not limited to traditional conferencing and on-demand applications, and the audio information may not be the most important media data. Third, most of existing studies focus on the skews incurred over the Internet. None of them manages to investigate the heterogeneity of the computation demands and to integrate the multilocation synchronization controls systematically and consistently in a single multimedia application. In the next section, we will show that the synchronization-

related issues mentioned above have become a challenge in NG-MS. We will present solutions in addressing these issues.

2.4 Synchronization in Next-Generation Multimedia Systems

NG-MS, like 3D tele-immersion (TI), Omnidirectional video, Virtual Reality (VR), and Internet of Things (IoT) applications, relies on rich multimodal multichannel media contents to provide geographically distributed users with a joint and realistic experience. Existing synchronization models and synchronization control schemes (as discussed in Sect. 2.3) show lots of limitations because synchronization in NG-MS is characterized by the following three attributes:

1. **Demands of scale and device heterogeneity**. Multiple sensory devices with heterogeneous media modalities can be configured in an NG-MS (e.g., multi-view videos, spatial audios, etc.). This requires both intra-media and intra-bundle synchronization. The immersive environment adds the demand for inter-sender synchronization, in addition to inter-receiver synchronization, to preserve the seamless interaction among both the sender and receiver sites.
2. **Multilocation synchronization controls**. An NG-MS can generally be divided into multiple locations, each of which can affect the synchronization skews. As an example, let us consider the TI system shown in Fig. 2.6. At the *capturing tier*, the sender site captures time-dependent multimodal media frames and encodes them in real time. The computation heterogeneity can contribute to the skews in all synchronization layers, as defined in Sect. 2.2.2. At the *distribution tier*, multimodal multi-stream data are sent from each sender gateway to multiple receivers. Synchronization skews are mainly caused by the Internet jitter and the use of an overlay network to distribute media contents. At the *presentation tier*, the multimedia streams are decoded and played at the corresponding output devices. The buffering latency is often introduced to compensate for the synchronization skew that has accumulated so far.
3. **Diverse applications on a single multimedia platform**. A variety of applications can be served on a single TI platform, including media consulting, remote education, and collaborative gaming. Different media modalities and sensory streams can have varying contributions to the functionality of each application, so they will have a different impact on the human perception [36]. Because the synchronization references usually represent the most important media information against which to synchronize, they must be selected depending on the user activities and their specific underlying application functionalities.

New synchronization attributes, arising from next-generation advanced multimedia and networking technologies, are not fully addressed in existing practices and standards. Hence, we will present next a multidimensional synchronization model that aims to work in the setting of NG-MS.

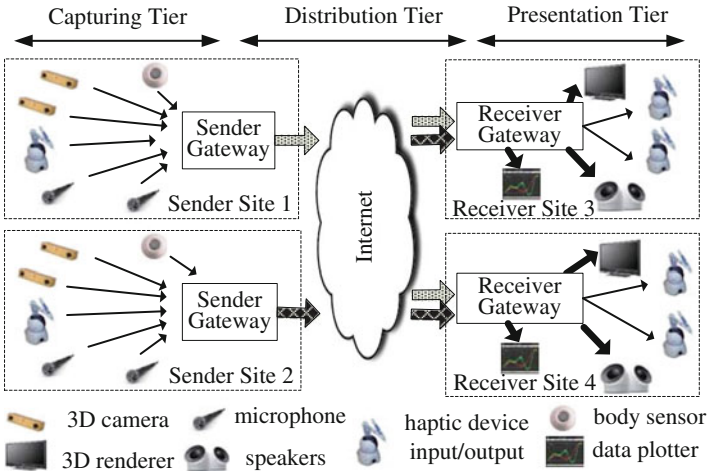


Fig. 2.6 The general architecture of TI systems

2.4.1 New Multidimensional Synchronization Classification Model

The scale and device heterogeneity, the multilocation synchronization control, and the application diversity of NG-MS, like the TI system, require considerations of three orthogonal dimensions in a synchronization model when performing temporal multimedia synchronization evaluations. The past models only captured one of the dimensions (e.g., Ehley's model considered only the location, while Steinmetz's model considered only the device heterogeneity). Here, we present a possible next-generation multidimensional synchronization model (Fig. 2.7), which includes the following:

1. **Dimension of scale and device heterogeneity**. This dimension is based on Steinmetz's model [13, 55]. It includes five layers. Four layers are used to meet the synchronization demands that we have discussed: intra-stream, intra-media, intra-bundle, and intra-session layers. The object layer in Steinmetz's model is removed because we only focus on continuous multimedia streams. The fifth layer, the specification layer, is used to specify the synchronization requirements of a multimedia application. Depending on the availability of sensory devices and participant sites, a multimedia application may only need a subset of the four synchronization demands. The specification layer also specifies the synchronization references and defines synchronization skews. Because the synchronization references may be updated online throughout a multimedia session, the specification layer should recompute synchronization skews based on the new references accordingly.

2. **Dimension of multilocation synchronization controls**. The orthogonal location-based dimension is directly extended from Ehley's model [25]. The location can either be a subcomponent of the media processing pipeline or an aggregation point during media distribution. We believe that there is a need to achieve multimedia synchronization in all locations, so that synchronization skews in one location will not be propagated to future locations. Hence, the multidimensional synchronization model adds the synchronization control at each location together with temporal support for large scale of heterogeneous devices.
3. **Dimension of application-dependent synchronization**. We argue that there is a strong demand to add this additional dimension, because NG-MS has a wider use space that can have numerous applications in different contexts. The dimension is used to describe the impact of the application heterogeneity on human perception of multimedia synchronization. It is not possible to use uniform synchronization references in a multimedia system that can have multiple applications. Each application achieved by a multimedia system must identify its own references based upon the functionality of performed activities and end user interests. Please note that this dimension must determine the synchronization references and work jointly with the specification layer in the dimension of scale and device heterogeneity, so that synchronization skews can be formulated based on specific applications. Appendix III presents an example of synchronization reference selection policy used in our current TI implementation.

2.4.2 Multilocation Collaborative Synchronization Controls

To demonstrate the usage of the multidimensional synchronization model in Fig. 2.7, we present the multilayer temporal synchronization control scheme at multiple locations (tiers) of the TI system, shown in Fig. 2.6. We rely on the RTP/RTCP protocol stack to achieve TI synchronization implementation.

2.4.2.1 Capturing Tier Control

The purpose of the capturing tier control is to constrain the synchronization skews arising from the computation heterogeneity of multimodal media data sourced at the same sender site. The heterogeneity is due to the fact that multiple time-correlated media frames can carry different amounts of media information, which require unequal CPU resources. The resulting variations of the computation overhead within and across the sensory streams cause intra-stream, intra-media, and intra-bundle skews.

To achieve bounded skews at the capturing tier, we utilize CloudStream [37], a cloud-based media encoding parallelization and scheduling scheme for data-intensive media, like 3D multi-view videos. The computation tasks are outsourced to cloud server farms to generate multiple resolutions of video streams to support a growing

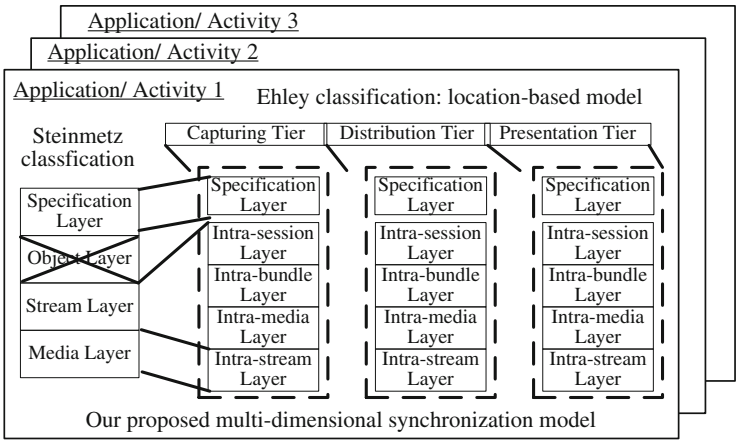


Fig. 2.7 The multidimensional synchronization model

number of end devices that demand different video qualities. In CloudStream, the media parallelization pipeline speeds up the computation process of data-intensive media modalities, by dividing each media frame (e.g., multi-view image) into multiple nonoverlapping data partitions, and encoding these partitions in parallel on multiple cloud nodes. This can effectively reduce the intra-bundle skews when comparing to media modalities with negligible overhead. The system scheduling component decides the correct amounts of computation resources (e.g., number of requested cloud nodes) for parallel encoding of each media frame. Different media frames of a same sensory stream may use different computation resources. This allows much reduced computation jitter (i.e., difference in encoding time) among media frames of a same sensory stream and across multiple streams. The computation jitter is closely related to the intra-media and intra-stream synchronization.

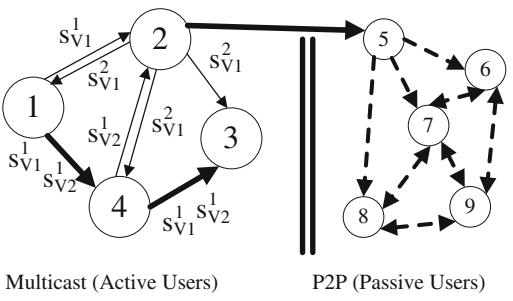


Fig. 2.8 A hybrid (multicast+P2P) synchronized distribution topology

2.4.2.2 Distribution Tier Control

The goal is to design an overlay topology with bounded synchronization skews during the media distribution over the Internet. A TI system is a combination of interactive and on-demand applications, because some *active receivers* are participating in or will join the shared activity, while other *passive receivers* are simply watching the active users conducting activities. The active receivers produce and send media packets, so they demand a much better interactive quality (lower latency) than the passive sites who only receive the media streams. Hence, we present a **hybrid** approach [10, 38], by performing media multicast among the active sites while relying on a peer-to-peer overlay for the rest of passive sites (Fig. 2.8). The hybrid approach only focuses on the multi-view video distribution and the resulting intra-session and intra-media (video) synchronization. Audio, haptic, and other media modalities are assumed to add negligible bandwidth overhead, so their packets can be multiplexed and follow the same distribution path as the video reference stream in the same media bundle. In other words, the intra-bundle and intra-media (audio, haptic, etc.) skews have already been minimized during the media distribution. In addition, the Internet jitter and the resulting intra-stream synchronization are not studied in this hybrid approach, and we assume they will be addressed in the presentation tier.

- **Multicast overlay**. The multicast overlay, proposed by Huang et al. [38], is based on solutions in [74, 75] (Sect. 2.3.3.2), which iterate over k-shortest path options for each sender–receiver pair, in order to find the paths which can achieve both synchronization and bandwidth constraints. Huang et al. [38] make three major extensions to [74, 75]. First, multiple video streams within the same media bundle are allowed to follow different paths from the same sender to the same receiver. For example, Fig. 2.8 shows that site 1 decides to multicast two video streams using different overlays: $s^1_{V,1}$ to both sites 2 and 4, and $s^1_{V,2}$ only to site 4. Hence, site 2 has to receive $s^1_{V,2}$ via the intermediate site 4. Second, previous studies only address the inter-receiver/group synchronization problem, while the overlay by Huang et al. [38] adds the constraints of both intra-media (video) and inter-sender synchronization to the problem formulation. For intra-media synchronization, all video streams captured by multiple cameras at the same site need to be synchronized, so that there is no inconsistency when changing views of that site. For inter-sender synchronization, multiple media bundles captured by different sender sites also need to be synchronized, so that the receiver sites will not watch these media bundles with temporal inconsistency. Third, the video reference streams now have priority in allocating bandwidth resources to preserve the most important synchronization information.
- **Peer-to-peer overlay**. Arefin et al. [10] follow existing studies in [85] to build a peer-to-peer distribution overlay, based on peer availability and bandwidth utilization fairness. Each video stream has its own individual distribution overlay, so multiple video streams from the same sender site can also follow different peer-to-peer paths to the same receiver site. Each peer-to-peer distribution overlay has a limited number of intermediate peer sites on the distribution path. This bounds the

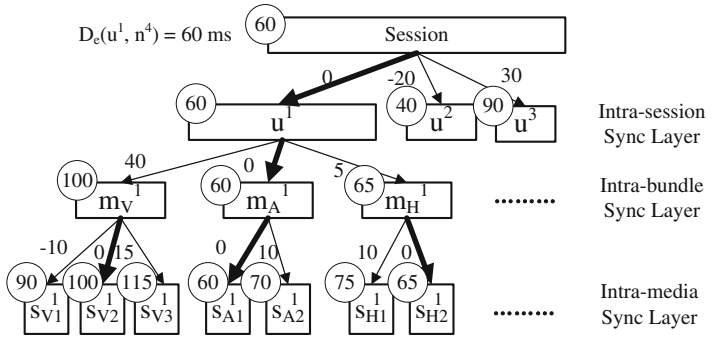


Fig. 2.9 Synchronization tree specification model

one-way video distribution latency and also constrains both intra-media (video) and intra-bundle (inter-sender and inter-receiver) skews.

2.4.2.3 Presentation Tier Control

The goal of the presentation tier control is to add buffer compensation and to bound the multilayer synchronization skews that are propagated from the capturing and distribution tiers. Existing buffer control algorithms and protocols in Sects. 2.3.2 and 2.3.3 must be extended to integrate the hierarchical synchronization references (one reference for each synchronization layer). To systematically model and visualize the interaction of synchronization layers during the buffer adaptations under scalable system configurations, we present a novel *synchronization tree* specification model.

Figure 2.9 shows an example of the specification tree formed by the receiver site n^4, where the inter-sender synchronization is demanded in the intra-session layer. Each vertex in the tree indicates a media object (i.e., session, media bundle, media modality, and media stream). The bolded edges denote the synchronization reference in each synchronization layer (see Appendix III for TI applications). The edge cost represents the synchronization skews, relative to the synchronization reference, of the media object during its presentation. In other words, the edge cost is ΔD_e, where D_e is the end-to-end latency of each media object after the buffer compensation, as discussed in Sect. 2.2.3. The vertex value (i.e., the circled number) specifies the D_e value of the corresponding media object, which can be computed by summing the edge costs on the path from the tree root to the current vertex, plus the root value. For example, we assume in Fig. 2.9 that the root value $D_e(u^1, n^4)$ is 60 ms. Hence, $\Delta D_e(u^2, n^4) = 60 - 20 = 40$ ms, $\Delta D_e(m_H^1, n^4) = 5$ ms, and $D_e(m_H^1, n^4) = 60 + 0 + 5 = 65$ ms. Note that the intra-stream skews should be zero due to the buffer compensation, so they are omitted in Fig. 2.9.

Based on the tree model, Huang et al. [39] propose a new buffer control algorithm. It uses an iterative approach to decide the minimal D_e of all media objects that

satisfy the synchronization constraints (i.e., with bounded synchronization skews in all synchronization layers).

2.5 Conclusion

We have seen multiple generations of multimedia systems, and particularly in the past two decades, owing to the rapid development and availability of broadband Internet technologies, computation powers, and high-quality media sensors. We have shown that multimedia synchronization has always been a challenge, and lots of synchronization research works have been done in the area of models, protocols, control algorithms, distribution network, subjective perception, etc. We have defined multi-demand synchronization requirements in multiple layers and formulate synchronization skews. We have grouped achievements of synchronization research into four primary generations, based on human understanding and technological development of multimedia and synchronization systems.

In the years of birth (in and before 1980s) when digital media technologies were not mature and the Internet was still new to most people, researchers mostly focused on understanding synchronization of analog media, and NTP was proposed to achieve Internet clock synchronization at a coarse granularity. In the years of understanding (early 1990s) when the Internet became gradually adopted and multiple digital video and audio applications were invented, researchers proposed classification and specification models to understand and describe the synchronization problems. Subjective evaluations and control algorithms were mostly done for stereo audio and audiovisual synchronization, which were mostly needed during these years. In the years of blossoms (late 1990s) when broadband Internet became more available and there were growing number of multi-party multimedia applications, lots of research works were done for inter-receiver or group synchronization, in the area of video multicast, multi-party conferencing and MMOG. In the years of leaps (2000 to date) when the Internet has been part of daily life, there have been a growing number of users demanding heterogeneous of media contents via both wireline and wireless networks. Synchronization has become a larger challenge because of system scalability, demand for high-precision clock distribution over wireless medium, and human subjective perception on heterogeneous media data presented on multiple forms of end devices (TVs, PCs, and mobiles).

In the future, we foresee a revolution of distributed multimedia systems with a wider variety of multimodal sensory devices, diversity of applications and activities, and complexity of spatial and other contextual information. Due to major advances in multimodal devices, IoT, distributed and mobile computing and network technologies, and due to the drop of their integration cost, these systems are already deployed and will be deployed at much faster pace and in a much broader applications and user environments such as VR and TI spaces, smart homes and smart cities, telehealth, and other applications. Although we only use TI system as an example, we believe our generic multidimensional multi-contextual synchronization model can be

extended to other applications. Multimedia data will be generated everywhere making use of a large variety of devices. Multimedia data will also be aggregated at the edges of a network as well as in the network. The evolution of multimedia systems is consistently posing new synchronization challenges. These challenges require to revisit past and current synchronization practices and standards, and demand development of new contextual-dependent approaches and principles as new multimedia environments arise.

Appendix

Appendix I: Mathematical Symbols and Denotations

Table 2.1 summarizes the mathematical symbols and denotations in this chapter.

Table 2.1 Mathematical symbols and denotations

Symbols	Denotations
t	Time
δ	Clock offset between two computing machines
x	Site index
y	Site index
i	Media modality index. $i = 1$ or "V": videos,
	$i = 2$ or "A": audios, $i = 3$ or "H": haptics
j	Sensory stream index
k	Media frame index
$*$	Synchronization reference index
n^x	Site x
n^*	Intra-session synchronization reference site
u^x	Media bundle outputted by n^x
u^*	Synchronization reference media bundle outputted by n^*
m_i^x	i-th media modality outputted by n^x
m_*^x	Intra-bundle synchronization reference modality outputted by n^x
$s_{i,j}^x$	j-th sensory stream of m_i^x outputted by n^x
$s_{i,*}^x$	Intra-media synchronization reference stream of m_i^x outputted by n^x
s_*^x	Inter-stream synchronization reference stream of u^x outputted by n^x
$f_{i,j}^x(k)$	k-th media frame of $s_{i,j}^x$ of m_i^x outputted by n^x
$f_{i,j}^x(*)$	Intra-stream synchronization reference frame of $f_{i,j}^x(k)$ outputted by n^x
D	Experienced latency of a media object
D_{net}	Latency incurred over the network
D_{buf}	Latency incurred during buffer control

(continued)

Table 2.1 (continued)

Symbols	Denotations
D_e	End-of-end latency
$D(u^x, n^y)$	Latency of u^x from its captured time, when it is being delivered to n^y
$D(m_i^x, n^y)$	Latency of m_i^x from its captured time, when it is being delivered to n^y
$D(s_{i,j}^x, n^y)$	Latency of $s_{i,j}^x$ from its captured time, when it is being delivered to n^y
$D(f_{i,j}^x(k), n^y)$	Latency of $f_{i,j}^x(k)$ from its captured time, when it is being delivered to n^y
$\Delta D(u^x, n^{y_0})$	Intra-session (inter-sender) synchronization skew of u^x against u^*,
	when it is being delivered to receiver site n^{y_0}
$\Delta D(u^{x_0}, n^y)$	Intra-session (inter-receiver) synchronization skew of u^{x_0} against n^*,
	when it is being delivered to receiver site n^y
$\Delta D(m_i^x, n^y)$	Intra-bundle synchronization skew of m_i^x against m_*^x,
	when it is being delivered to receiver site n^y
$\Delta D(s_{i,j}^x, n^y)$	Either intra-media synchronization skew of $s_{i,j}^x$ against $s_{i,*}^x$,
	or inter-stream synchronization skew of $s_{i,j}^x$ against s_*^x,
	when it is being delivered to receiver site n^y
$\Delta D(f_{i,j}^x(k), n^y)$	Intra-stream synchronization skew of $f_{i,j}^x(k)$ against $f_{i,j}^x(*)$,
	when it is being delivered to receiver site n^y
$\mathbf{O}(s_{\mathrm{V},i}^x))$	Camera orientation of $s_{\mathrm{V},i}^x$
$\mathbf{O}^{x,y}$	Desired view orientation of n^x's videos for receiver site n^y
$\mathrm{CF}(s_{\mathrm{V},i}^x, n^y)$	Contribution factor of $s_{\mathrm{V},i}^x$ to the receiver site n^y

Appendix II: Comparison Summary of Synchronization Studies

We summarize two comparison tables for the synchronization studies we have discussed in Sect. 2.3. Table 2.2 is for discussing the synchronization specification models in Sect. 2.3.2.3. Compared to interval-based and Petri-net-based specification models, Table 2.2 shows that both axis-based and control-based specification models are easy to implement and add/remove media objects, but still they require additional information and mechanisms during synchronization specifications.

Table 2.2 Comparisons of four specification models discussed in Sect. 2.3.2

Specification models	Axis	Control	Interval	Petri-net
Implementation	Easy	Easy	Complex	Complex
Media objects	Independent	Independent	Dependent	Dependent
Adding/Removing media objects	Easy	Easy	Complex	Complex
Media object duration	Required	Not required	Not required	Required
Synchronization skew	Supported	Need additional mechanism	Supported	Supported

Table 2.3 Comparisons of inter-receiver/group synchronization control algorithms

Control algorithms	Receiver-based (Master–slave)	Receiver-based (Distributed)	Sender-based (Maestro)	Multicast routing
Centralized/distributed	Centralized	Distributed	Centralized	Centralized
Adding/removing receivers	Complex if master is changed	Easy	Easy	Complex
Communication overhead	Medium	Large	Small	Large
Adaptation responsiveness	Round-trip delay	Slow	Round-trip delay	N/A

Table 2.3 is for evaluating the inter-receiver/group synchronization control algorithms in Sect. 2.3.3.2. In general, centralized approaches have lower communication overhead, and adaptive responsiveness is much faster when compared to distributed approaches.

Appendix III: Synchronization Reference Selection in Tele-immersive (TI) System

In this section, we present an example of synchronization reference selection methodology in our current TI implementation. Note that the selection rule is policy-based, meaning that it can vary depending on specific end user interests in different multimedia applications.

Intra-stream Synchronization

The reference frame or the intra-stream synchronization reference is usually selected as the first media frame within a sensory stream at each system control update. Hence, other media frames behind it can be played at the output devices by consulting their original captured inter-frame periods at the media sensor.

Intra-media Synchronization

The intra-media synchronization reference is selected as the reference stream which has the largest contribution to end user interests within a media modality. The media contribution can vary depending on the characteristics of each modality. Here, we discuss four commonly deployed media modalities which we have used.

Multi-view videos. Multi-view video streams capture the same physical object at the same time, but from different viewpoints. The importance of each video stream is decided by their contributions of 3D image pixels to the end user viewpoint [36], which can be computed using the orientation difference between the sender camera and the receiver view. Given the sender n^x's camera orientation of a video stream $s^x_{\mathrm{V},i}$ (denoted as $\mathbf{O}(s^x_{\mathrm{V},i})$), and the desired view orientation of n^x's videos for receiver site n^y (denoted as $\mathbf{O}^{x,y}$), the visual contribution or the *contribution factor* (CF) of $s^x_{\mathrm{V},i}$ to the receiver site n^y is defined by 2.9 as

$$\mathrm{CF}(s^x_{\mathrm{V},i}, n^y) = \mathbf{O}(s^x_{\mathrm{V},i}) \cdot \mathbf{O}^{x,y} \tag{2.9}$$

Hence, the video reference stream is elected as the video stream with the largest CF within the video modality for each receiver.

Spatial audios. Multiple omnidirectional microphones concurrently record the same physical ambient environment. The contribution of each audio stream is decided by its signal-to-noise ratio (SNR), a metric indicating the intelligibility of the speaker's utterances. SNR can be computed online by estimating the noises during silence periods. We prescribe that the audio reference stream is the audio stream with the largest SNR within the audio modality.

Haptics or Body sensory streams. Multiple haptic or body sensory streams may record different parts of a physical object. In the TI systems, we decide the haptic/body reference stream as the one with the largest data rate within the haptic/body sensory modality, because a larger data rate for these sensory streams usually means higher precision information.

Intra-bundle Synchronization

The importance of media modalities can vary at different applications, and the intra-bundle synchronization reference is defined as the most important reference modality. Empirically, for TI systems, we can classify different applications based on real user perceptual feedback. (1) Users attach more importance to the intelligibility of audio signals in a conversation-oriented application (e.g., conferencing or remote education), so the reference modality is the audio. (2) The clarity of video signals is of the greatest significance in a collaborative task with fine motor skills (e.g., rock–paper–scissor gaming or cyber-archeology), so the video is selected as the reference modality. (3) The body sensory streams can have the largest contribution in the telehealth or the remote rehabilitation application, because the doctors need to evaluate the patient's health status by consistent body sensory feedback. Thus, we choose the body sensory modality as the reference.

Intra-session Synchronization

In multisite interactive multimedia systems, the most active site usually demands higher quality streaming bundles in order to guarantee uninterrupted collaborations in a session. The intra-session synchronization reference of inter-sender or inter-receiver synchronization is, thus, selected as the media bundle corresponding to the most active user among all senders or receivers. In the TI systems, for example, this user usually takes the lead in the multimedia applications (e.g., a trainer in the remote education, a director in the conferencing, or a doctor in the telehealth). The selection of the lead person is context-dependent, so it must be specified explicitly by the media applications.

References

1. BELLLABS: The picture of the future. Bell Labs Rec. **47**(5), 134–186 (1969)
2. RFC-958: Network Time Protocol (NTP). http://www.ntp.org/. Accessed 28 Apr 2017
3. Cornell University: The CU-SeeMe Project. http://ftp.icm.edu.pl/packages/cu-seeme/html/Welcome.html. Accessed 28 Apr 2017
4. The Cambridge iTV Trial. http://koo.corpus.cam.ac.uk/projects/itv/. Accessed 28 Apr 2017
5. Caltech/CERN Project. http://pcbunn.cithep.caltech.edu/. Accessed 28 Apr 2017
6. IEEE-1588 standard: Precise time synchronization as the basis for real time applications in automation. https://standards.ieee.org/findstds/standard/1588-2008.html. Accessed 28 Apr 2017
7. RFC-5905: Network Time Protocol version 4: Protocol and algorithms specification. http://www.ntp.org/. Accessed 28 Apr 2017
8. Akyildiz, I.F., Yen, W.: Multimedia group synchronization protocols for integrated services networks. IEEE J. Sel. Areas Commun. **14**(1), 162–173 (1996)
9. Anderson, D.P., Homsy, G.: A continuous media I/O server and its synchronization mechanism. IEEE Comput. **24**(10), 51–57 (1991)
10. Arefin, A., Huang, Z., Nahrstedt, K., Agarwal, P.: 4D Telecast: Towards large scale multi-site and multi-view dissemination of 3DTI contents. In: Proceedings of IEEE 32nd International Conference on Distributed Computer Systems (ICDCS), Macau, China, pp. 82–91 (2012)
11. Basilio, C.: Antonio meucci inventore del telefono. Notiziario Tec. Telecommun. Ital. **12**(1), 114 (2003)
12. Baxter, B., Scheib, V., Lin, M.C., Manocha, D.: DAB: interactive haptic painting with 3D virtual brushes. In: Proceedings of ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), Los Angeles, CA, pp. 461–468 (2001)
13. Blakowski, G., Steinmetz, R.: A media synchronization survey: reference model, specification, and case studies. IEEE J. Sel. Areas Commun. **1**, 5–35 (1996)
14. Blesser, B.: Digitization of audio: a comprehensive examination of theory, implementation, and current practice. AES J. Audio Eng. Soc. **26**(10), 739–771 (1978)
15. Boronat, F., Cebollada, J.C.G., Mauri, J.L.: An RTP/RTCP based approach for multimedia group and inter-stream synchronization. Springer J. Multimedia Tools Appl. **40**(2), 285–319 (2008)
16. Boronat, F., Lloret, J., Garcia, M.: Multimedia group and inter-stream synchronization techniques: a comparative study. Elsevier Inf. Syst. **34**(1), 108–131 (2009)
17. Bulterman, D.: Specification and support of adaptable networked multimedia. Springer Multimedia Syst. **1**(2), 68–76 (1993)
18. Campbell, A., Coulson, G., Garcła, F., Hutchison, D.: Orchestration services for distributed multimedia synchronisation. In: Proceedings of IFIP International Conference on High Performance Networking (HPN), Liegel, Belgium (1992)
19. Chung, S.M., Pereira, A.L.: Timed petri net representation of SMIL. IEEE Multimedia **12**(1), 64–72 (2005)
20. Courtiat, J., de Oliveira, R.C.: Proving temporal consistency in a new multimedia synchronization model. In: Proceedings of ACM International Conference on Multimedia (MM), Boston, USA, pp. 141–152 (1996)
21. Curcio, I., Lundan, M.: Human perception of lip synchronization in mobile environment. In: Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Espoo, Finland, pp. 1–7 (2007)
22. Damer, B.: Avatars: Exploring and Building Virtual Worlds on the Internet, pp. 383–386 . Peachpit Press (1998)
23. Dannenberg, R., Stern, R.: Experiments concerning the allowable skew of two audio channels operating in the stereo mode. Pers. Commun. (1993)
24. Deventer, M., Stokking, H., Hammond, M., Cesar, P.: Standards for multi-stream and multi-device media synchronization. IEEE Commun. Mag. **54**(3), 16–21 (2016)

25. Ehley, L., Furth, B., Ilyas, M.: Evaluation of multimedia synchronization techniques. In: Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS), Boston, USA, pp. 110–119 (1994)
26. Fujimoto, T., Ishibashi, Y., Sugawara, S.: Influences of inter-stream synchronization error on collaborative work in haptic and visual environments. In: Proceedings of IEEE Symposium on Haptic Interfaces for Virtual Environment and Teleoperator System (HAPTICS), Reno, USA, pp. 113–119 (2008)
27. Gardner, B.: A realtime multichannel room simulator. In: Proceedings of 124th Meeting of the Acoustical Society of America, New Orleans, USA (1992)
28. Geerts, D., Vaishnavi, I., Mekuria, R., van Deventer, O., Cesar, P.: (2011) Are we in sync? Synchronization requirements for watching online video together. In: Proceedings of the 29th ACM Conference on Human Factors in Computing Systems (SIGCHI), Vancouver, Canada, pp. 311–314
29. Ghinea, G., Ademoye, O.A.: Perceived synchronization of olfactory multimedia. IEEE Trans. Syst. Man Cybern. **40**(4), 657–663 (2010)
30. Goldmann, L., Lee, J.S., Ebrahimi, T.: Temporal synchronization in stereoscopic video: Influence on quality of experience and automatic asynchrony detection, hong kong, china. In: Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 3241–3244 (2010)
31. Hodges, M., Sasnett, R., Ackerman, M.: Athena Muse: a construction set for multimedia applications. IEEE Softw. **6**(1), 37–43 (1989)
32. Hoshino, S., Ishibashi, Y., Fukushima, N., Sugawara, S.: Qoe assessment in olfactory and haptic media transmission: Influence of inter-stream synchronization error. In: Proceedings of IEEE International Workshop on Communications Quality and Reliability (CQR), Naples, FL, USA, pp. 1–6 (2011)
33. Hsu, P., Chen, Y., Chang, Y.: STRPN: a petri-net approach for modeling spatial-temporal relations between moving multimedia objects. IEEE Trans. Softw. Eng. **29**(1), 63–76 (2003)
34. Hu, N., Steenkiste, P.: Estimating available bandwidth using packet pair probing. Carnegie Mellon University Techical Report, CMU-CS-02-166 (2002)
35. Huang, Z., Nahrstedt, K.: Perception-based media packet scheduling for high-quality teleimmersion. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM), Orlando, USA, pp. 29–34 (2012)
36. Huang, Z., Wu, W., Nahrstedt, K., Arefin, A., Rivas, R.: TSync: A new synchronization framework for multi-site 3D tele-immersion. In: Proceedings of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Amsterdam, the Netherlands, pp. 39–44 (2010)
37. Huang, Z., Mei, C., Li, L., Woo, T.: CloudStream: delivering high-quality streaming video through a cloud-based H.264/SVC proxy. In: Proceedsings of IEEE International Conference on Computer Communications (INFOCOM), Shanghai, China, pp. 201–205 (2011)
38. Huang, Z., Wu, W., Nahrstedt, K., Rivas, R., Arefin, A.: Synccast: synchronized dissemination in multi-site interactive 3D tele-immersion. In: Proceedings of ACM Multimedia Systems Conference (MMSYS), San Jose, USA, pp. 69–80 (2011)
39. Huang, Z., Nahrstedt, K., Liang, K.: Human-centric multi-layer synchronization scheme with inter-sender synchronization skew control. In: Proceedings of IEEE International Workshop on Quality of Multimedia Experience (QoMEX), Singapore, pp. 25–30 (2014)
40. Iimura, T.: Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games. In: Proceedings of ACM Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames), Portland, Oregon, pp. 116–120 (2004)
41. Ishibashi, Y., Tasaka, S.: A distributed control scheme for group synchronization in multicast communications. In: Proceedings of International Symposium Communications (ISCOM), Japan, pp. 317–323 (1999)
42. Ishibashi, Y., Tasaka, S.: A comparative survey of synchronization algorithms for continuous media in network environments. In: Proceedings of IEEE Conference on Local Computer Networks (LCN), Tampa, FL, USA, pp. 337–348 (2000)

43. Ishibashi, Y., Tsuji, A., Tasaka, S.: A group synchronization mechanism for stored media in multicast communications. In: Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Kobe, Japan, pp. 692–700 (1997)
44. ISO International Standard: Information technology hypermedia/time-based structuring language (HyTime). https://www.iso.org/standard/18834.html (1992). Accessed 28 Apr 2017
45. ITU-H263: Video coding for low bit rate communication. http://www.itu.int/rec/T-REC-H.263/en/ (2005). Accessed 28 Apr 2017
46. ITU-H323: Packet-based multimedia communications systems. http://www.itu.int/rec/T-REC-H.323/en/ (2009). Accessed 28 Apr 2017
47. King, P.: Towards a temporal logic based formalism for expressing temporal constraints in multimedia documents. Technical Report, 942, LRI, Universite de Paris-Sud, Orsay, France (1994)
48. Leroux, P., Verstraete, V., De Turck, F., Demeester, P.: Synchronized interactive services for mobile devices over IPDC/DVB-H and UMTS. In: Proceedings of IEEE/IFIP International Workshop on Broadband Convergence Networks (BCN), Munich, Germany, pp. 1–12 (2007)
49. Little, T.: A framework for synchronous delivery of time-depdent multimedia data. Springer Multimedia Syst. **1**(2), 87–94 (1993)
50. Little, T., Ghafoor, A.: Synchronization and storage models for multimedia objects. IEEE J. Sel. Areas Commun. **8**(3), 413–427 (1990)
51. Little, T., Ghafoor, A.: Spatio-temporal composition of distributed multimedia objects for value-added networks. IEEE Comput. **24**(10), 42–50 (1991)
52. Lo, B., Thiemjarus, S., King, R., Yang, G.: Body sensor network - a wireless sensor platform for pervasive healthcare monitoring. In: Proceedings of IEEE International Conference on Pervasive Computing (PERCOM), pp. 77–80 (2005)
53. Marcheschi, S., Portillo ,O., Raspolli, M., Avizzano, C., Bergamasco, M.: The haptic desktop: a novel 2D multimodal device. In: Proceedings of IEEE International Conference on Robot and Human Interactive Communication (ROMAN), Kurashiki, Okayama, Japan, pp. 521–526 (2004)
54. Mayagoitia, R.E., Nene, A.V., Veltink, P.H.: Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems. Elsevier J. Biomech. **35**(4), 537–542 (2002)
55. Meyer, T., Effelsberg, W., Steinmetz, R.: A taxonomy on multimedia synchronization. In: Proceedings of IEEE Workshop on Future Trends of Distributed Computing Systems, Lisbon, Portugal, pp. 97–103 (1994)
56. Michel, U.: History of acoustic beamforming. In: Proceedings of Berlin Beamforming Conference (BeBeC), Berlin, Germany (2006)
57. Montagud, M., Boronat, F.: On the use of adaptive media playout for inter-destination synchronization. IEEE Commun. Lett. **15**(8), 863–865 (2011)
58. Montagud, M., Boronat, F., Stokking, H., van Brandenburg, R.: Inter-destination multimedia synchronization: schemes, use cases and standardization. Springer Multimedia Syst. **18**(6), 459–482 (2012)
59. Montagud, M., Boronat, F., Stokking, H., César, P.: Design, development and assessment of control schemes for IDMS in a standardized RTCP-based solution. Elsevier Comput. Netw. **70**(1), 240–259 (2014)
60. Montagud, M., Boronat, F., Roig, B., Sapena, A.: How to perform AMP? Cubic adjustments for improving the QoE. Elsevier Comput. Commun. **103**, 61–73 (2017)
61. Montagud Climent, M.A., Jansen, AJ., Cesar Garcia, PS., Boronat, F.: Review of media sync reference models: Advances and open issues. Media Synchronization Workshop (MediaSync), Brussels, Belgium (2015)
62. Murray, N., Lee, B., Qiao, Y., Muntean, G.:The influence of human factors on olfaction based mulsemedia quality of experience. In: Proceedings of IEEE International Conference on Quality of Multimedia Experience (QoMEX), Lisbon, Portugal, pp. 1–6 (2016)
63. PictureTel: Picturetel In Project With I.B.M., New York Times. http://www.nytimes.com/1991/10/22/business/company-news-picturetel-in-project-with-ibm.html (1991). Accessed 28 Apr 2017

64. Rainer, B., Timmerer, C.: A quality of experience model for adaptive media playout. In: Proceedings of IEEE International Workshop on Quality of Multimedia Experience (QoMEX), Singapore, pp. 1–4 (2014)
65. Rainer, B., Petscharnig, S., Timmerer, C.: Is one second enough? Evaluating QoE for inter-destination multimedia schronization using human computation and crowdsourcing. In: Seventh International Workshop on Quality of Multimedia Experience, pp. 1–6 (2015)
66. Ramanathan, S., Rangan, P.: Feedback techniques for intra-media continuity and inter-media synchronization in distributed media systems. Comput. J. Oxford Univ. Press **36**(1), 19–31 (1993)
67. Ramanathan, S., Rangan, P.V.: Continuous media synchronization in distributed multimedia systems. In: Proceedings of ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), La Jolla, CA, USA, pp. 289–296 (1992)
68. Ravindran, K., Bansal, V.: Delay compensation protocols for synchronization of multimedia data streams. IEEE Trans. Knowl. Data Eng. **4**(5), 574–589 (1993)
69. RFC-1889: Obsolete version—RTP: a transport protocol for real-time applications. http://tools.ietf.org/html/rfc1889/ (1996). Accessed 28 Apr 2017
70. RFC-3550: RTP: a transport protocol for real-time applications. http://tools.ietf.org/html/rfc3550/ (2003). Accessed 28 Apr 2017
71. RFC-3611: RTP control protocol extended reports (RTCP XR). http://tools.ietf.org/html/rfc3611/ (2003). Accessed 28 Apr 2017
72. RFC-7272: Inter-destination media synchronization (IDMS) using the RTP control protocol (RTCP). http://tools.ietf.org/html/rfc7272 (2014). Accessed 28 Apr 2017
73. Rothermel, K., Helbig, T.: An adaptive stream synchronization protocol. In: Proceedings of ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Durham, NH, USA, pp. 189–202 (1995)
74. Rouskas, G.N., Baldine, I.: Multicast routing with end-to-end delay and delay variation constraints. IEEE J. Sel. Areas in Commun. **15**(3), 346–356 (1997)
75. Shi, S.Y., Turner, J.S., Waldvogel, M.: Dimensioning server access bandwidth and multicast routing in overlay networks. In: Proceedings of ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Danfords on the Sound, NY, USA, pp. 83–92 (2001)
76. Steinmetz, R.: Analyse von synchronisation mechanismen mit anwendung im multimediabereich. In: Proceedings of GI ITG Workshop Sprachen und System zur Parallelverarbeitung, Arnoldshain, Germany, pp. 39–47 (1990)
77. Steinmetz, R.: Human perception of jitter and media synchronation. IEEE J. Sel. Areas Commun. **14**(1), 61–72 (1996)
78. Steinmetz, R., Nahrstedt, K.: Multimedia Computing, Communications and Applications. Prentice Hall (2015)
79. Stockham, T.: A/D and D/A converters: their effect on digital audio fidelity. IEEE Digital Signal Process. 55–66 (1972)
80. Tov, S.Y.: Happy 10th birthday, VoIP, The Marker. http://archive.li/TqIrI (2005). Accessed 28 Apr 2017
81. Vaishnavi, I., Cesar, P., Bulterman, D., Friedrich, O., Gunkel, S., Geerts, D.: From IPTV to synchronous shared experiences: challenges in design: distributed media synchronization. Signal Process. Image Commun. **26**, 370–377 (2011)
82. Wahl, T., Rothernel, K.: Representing time in multimedia systems. In: Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS), Boston, USA, pp. 538–543 (1994)
83. Woo, M., Qazi, N., Ghafoor, A.: A synchronization framework for communication of pre-orchestrated multimedia information. IEEE Netw. **1**(8), 52–61 (1994)
84. Yavatkar, R.: MCP: A protocol for coordination and temporal synchronization in collaborative applications. In: Proceedings of the IEEE International Conference Distributed Computing Systems (ICDCS), Yokohama, Japan, pp. 606–613 (1992)

85. Zhang, X., Liu, J., Li, B., shing Peter Yum T.: CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM), Miami, USA, pp. 2102–2111 (2005)
86. Zimmermann R, Liang K.: Spatialized audio streaming for networked virtual environments. In: Proceedings of ACM International Conference on Multimedia (MM), Vancouver, Canada, pp. 299–308 (2008)

Chapter 3
Theoretical Foundations: Formalized Temporal Models for Hyperlinked Multimedia Documents

Britta Meixner

Abstract Consistent linking and accurate synchronization of multimedia elements in hypervideos or multimedia documents are essential to provide a good quality of experience to viewers. Temporal models are needed to define relationships and constraints between multimedia elements and create an appealing presentation. However, no commonly used description language for temporal models exists. This makes existing temporal models harder to understand, compare, and transform from one to another temporal model. Using a formal description is more accurate than commonly used textual descriptions or figures of temporal models. This abstract representation makes it is easier to precisely define algorithms and constraints for delivery and buffering, as well as behavior of user and/or multimedia document. The use of a common formalism for all temporal models makes it possible to define synchronization constraints and media management. The same variables and terminology can then be used for describing algorithms that are applied to the documents, for example, to implement pre-fetching or download and cache management in order to increase the quality of experience for users. In this chapter, we give an overview of different existing temporal models for linked and temporally synchronized multimedia documents, like point-based, event-based, or interval-based temporal models. We analyze their common features and formally define their elementary components. We then give formal definitions for each temporal model covering essential features. These can then be used to computationally solve existing problems. We show this by defining basic functions that can be used in algorithms. We also show how user interaction and resulting video behavior can be precisely defined.

Keywords Multimedia document · Hypervideo · Temporal model
Formal definition

B. Meixner (✉)
CWI, Science Park 123, 1098 XG Amsterdam, Netherlands
e-mail: britta.meixner@cwi.nl

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_3

3.1 Introduction

Recent Web technologies like HTML5 allow the creation of appealing and interactive presentations consisting of various linked multimedia elements like images, videos, audio, and text. Users can interact and explore contents, find additional information about topics, and maybe even add their own contents. With growing Internet bandwidths and fast end user devices like smartphones and tablets, even synchronized and interactive multiscreen presentations are possible. Users can, for example, watch sports broadcasts on TV and, at the same time, follow their favorite athlete on a smartphone. They may get statistics and current standings in an object-based way that they can enable and disable on one of the screens. However, while it is technically possible to create hyperlinked single-device multimedia presentations or synchronized multidevice services, they are not popular.

The major challenge in this area is to ensure a good quality of experience by avoiding stalling event and synchronization issues between devices during playback. Work trying to optimize the user quality of experience by pre-fetching or adapting multimedia elements in hyperlinked or synchronized environments is needed in the future. This chapter helps researchers trying to create and optimize algorithms for temporal synchronization in multimedia presentations by providing models which can be used for computation. This approach was, for example, used for video centered hypervideos during the standardization process of HTML5. While a large part of the standard was already implemented in the commonly used browsers, the standard definitions and browser implementation constantly changed making it impossible to implement and test the developed algorithms. Using the formal temporal model and definitions enabled us to describe our algorithms in a platform-independent way and implement them in a test framework as described in [38]. After the HTML5 standard was finished and implemented in browsers, the findings were transformed to match the constraints given by the standards and could be easily reimplemented in HTML5 and JavaScript as shown in [35].

Giving formal definitions for the existing temporal models, algorithms can be defined using them. In order to explain basic parts and concepts of a multimedia document, we give and explain an example of a hypervideo. Figure 3.1 shows an exemplary structure of a hypervideo representing a tour through the ground floor of a house. Navigation between the rooms is realized via user-selection events. The lower left part of the image depicts the layout of the ground floor. The video scenes shown in the scene graph in the rest of the figure are filmed paths through rooms of the house, from one door to another. Viewers are asked where they want to go at certain points and are able to choose their own unique way through the house. The structure of the scenes defines a scene graph. The 16 scenes of the video are represented as labeled rectangles. The diamond symbolizes a fork in the flow where the viewer can choose a scene. Possible targets of a scene are other scenes, a fork, or the end of the video. The decision, which path is followed, depends on the click of the appropriate button. Here, the scene graph has a source (yellow triangle) and a sink (red circle). It is directed, weighted, and possibly cyclic.

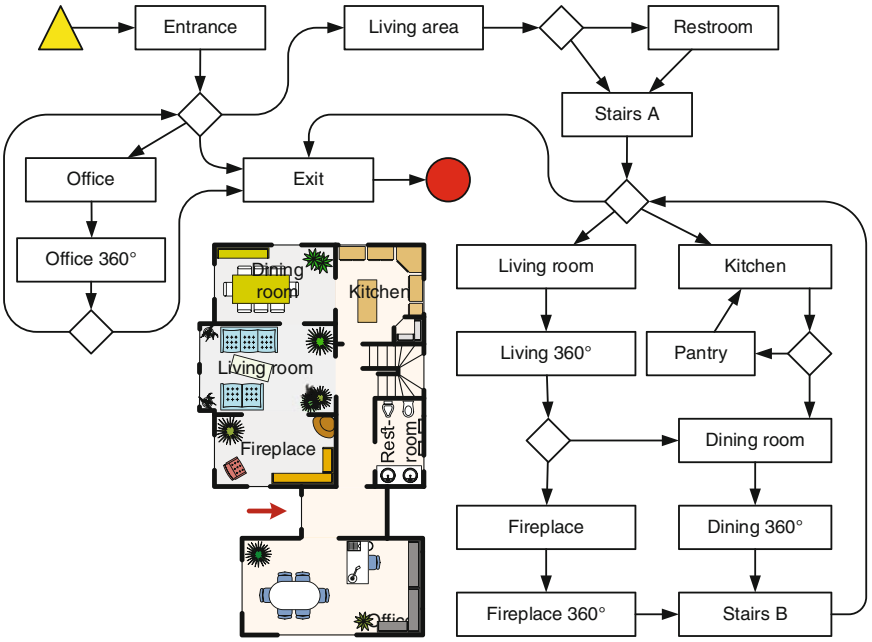


Fig. 3.1 Scene graph of a tour through the ground floor of a house with six rooms

Individual scenes are annotated with detailed images of furniture or flooring adding parallel multimedia elements to the main video. Doing this, a plain video can be enriched with additional information that may not be relevant to all viewers. The viewer may interact and get more information if desired. Text annotations (for example, showing prices and contact information), images (for example, showing alternative floor tiles), and audio files (for example, playing background music) describe room specifics or items shown in a scene that reach beyond the information provided by the video. Images provide detailed views of objects in the video. Figure 3.2 shows the time spans for displaying annotations of the entrance scene in a detailed view. This scene consists of a video with 710 frames (solid green boxes). During the playback of the video, four annotations are shown and hidden as exemplified by the diamond patterned boxes.

However, mixing different types of media and displaying them in parallel as done in the previous example may lead to QoE issues when, for example, videos are not properly synchronized or long buffering times after user interactions appear. Models help to define synchronization constraints and to create and specify algorithms for pre-fetching, download and buffer management. These algorithms can be pre-evaluated in simulations in test beds. A transformation of successfully pre-evaluated algorithms into real-world implementations helps solving synchronization and buffering issues, especially when underlying technologies and standards change.

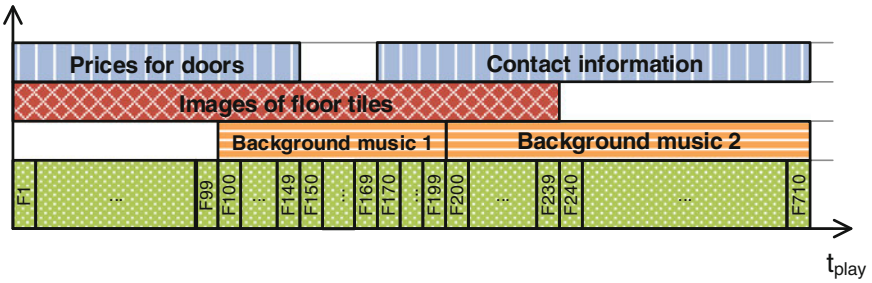


Fig. 3.2 Schedule for six annotations of one scene in detail

In this chapter, we first introduce fundamentals and terms in Sect. 3.2. We then provide universal definitions and descriptions for different temporal models (point-based, event-based, and interval-based) as described in literature in Sect. 3.3. We then give definitions of basic elements of multimedia documents in Sect. 3.4. After that, we provide formals descriptions for each temporal model in Sect. 3.5. Exemplary applications of definitions and models can be found in Sect. 3.6. This chapter ends with a conclusion in Sect. 3.7.

3.2 Fundamentals and Terms

Depending on used media, temporal model, and allowed interaction, we are talking about different forms of multimedia documents, depending if they are hyperlinked and/or synchronized. These can be described by the following terms (definitions beyond the below mentioned can be found in [39]):

- **multimedia element**: A multimedia element is an image, a video, an audio, a text, or any other type of audiovisual medium. It is the atomic object of any multimedia document.
- **annotation**: An annotation is additional information displayed with a main medium. It consists of an anchor attaching it to the main medium and a body. The body of an annotation is a multimedia element that can be shown in a player [39].
- **static multimedia element**: Static multimedia elements are time independent and always show the same content, like images and/or text.
- **continuous multimedia element**: Continuous multimedia elements are time dependent showing/playing different contents over time, like videos or audios.
- **hyperlinked media**: Hyperlinked media are multimedia elements which are linked with each other by hyperlinks (as known from hypertext). Static media may be clickable or have clickable areas. Continuous media may provide links depending on the media time.

- **media synchronization**: Synchronization of multimedia elements requires mechanisms to prepare the media for display (i.e., pre-fetch, buffering, rendering) and to ensure that timing constraints are met.
- **multimedia document**: A multimedia document is a *self-contained* presentation of linked and synchronized multimedia elements which allows user interaction and navigation. Usually, it is about a certain topic.

Many subcategories of multimedia documents providing more or less interactive features exist. They may focus on one main medium like video (called "video centered") or allow mixing different media files. Regardless of the differences, for each form of multimedia document, a description of some sort is necessary to define possible interactions as well as temporal and spatial relationships between multimedia elements. For example, a video description may specify sequences of scenes and the point in time at which the viewer can interact with the video, as well as the points in time where annotations are displayed or hidden. In addition, the description defines relationships and outlines their structure. Two multimedia elements may have a sequential, a conditional, or a parallel relationship, which needs to be defined in order to create a multimedia document. Describing the relationships and links between multimedia elements creates a structure which may be a linear, a tree, or a graph.

Depending on the desired output and use case of the multimedia document, different temporal models may be used. They are different in the way in which temporal relationships and control are specified. They may be using points, events, or intervals which will be explained hereafter:

- **point**: A (time) point is a precise moment in time [41]. It is synchronized with a clock.
- **event**: An event is something that happens or takes place [41]. It may be triggered by a clock or by a user interaction.
- **interval**: A (time) interval is the time between start and end of a time span.

Distinguishing among hypermedia, passive multimedia (presentations), and active multimedia (presentations) is useful for the definition of temporal models, because it limits their scope. The terms can be defined as follows:

- **hypermedia**: Hypermedia is an extension to hypertext providing multimedia facilities, such as those handling sound and video [41]. Keeping the hyperlink structure from hypertext, multimedia elements of different types are added.
- **multimedia**: Multimedia uses a variety of artistic or communicative media that are presented in one presentation [41].
  - **passive multimedia**: Passive multimedia presentations are started and then watched with little to no interaction. Available forms of interaction are starting, pausing, and stopping the presentation.
  - **active multimedia**: Active multimedia presentations allow more interaction compared to passive multimedia presentations. They may have hyperlinks or other interactive control elements.

According to Hirzalla et al., hypermedia "implies store-and-forward techniques where user actions, typically mouse-selections on hotspots, cause the system to retrieve a new 'page' of data which could be an image, text, video etc. There are usually no temporal relationships between media. Passive multimedia implies a fully synchronized document that 'plays itself back' in time, synchronizing all multimedia elements together. Active multimedia implies that there are hypermedia-type choices presented to users during the playback of a multimedia document which allow the user's interaction to 'drive' the playback" [22].

3.3 Temporal Models

Hereafter, we describe and analyze related work for point-based, event-based, interval-based, and other less common temporal models and show common features of the temporal models that will be formalized in Sect. 3.5. As an example, we use the scene from Fig. 3.2 in Sect. 3.1. An overview of all variables is given in Table 3.4 in the Appendix.

3.3.1 Point-Based Temporal Models

Using point-based temporal models, each event is triggered by its point in time. The points in time are ordered on a timeline [15]. The points in time form a total order [15]. "For any two points in time [,] one of the relationships before (<), after (>), or equals (=) holds" [7]. "An example of the point-based approach is [a] timeline, in which multimedia elements are placed on several time axes called tracks, one per each media type. All events such as the beginning or the end of a segment are totally ordered on the time line" [15]. Figure 3.3 shows an exemplary presentation with a point-based temporal model. The presentation consists out of six multimedia elements. Three of them start at t_0, one starts at t_7, one starts at t_{13}, and the last one starts at t_{16}. One element is displayed over the whole presentation while the others start later or end earlier. The presentation ends at t_{29}.

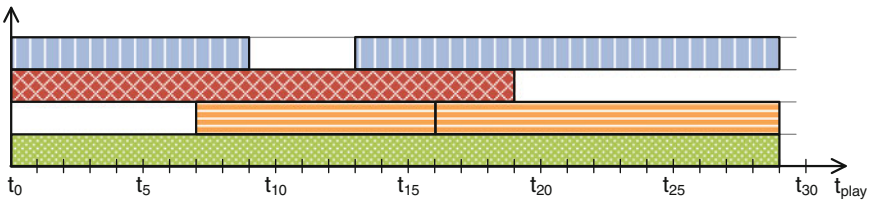


Fig. 3.3 Exemplary timeline of a presentation with six multimedia elements and time points for starting/stopping their display

Other approaches that incorporate this temporal model are point nets [9]. This temporal model requires media segments with known durations, and it is not possible to use it for unknown durations [15]. Wahl et al. call the previously described relationships before (<), after (>), or equals (=) "basic point relations (basic PRs)" [49]. They also differentiate between relations in the past and relations between future events which might be indefinite and it might not be known which one out of two possible relations will become true. "Typically, indefinite relations are represented as disjunctions of basic PRs. Since there are three basic PRs, $2^3 = 8$ disjunctions exist each representing an indefinite relation. Any of the eight indefinite relations has an associated symbolic notation. The eight indefinite relations are as follows: $\oslash, \leq, <, =, >, \geq, \neq, ?$, where ? is the full set of basic PRs $<, =, >$, $\oslash$ is the empty set $\{\}$" [49].

Blakowski and Steinmetz call the timeline "axis". They differentiate between "synchronization based on a global timer" and "synchronization based on virtual axes". They describe the use of different clocks depending on the underlying content. They point out that the temporal model using the global timer is easy to understand, it supports hierarchies, it is easy to maintain, it provides a good abstraction for multimedia elements, and the integration of time-dependent objects is easy. Disadvantages are that objects need a previously defined duration and additional effort is necessary to implement QoS. The temporal model using virtual axes allows in addition to add specifications according to a certain problem space. Time-independent multimedia elements can be integrated, and interactive objects are possible. However, the additional axes may lead to complex specifications and the mapping of the axes during runtime may be complex and time-consuming [5].

Different works use the point-based temporal model. "Prior to 1993, few multimedia systems had reached the 'document formatter' stage. They typically required authors to create temporal layouts manually by positioning media at absolute points on a document timeline [14, 34, 40, 43], a tedious and error-prone process" [10]. Further systems and standards using this temporal model are, for example: the Firefly multimedia document system [9] and the works by [14, 20].

3.3.2 Event-Based Temporal Models

In event-based temporal models, synchronization events trigger presentation actions. Synchronization events aim at a target (like a multimedia element or a point in time) and contain an associated action (like show/hide a medium) that must be triggered at a given time [44]. Typical actions are starting, stopping, and preparing a presentation. Events can be external (e.g., generated by a timer) or internal (e.g., generated by time-dependent multimedia element) [5]. Figure 3.4 shows an exemplary presentation with an event-based temporal model. The presentation consists out of six multimedia elements. Three of them start at the beginning, triggered by e_0, e_1, and e_2, and another one is started at event e_3. At e_4, the first element is stopped/hidden, and then another one is started at e_5. Another element is hidden at e_6, immediately after-

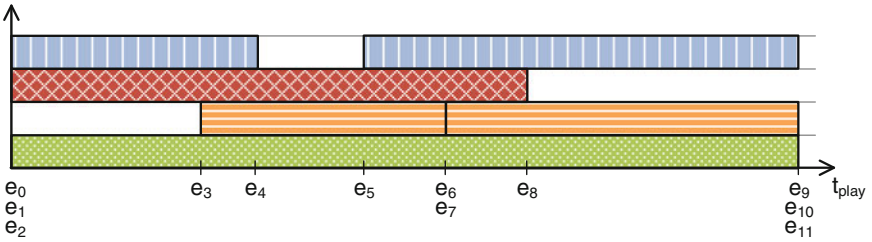


Fig. 3.4 Exemplary timeline of a presentation with six multimedia elements and starting and stopping events

ward, an element is started at e_7. The events e_8, e_9, e_{10}, and e_{11} are stopping/hiding multimedia elements which finally finish the presentation.

According to Blakowski and Steinmetz, event-based temporal models are easily extensible with new events, they are easy to integrate interactive objects, and they are flexible, because any event can be specified [5]. Disadvantages are that they are difficult to handle, have a complex specification, are hard to maintain, and time-dependent objects can only be integrated by using additional timers [5]. Boll et al. describe event-based temporal models as follows: "In an event-based model of time, events determine the temporal course of the presentation. An event is connected to actions and when an event occurs, e.g., a video reaches a certain point in time, the corresponding actions, typically start and stop of the presentation of other media elements, is carried out" [7].

Works using the event-based temporal model are, for example, HyTime [26], HyperODA [3], MHEG-5 [25], or the SIVA Suite [36–38]. "Events are defined in HyTime as presentations of media objects along with the playout specifications and finite coordinate system (FCS) coordinates. HyperODA events happen instantaneously and mainly correspond to start and end of media objects or timers. All these approaches suffer from poor semantics conveyed by the events. Moreover, they don't provide any scheme for composition and consumption architectures" [48]. The temporal model of the SIVA Suite has several layers where events can occur. The hypervideo is video-based and divided into smaller units called scenes. A scene has one single main video and may have several annotations which can either be triggered by time or by a user interaction. Interaction with the annotations is also possible. In a scene, showing and hiding of media or interactive elements that can show annotations is triggered when the main video reaches a certain point in time. Scenes are linked with each other, and the follow-up scene is triggered by a user-selection event.

3.3.3 Interval-Based Temporal Models

Interval-based temporal models are based on intervals which are defined by two points on a timeline. "Interval-based models consider elementary media entities as

time intervals ordered according to some relations" [48]. Depending on the complexity and possible interactions with the temporal model, one can differentiate between basic and enhanced interval-based temporal models. However, enhanced interval-based temporal models are always extensions of basic interval-based temporal models.

3.3.3.1 Basic Interval-Based Temporal Models

Following the definitions of Allen ("Assuming a model consisting of a fully ordered set of points of time, an interval is an ordered pair of points with the first point less than the second" [2]) and Wahl and Rothermehl ("As any interval can be characterized by its beginning and end, any basic [interval relations] can be represented by a conjunction of [point relations] on its margins" [49]), 13 basic relations between two intervals can be defined [2]:

- *X equals Y* (meaning start and end point of both intervals are the same, parallel intervals);
- *X before Y* and *Y before X* (meaning one interval has no overlap with the other);
- *X meets Y* and *Y meets X* (a sequence of intervals);
- *X overlaps Y* and *Y overlaps X* (overlap of the intervals but also parts where only one interval is played);
- *X during Y* and *Y during X* (first interval shorter than second, all of first is parallel with second);
- *X starts Y* and *Y starts X* (both intervals have the same start point and are parallel, but the first is shorter than the second); and
- *X finishes Y* and *Y finishes X* (both intervals have the same end point and are parallel, but the first is shorter than the second).

Figure 3.5 shows an exemplary presentation with an interval-based temporal model using Allen's temporal relationships. The presentation consists out of six multimedia elements. Three of them start at the beginning, (*M*1 *starts M*3, *M*2 *starts M*3), three end at the same time (*M*5 *finishes M*3, *M*6 *finishes M*3), and two form a sequence of intervals (*M*4 *meets M*6).

Little and Ghafoor extend this temporal model to n-ary temporal relations [30]. Wahl and Rothermehl propose an enhanced interval-based temporal model with 29

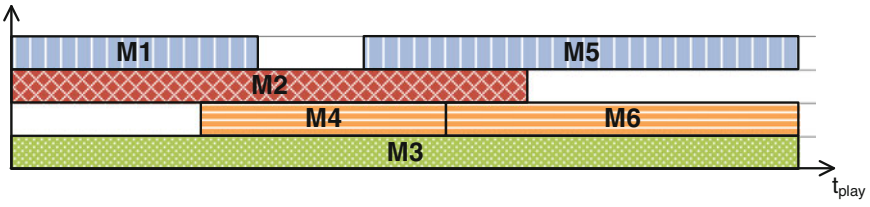


Fig. 3.5 Exemplary timeline of a presentation with six multimedia elements

interval relations in [49], which extends the 13 basic relations of [2]. Advantages of the interval-based temporal model are that logical objects can be kept, there is a good abstraction for media content, it is easy to integrate time-independent and/or interactive objects, and the specification of indeterministic temporal relations is supported. Disadvantages are a complex specification, the necessity of additional specifications for QoS that it is only possible to specify relations between whole multimedia elements, not subparts of multimedia elements, and that resolving indeterminism at runtime may lead to inconsistencies [5].

Interaction with interval-based temporal models is discussed by Wahl et al. [50] and Little and Ghafoor [30]. Wahl et al. define two different temporal interaction forms: context interaction (start, stop, and selection (jump to new document)) and speed interaction (for example, pause/continue, faster, slower, forward, backward, reverse, and set speed). These interactions always affect the whole presentation, not single elements [50]. Little and Ghafoor state that "uncertainty created by random user interaction is an additional complexity in managing time in multimedia information systems" [30].

Works using the interval-based temporal model are in addition to the already mentioned the following: The work of Fujikawa et al. describes "Harmony" which uses a combination of events and intervals [19]. Euzenat et al. propose a semantic framework for multimedia adaptation for heterogeneous devices resulting in various constraints like different display sizes and available bandwidths. Thereby, a temporal "model of a multimedia document is a potential execution of this document and a context defines a particular class of models" [17]. Little and Ghafoor use a Petri Net definition where intervals are represented by places and relations by transitions to deal with the interval-based temporal model [32]. A later work of Little and Ghafoor proposes a solution to store a temporal model in a database [30]. Wahl and Rothermehl analyze path expressions [11, 23] which include three operators to represent temporal relations: sequence, parallel-first, and parallel-last deal with interval-based temporal models. They also discuss MHEG [25, 29] which uses two temporal operators sequential and parallel similarly to the path expression model [49].

3.3.3.2 Enhanced Interval-Based Temporal Models

There are several issues with Allen's relations [2], according to Duda and Keramane [15]. The relations were designed for intervals with fixed durations. Changes in durations may transform one relation into another. "Another problem with the Allen relations is their descriptive character, they allow expression of an existing, a posteriori arrangement of intervals, but they do not express any causal or functional relation between intervals" [15]. This makes Allen's relations useful for characterizing existing, instantiated presentations where all start and termination points of media segments are known. The third problem is, that the relations may lead to inconsistent specifications that may occur in a multimedia presentation. The detection of the third problem requires algorithms of complexity [$O(N^2)$], where N is the number of intervals [2]. To deal with these issues, enhanced interval-based

temporal models use the 13 basic relations of [2], or similar definitions but extend them with, for example, mechanisms to deal with unknown durations of intervals, semantics on the intervals, or substitutions of elements in certain contexts.

Benbernou et al. use techniques called "Augmentation and Substitution" [4]. They find alternatives for multimedia elements that are "semantically closed" using definitions of semantic constraints between media elements. They also substitute "unwanted media" using alternatives which take the spatiotemporal coherence of the presentation into account [4]. Boll et al. introduce intervals of unknown durations for user interaction in multimedia presentations. "With the Interval Expressions [15] we find a temporal model for multimedia presentations on the level of intervals with a set of temporal operators to relate time intervals which possibly have an unknown duration, that also overcomes the problem of temporal inconsistencies by construction. The Interval Expressions form the basis of the underlying temporal model of the ZyX data model" [8]. SMIL also uses an enhanced interval-based temporal model.

3.3.4 Other

Other temporal models are mentioned in the literature, but are less important and significant compared to the already mentioned temporal models. Furthermore, it may be possible to categorize specific implementations in one of the other categories. The two major subcategories are script-based and tree-based temporal models. According to Blakowski and Steinmetz, scripts have the following advantages: they have a good support for hierarchies, logical objects can be kept, it is easy to integrate time-independent and interactive objects, these are easily extensible with new constructs and flexible due to their programmability. However, they are not easy to handle, can have complex specifications, additional timers are necessary as well as constructs to implement QoS. Fiume et al. use temporal scripting languages [18].

Kim et al. use temporal relation trees [27]. Hirzalla et al. propose a timeline-tree temporal model which introduces choice elements into the timelines known from point-based temporal models [22]. Courtiat and De Oliveira use presentation and constraint objects in a hierarchical composition which is then translated into a complete RT-LOTOS formal specification (extension of [16]) [13].

According to Blakowski and Steinmetz, control flow-based specifications may use basic hierarchical descriptions [1, 45] (serial/parallel), reference points [6, 47] or timed Petri Nets [31, 33]. For further descriptions, see [5].

3.3.5 Summary

In this section, we showed differences for existing temporal models. Point-based temporal models are the easiest to define, but are based on one timeline which requires fixed start and end points for multimedia elements. They allow only basic

VCR actions for the overall presentation. Event-based temporal models are more advanced, allowing different types of events and interactions. They are more flexible than point-based temporal models. However, no commonly valid definition of events or the temporal model itself could be found in existing literature. The interval-based temporal models are the most discussed and researched in related work. All of the works are based on the basic interval relationships defined by Allen in 1983 [2]. However, due to their possible nondeterministic presentation, several methods are proposed to synchronize the media for playback. Adding interactivity to the temporal models makes them more complicated, especially with regard to synchronization. If interactivity is possible on single multimedia elements, it gets even more complicated. Table 3.1 shows an overview of the different temporal models with regard to possible relationships between objects/media and events, interactivity, and timing, which are the most distinctive differences between the temporal models. Depending on the given task, one or the other temporal model may have its advantages or disadvantages. Table 3.1 may help to pick the right temporal model given relationships between media and events, timing constraints, or requirements regarding interactivity.

Table 3.1 Comparison of temporal models

Temporal model	Relationships between objects/media	Relationships between events	Interactivity	Timing
Point-based	Sequential, parallel	Before (<), after (>), or equals (=)	Start/stop/ pause whole presentation	Display/hiding and start/stop of media are triggered by points in time on a timeline, total order for points on timeline
Event-based	Sequential, parallel, conditional, linked	Events are not clearly defined throughout literature	Depending on model	Events are triggered by other events
Basic interval-based	13 relationships defined by Allen [2] or 29 interval relations in [49]		VCR actions for whole presentation	Based on relations, not necessarily deterministic
Enhanced interval-based	13 relationships defined by Allen [2] or 29 interval relations in [49]	Before (<), after (>), or equals (=) for start and end points of intervals	Interactions on multimedia elements	Based on relations, not necessarily deterministic

3.4 Definition of Basic Multimedia Elements

As shown in the previous sections, existing temporal models are capable of describing relationships between elements, timing, and interactivity. However, no commonly used mechanism for describing them is available. The authors of each paper use their own formalisms making it hard to compare the temporal models or transform one of the temporal models into another. Hereafter, we give formal definitions of all basic elements that may be part of a multimedia document, namely, static (like text, images) and continuous (like audio, video) multimedia elements. Static and continuous media files are fundamental elements of a multimedia presentation and appear in each of the temporal models in some way.

A static annotation is a static multimedia element which contains a multimedia element α (see Definition 1) and has a priority Λ (see Definition 2). The multimedia element may be any type of additional content like text, image, video, or audio files and is defined as a sequence of bits. These usually have to be transmitted over a network. For simplification purposes, the content of a continuous multimedia element may be handled as a single block whether it is a continuous or a static medium. More precisely, a static multimedia element a is a pair of the multimedia element α and a priority Λ (see Definition 3). We also define the set of static multimedia elements $\mathcal{A}_D$ (see Definition 4) of a multimedia document $\mathcal{D}$.

Definition 1 (*Content of a Static Multimedia Element* α) The content of a static multimedia element α is an n-tuple of bits representing a multimedia element; $\alpha := (\chi_1, \ldots, \chi_n) \in \{0,1\}^n, n \in \mathbb{N}^+$.

Definition 2 (*Priority of a Static Multimedia Element* Λ) The priority of a static multimedia element is Λ, $\Lambda \in \mathbb{N}^+$. The higher Λ is, the lower is the priority of the static multimedia element.

Definition 3 (*Static Multimedia Element* a) A static multimedia element $a := (\alpha, \Lambda)$, $\Lambda \in \mathbb{N}^+$ is a pair of the content of the static multimedia element α and the priority of the static multimedia element Λ.

Definition 4 (*Set of Static Multimedia Elements* $\mathcal{A}_D$) $\mathcal{A}_D$ is a finite set of static multimedia elements of the multimedia document $\mathcal{D}$;
$\mathcal{A}_D := \{a_i | a_i \text{ is a static multimedia element.}\}$

A continuous multimedia element consists of frames (video) or samples (audio). These consist of a sequence of bits as defined in Definition 5.

Definition 5 (*Frame/Sample* f) A frame/sample f is an n-tuple of bits representing an image or audio sample; $f := (\chi_1, \ldots, \chi_n) \in \{0,1\}^n, n \in \mathbb{N}^+$.

We define the set of frames/samples $\mathcal{F}_D$ (see Definition 6) of a multimedia document as well as an ordered n-tuple of frames/samples F. Similar to the set of static multimedia elements $\mathcal{A}_D$, we also define a set of continuous multimedia elements $\mathcal{C}_D$.

Definition 6 (*Set of Frames* $\mathcal{F}_D$) $\mathcal{F}_D$ is a finite set of frames/samples of the multimedia document $\mathcal{D}$.

Definition 7 (*Continuous Multimedia Element F*) A continuous multimedia element $F := (f_\sigma, f_1, \ldots, f_n, f_\epsilon)$, $n \in \mathbb{N}^+, f_i \in \mathcal{F}_D, 1 \leq i \leq n$ is an n-tuple of frames/samples f with start frame/sample f_σ and end frame/sample f_ϵ.

Definition 8 (*Set of Continuous Multimedia Elements* C_D) C_D is a finite set of continuous multimedia elements of the multimedia document $\mathcal{D}$;
$C_D := \{F_i | F_i$ is a continuous multimedia element.$\}$

3.5 Formalized Temporal Models

Now we use the definitions of basic multimedia elements from Sect. 3.4 to describe more complex structures in point-based, event-based, and interval-based temporal models.

3.5.1 Point-Based Temporal Model

With the definitions in Sect. 3.4, all multimedia elements are already defined. Additional definitions are necessary for the timeline and events which show or hide a multimedia element. Interaction with single multimedia elements is usually not possible in point-based temporal models and does not need to be defined accordingly.

3.5.1.1 Additional Definitions

Points in time t (see Definition 9) as well as time intervals T (see Definition 10) need to be defined to control when events are applied to multimedia elements. The timeline is given as an abstract external clock.

Definition 9 (*Point in Time t*) The point in time of a presentation is t, $t \in \mathbb{N}$.

Definition 10 (*Time Interval T*) A time interval T is a n-tuple of discrete points in time t_i; $T := (t_0, \ldots, t_n)$, $n \in \mathbb{N}^+, t_i \in \mathbb{N}, 1 \leq i \leq n$.

In order to define what happens to a multimedia element at a certain point in time, an executable action *ea* (see Definition 11) for a multimedia element e is defined. Possible executable actions vary between static and continuous media. Static media can be shown and hidden. Continuous media can be started, paused, and stopped. Sets for executable actions for static multimedia elements $\mathcal{EA}_a$, dynamic multimedia elements $\mathcal{EA}_F$ and all executable actions $\mathcal{EA}$ are defined in Definitions 12, 13, and 14.

Definition 11 (*Executable Action ea*) An executable action *ea* is an action that is applied to a multimedia element *e*, $e \in \mathcal{A}_D \cup C_D$.

Definition 12 (*Set of Executable Actions for Static Multimedia Elements* $\mathcal{EA}_a$) For static multimedia elements *a*, the finite set of actions is $\mathcal{EA}_a := \{show_a, hide_a\}$.

Definition 13 (*Set of Executable Actions for Dynamic Multimedia Elements* $\mathcal{EA}_F$) For continuous multimedia elements, the finite set of actions is $\mathcal{EA}_F := \{start_F, pause_F, stop_F\}$.

Definition 14 (*Set of Executable Actions* $\mathcal{EA}$) The finite set of executable actions $\mathcal{EA}$ of a presentation is defined as $\mathcal{EA} := \mathcal{EA}_a \cup \mathcal{EA}_F$.

After defining executable actions, these need to be linked to multimedia elements in order to indicate what should happen with the multimedia element. Therefore, we define a pair *ee* called element event in Definition 15. The set of element events $\mathcal{EE}$ is defined in Definition 16.

Definition 15 (*Element Event ee*) An element event *ee* is a pair containing an element *e* and an executable action *ea*; $ee_j := (e_j, ea_n),\ e_j \in \mathcal{A}_D \cup \mathcal{F}_D,\ ea_n \in \mathcal{EA}$.

Definition 16 (*Set of Element Events* $\mathcal{EE}$) $\mathcal{EE}$ is a finite set of element events *ee*.

At each point in time, one or more events may occur. For that reason, we link a set of element events to a point in time. The resulting pair *ep* is called event point (see Definition 17), the set of event points $\mathcal{EP}$ is defined in Definition 18.

Definition 17 (*Event Point ep*) An event point *ep* is a pair containing a point in time t_i, and a set of element events EE_i; $ep_i := (t_i, EE_i),\ t_i \in T,\ EE_i \subset \mathcal{EE} \cup \emptyset$.

Definition 18 (*Set of Event Points* $\mathcal{EP}$) $\mathcal{EP}$ is a finite set of event points *ep*.

The whole presentation can be described as an n-tuple of event points *ep* and is called a timeline *tl* (see Definition 19).

Definition 19 (*Timeline tl*) A timeline *tl* is an n-tuple of event points; $tl := (ep_0, \ldots, ep_n),\ n \in \mathbb{N}^+, ep_i \in \mathcal{EP},\ 1 \leq i \leq n$.

3.5.1.2 Example

Definition 19 is now illustrated with a small example as shown in Fig. 3.6 and specified in Eq. 3.1. This timeline has 30 time points. The presentation consists of three continuous multimedia elements (two videos (F_1 and F_2) and one audio (F_3)) and four static multimedia elements (one text (a_1) and three images (a_2, a_3, and a_4)). The text is shown over the whole time. The images are shown one after the other with a break in-between. First, video F_1 is shown, then video F_2, which is muted and audio F_3 is used to replace its sound.

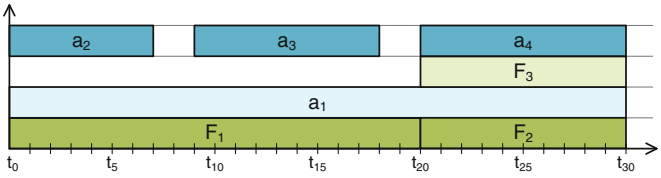


Fig. 3.6 Exemplary timeline of a presentation with continuous and static multimedia elements

The **set of continuous media** from Eq. 3.1 is set to $C_D = \{F_1, F_2, F_3\}$ and contains two videos, F_1 and F_2, and one audio F_3. The **set of static media** from Eq. 3.1 contains four elements. It is defined as $\mathcal{A}_D = \{a_1, a_2, a_3, a_4\}$ and contains one text a_1 and three images a_2, a_3, and a_4.

$$
\begin{aligned}
tl = &((t_0, \{(F_1, start_F), (a_1, show_a), (a_2, show_a)\}),\\
&(t_1, \emptyset),\\
&\dots,\\
&(t_6, \emptyset),\\
&(t_7, \{(a_2, hide_a)\}),\\
&(t_8, \emptyset),\\
&(t_9, \{(a_3, show_a)\}),\\
&(t_{10}, \emptyset),\\
&\dots,\\
&(t_{17}, \emptyset),\\
&(t_{18}, \{(a_3, hide_a)\}),\\
&(t_{19}, \emptyset),\\
&(t_{20}, \{(F_1, stop_F), (F_2, start_F), (F_3, start_F), (a_4, show_a)\}),\\
&(t_{21}, \emptyset),\\
&\dots,\\
&(t_{29}, \emptyset),\\
&(t_{30}, \{(F_2, stop_F), (F_3, stop_F), (a_1, hide_a), (a_4, hide_a)\})).
\end{aligned}
\tag{3.1}
$$

3.5.2 *Event-Based Temporal Model*

While no clear definition and common understanding of the event-based temporal model exists, we show one form of hypervideo where this temporal model is used, the so-called Annotated Interactive Nonlinear Video $\mathcal{V}$ [38]. This video centered type of hypervideo always has a video as main medium, which also defines the clock during

playback. All other multimedia elements are called annotations and are treated as single block of data, no matter if the annotation is a static or continuous medium. In addition to video scenes and annotations, control information is defined which is needed during playback to select the successor scene at a fork. With the definitions in Sect. 3.4, we can define videos scenes. These can then be extended and used to formulate the definition of annotated interactive nonlinear video $\mathcal{V}$ as a deterministic finite state machine.

3.5.2.1 Additional Definitions

An n-tuple containing pairs of a frame and a set of annotations is representing a scene p (see Definition 20). The set of annotations attached to the frame indicates that all annotations in the set are displayed with the frame. An annotated interactive nonlinear video furthermore has a start scene p_σ and an end scene p_ϵ with just one frame and an empty set of annotations (see Definitions 21 and 22). All scenes can be combined to a set of scenes $\mathcal{P}_\mathcal{V}$ (see Definition 23).

Definition 20 (*Scene p*) A scene p is an n-tuple of pairs each containing a frame and a set of annotations which are displayed with the frame; $p_x := ((f_{x,1}, A_{x,1}), \ldots, (f_{x,n}, A_{x,n}))$, $x, n \in \mathbb{N}^+, f_{x,i} \in \mathcal{F}_\mathcal{V}, A_{x,i} \subseteq \mathcal{A}_\mathcal{V}, 1 \leq i \leq n$.

Definition 21 (*Start Scene p_σ*) The start scene p_σ is a 1-tuple containing a pair representing a single frame without an annotation; $p_\sigma := ((f_{\sigma,1}, \{\}))$.

Definition 22 (*End Scene p_ϵ*) The end scene p_ϵ is a 1-tuple containing a pair representing a single frame without an annotation; $p_\epsilon := ((f_{\epsilon,1}, \{\}))$.

Definition 23 (*Set of Scenes $\mathcal{P}_\mathcal{V}$*) The set of scenes $\mathcal{P}_\mathcal{V}$ of the annotated interactive nonlinear video is defined as $\mathcal{P}_\mathcal{V} := \{p_\sigma, p_1, \ldots, p_x, p_\epsilon\}, x \in \mathbb{N}^+$.

The whole annotated interactive nonlinear videos can be described with the elements and sets defined so far as a deterministic finite state machine $\mathcal{N}_\mathcal{V}$ (see [24, p. 14 et seq.]). The finite set of states is represented by the set of scenes $\mathcal{P}_\mathcal{V}$. The input symbols Σ are defined as a set of Boolean functions as defined in [21, p. 40]. The transition function is δ, it takes a scene as the current state and a button click as input and returns a scene as next state. The start state is p_σ, the set of end states only contains one element, $\{p_\epsilon\}$.

Definition 24 (*Annotated Interactive Nonlinear Video $\mathcal{V}$*) An annotated interactive nonlinear video $\mathcal{V}$ is defined as a deterministic finite state machine $\mathcal{N}_\mathcal{V} := (\mathcal{P}_\mathcal{V}, \Sigma, \delta, p_\sigma, \{p_\epsilon\})$ with $\Sigma := \{w_{i,j} | w_{i,j}$ is a button triggering the selection of a successor scene, $i \in \{1, \ldots, |\mathcal{P}_\mathcal{V}| - 2, \sigma\}, j \in \{1, \ldots, |\mathcal{P}_\mathcal{V}| - 2, \epsilon\}\}$ and $\delta : \mathcal{P}_\mathcal{V} \times \Sigma \rightarrow \mathcal{P}_\mathcal{V}$.
The following restrictions are applied: $\exists! k : \delta(p_\sigma, w_{\sigma,k}) \rightarrow p_k \wedge \nexists k : \delta(p_k, w_{k,\sigma}) \rightarrow p_\sigma \wedge \exists k : \delta(p_k, w_{k,\epsilon}) \rightarrow p_\epsilon \wedge \nexists k : \delta(p_\epsilon, w_{\epsilon,k}) \rightarrow p_k$.

The deterministic finite state machine $\mathcal{N}_{\mathcal{V}}$ defines possible successors of a scene and which buttons have to be clicked to access a designated successor scene. The transition $(p_m, w_{i,j}) \rightarrow p_n \in \delta$ implies that scene p_n is successor of scene p_m [38]. Applied restrictions define that there is exactly one transition from the start scene to the first scene ($\exists! k : \delta(p_\sigma, w_{\sigma,k}) \rightarrow p_k$), that once the video is started, the start scene cannot be reached ($\nexists k : \delta(p_k, w_{k,\sigma}) \rightarrow p_\sigma$), that there is at least one scene connected to the end scene ($k : \delta(p_k, w_{k,\epsilon}) \rightarrow p_\epsilon$), and once the end scene is reached, the video ends ($\nexists k : \delta(p_\epsilon, w_{\epsilon,k}) \rightarrow p_k$).

Restrictions need to be applied for the start and the end scene.

3.5.2.2 Example

Definition 24 will now be illustrated with a small example as shown in Fig. 3.7. This annotated interactive nonlinear video has six scenes, including start and end scene, and five annotations. The **set of scenes** is defined as $\mathcal{P}_{\mathcal{V}} = \{p_\sigma, p_1, p_2, p_3, p_4, p_\epsilon\}$. The different scenes can be described as follows:

- $p_\sigma = ((f_{\sigma,1}, \{\}))$,
- $p_1 = ((f_{1,1}, \{a_1\}), \ldots, (f_{1,750}, \{a_1\}), (f_{1,751}, \{\}), \ldots, (f_{1,1500}, \{\}))$,
- $p_2 = ((f_{2,1}, \{a_2\}), \ldots, (f_{2,500}, \{a_2\}), (f_{2,501}, \{a_3\}), \ldots, (f_{2,1500}, \{a_3\}))$,
- $p_3 = ((f_{3,1}, \{a_5\}), \ldots, (f_{3,1000}, \{a_5\}))$,

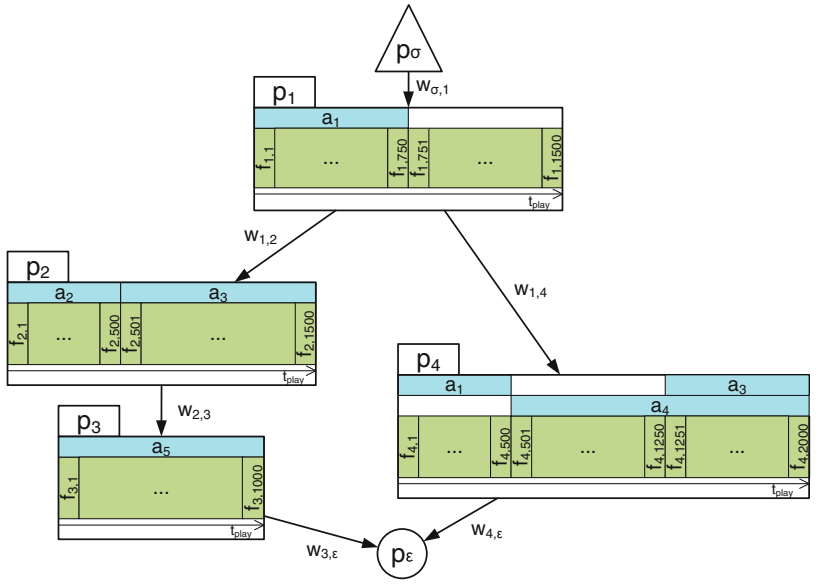


Fig. 3.7 Example of an annotated interactive nonlinear video with six scenes (including start and end scene) and five annotations

- $p_4 = ((f_{4,1}, \{a_1\}), \ldots, (f_{4,500}, \{a_1\}), (f_{4,501}, \{a_4\}), \ldots, (f_{4,1250}, \{a_4\}), (f_{4,1251}, \{a_3, a_4\}), \ldots, (f_{4,2000}, \{a_3, a_4\}))$, and
- $p_\epsilon = ((f_{\epsilon,1}, \{\}))$.

The **set of frames** is set to $\mathcal{F}_D = \{f_{\sigma,1}, f_{1,1}, \ldots, f_{1,1500}, f_{2,1}, \ldots, f_{2,1500}, f_{3,1}, \ldots, f_{3,1000}, f_{4,1}, \ldots, f_{4,2000}, f_{\epsilon,1}\}$ and contains 6002 frames which are divided up into the six scenes. The first and second scene, p_1 and p_2, each consist of 1500 frames, the third scene p_3 consists of 1000 frames and the fourth scene p_4 consists of 2000 frames. The **set of annotations** contains five elements. It is defined as $\mathcal{A}_D = \{a_1, a_2, a_3, a_4, a_5\}$.

The **transition function** δ defines where and under what conditions transitions from one scene to another are allowed. The transition $\delta(p_\sigma, w_{\sigma,1}) \rightarrow p_1$ sets the first scene of the video. Transitions $(p_3, w_{3,\epsilon}) \rightarrow p_\epsilon$ and $(p_4, w_{4,\epsilon}) \rightarrow p_\epsilon$ indicate two different last scenes of the video followed by the end scene. A linear transition is also defined from scene p_2 to p_3 with $\delta(p_2, w_{2,3}) \rightarrow p_3$. In these cases, the follow-up scenes start immediately after the predecessor scenes end. The remaining two transitions, $\delta(p_1, w_{1,2}) \rightarrow p_2$ and $\delta(p_1, w_{1,4}) \rightarrow p_4$, describe a selection panel at the end of scene p_1. The viewer in this example selects button $w_{1,2}$ or button $w_{1,4}$. Only one of the buttons/paths can be selected [38].

3.5.3 Interval-Based Temporal Model

Interval-based temporal models are based on intervals where multimedia elements are shown. Hereafter, we use Allen's 13 basic dual relationships [2] between individual intervals.

3.5.3.1 Additional Definitions

For the definition of relationships between intervals (see Definition 28), intervals have to be defined (see Definition 26). Both are then linked in interval relations as defined in Definition 29. The set of interval relations then defines the whole presentation (see Definition 30). While continuous media are already representing intervals, static media need a display duration to show them in interval-based temporal models. This is defined in Definition 25.

Definition 25 (*Display Duration A*) The display duration of a static multimedia element a is defined as a pair A_i consisting of a multimedia element a_i and its duration t_i; $A_i := (a_i, t_i),\ i \in \mathbb{N}^+, a_i \in \mathcal{A}_D, t_i \in \mathbb{N}^+$.

Definition 26 (*Interval I*) An interval I_e for a multimedia element e is defined as a pair consisting of the start point t_{s_e} and the end point t_{e_e} of the interval;

$$I_e := (t_{s_e}, t_{e_e}), \begin{cases} t_{s_e} - t_{e_e} = t_i & \text{if } e \in \mathcal{A}_D \\ t_{s_e} - t_{e_e} = dim(F) & \text{if } e \in \mathcal{F}_D \end{cases} \tag{3.2}$$

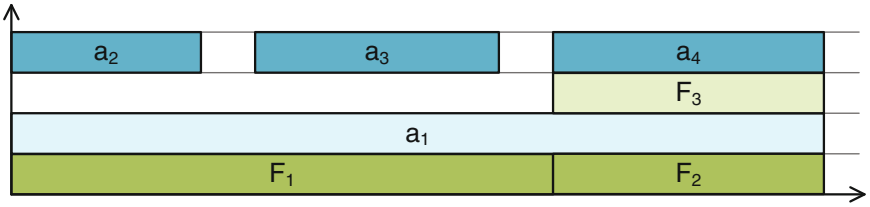


Fig. 3.8 Exemplary relations between intervals with continuous and static multimedia elements

Definition 27 (*Set of Intervals* $\mathcal{I}$) $\mathcal{I}$ is a finite set of intervals I.

Definition 28 (*Set of Relationships* $\mathcal{RS}$) For two intervals, the finite set of interval relationships is $\mathcal{RS} := \{equal, before, meets, overlaps, during, starts, finishes\}$.

Definition 29 (*Interval Relation* R) A relationship between two intervals can be defined as a 4-tuple R consisting of the first interval I_j, the second interval I_k, the relationship between the two intervals rs, and the offset t (in case the relationship requires an offset);

$$R := \begin{cases} (I_j, I_k, rs, \emptyset), & I_j, I_k \in \mathcal{I}, rs \in \mathcal{RS} \\ & \text{if } rs \in \{equal, meets, starts, finishes\} \subset \mathcal{IR} \\ (I_j, I_k, rs, t), & I_j, I_k \in \mathcal{I}, rs \in \mathcal{RS}, t \in \mathbb{N}^+ \\ & \text{if } rs \in \{before, overlaps, during\} \subset \mathcal{IR} \end{cases} \tag{3.3}$$

Definition 30 (*Set of Interval Relations* $\mathcal{IR}$) $\mathcal{IR}$ is a finite set of interval relations R representing a presentation.

3.5.3.2 Example

We now use the example from Sect. 3.5.1.2 in an interval-based temporal model (see Fig. 3.8). As in the point-based example, the presentation has two videos (F_1 and F_2), one audio (F_3), one text (a_1), and three images (a_2, a_3, and a_4). The text is shown all of the time. The images are shown one after the other with a break in-between. First, video F_1 is shown, then video F_2, which is muted and audio F_3 is used to replace its sound.

The elements are defined as follows:

$$\begin{aligned} A_1 &= (a_1, 30) \\ A_2 &= (a_2, 7) \\ A_3 &= (a_3, 9) \\ A_4 &= (a_4, 10) \end{aligned}$$

$$\begin{aligned} F_1 &= (f_{1,1}, \ldots, f_{1,20}) \\ F_2 &= (f_{2,1}, \ldots, f_{2,10}) \\ F_3 &= (f_{3,1}, \ldots, f_{3,10}). \end{aligned} \tag{3.4}$$

Accordingly, the set of intervals is $\{I_{A_1}, I_{A_2}, I_{A_3}, I_{A_4}, I_{F_1}, I_{F_2}, I_{F_3}\}$.

Possible definitions of the set of interval relations for the given example are as follows:

$$\begin{aligned} IR_1 = \{&(I_{A_2}, I_{A_3}, before, 2), (I_{A_3}, I_{A_4}, before, 2), (I_{A_4}, I_{F_3}, equal, \emptyset), \\ &(I_{A_2}, I_{A_1}, starts, \emptyset), (I_{A_2}, I_{F_1}, starts, \emptyset), (I_{F_1}, I_{F_2}, meets, \emptyset)\} \\ IR_2 = \{&(I_{A_2}, I_{A_3}, before, 2), (I_{A_3}, I_{A_4}, before, 2), (I_{A_4}, I_{F_3}, equal, \emptyset), \\ &(I_{A_1}, I_{F_3}, ends, \emptyset), (I_{A_2}, I_{F_1}, starts, \emptyset), (I_{F_1}, I_{F_2}, meets, \emptyset)\} \\ IR_3 = \{&(I_{A_2}, I_{F_1}, starts, \emptyset), (I_{F_1}, I_{A_1}, starts, \emptyset), (I_{A_3}, I_{F_1}, during, 9), \\ &(I_{A_4}, I_{F_3}, equal, \emptyset), (I_{A_4}, I_{F_2}, equal, \emptyset), (I_{F_3}, I_{A_1}, ends, \emptyset)\}. \end{aligned}$$

3.6 Exemplary Applications

With the definitions given in the previous sections, we can now calculate values for algorithms (see Sect. 3.6.1) and give precise definitions of user interactions and resulting video behavior (see Sects. 3.6.2 and 3.6.3). These can also be used to calculate timing information in multimedia documents as, for example, done by Meixner in [38].

3.6.1 Basic Calculations and Definitions

Further definitions and various basic functions are useful for calculations in algorithms, which will be defined hereafter. Thereby, $\mathcal{X} \in \{\mathcal{A}_D, \mathcal{C}_D\}$ or $\mathcal{X} \in \{\mathcal{A}_V, \mathcal{C}_V\}$ and $\mathcal{X}^k$ is a tuple where k is the number of elements in the tuple.

The **frame rate** r of the video may either be a constant or variable over time t (as described in [12, 28, 42, 46]). A constant frame rate c_r is defined in Function 3.5. It is set to a fixed value c_r for all calculations in this work. Usually, c_r is set to 25 or 30 fps. We use $c_{r_{SFW}}$ as a constant for the slow-forward frame rate, $c_{r_{SBW}}$ as a constant for the slow rewind frame rate, $c_{r_{FFW}}$ as a constant for the fast-forward frame rate, and $c_{r_{FBW}}$ as a constant for the fast rewind frame rate. $\mathbb{R}^+$ is the set of positive real numbers (without zero).

$$r : \mathbb{R}^+ \mapsto \mathbb{N}^+, t \mapsto r(t) := c_r \tag{3.5}$$

A **dimension function** *dim* is needed to get the size/length of a tuple. This basic function is defined in Function 3.6.

$$dim : \mathcal{X}^k \mapsto \mathbb{N}^+, (x_1, \ldots, x_k) \mapsto dim(x_1, \ldots, x_k) := k \tag{3.6}$$

A **projection function** π_i is needed to get a specific value from a tuple. This basic function is defined in Function 3.7.

$$\pi_i : \mathcal{X}^k \mapsto \mathcal{X}, k \in \mathbb{N}^+, (x_1, \ldots, x_k) \mapsto \pi_i(x_1, \ldots, x_k) := x_i,\ 1 \leq i \leq k \tag{3.7}$$

A second **projection function** π_{i_1,i_2} can be used to get a part of a tuple. This basic function is defined in Function 3.8.

$$\begin{aligned} \pi_{i_1,i_2} : \mathcal{X}^j \mapsto \mathcal{X}^k, j, k \in \mathbb{N}^+, k \leq j, (x_1, \ldots, x_j) \mapsto \pi_{i_1,i_2}(x_1, \ldots, x_j) \\ := (x_{i_1}, \ldots, x_{i_2}), 1 \leq i_1 < i_2 \leq j, i_2 - i_1 + 1 = k \end{aligned} \tag{3.8}$$

Furthermore, a generalization from frames/samples and static multimedia elements to "downloadable objects" may simplify some calculations. The static multimedia elements $\mathcal{A}_D/\mathcal{A}_\mathcal{V}$ and the frames/samples $\mathcal{F}_D/\mathcal{F}_\mathcal{V}$ of a multimedia document D or an annotated interactive nonlinear video $\mathcal{V}$ are joined to a **set of (downloadable) elements** of a multimedia document $\mathcal{E}_D$ (see Definition 31) or of an annotated interactive nonlinear video $\mathcal{E}_\mathcal{V}$ (see Definition 32).

Definition 31 (*Set of Downloadable Elements $\mathcal{E}_D$ of a Multimedia Document*) $\mathcal{E}_D$ is a set of downloadable elements of a multimedia document D, which is defined as the union $\mathcal{E}_D = \mathcal{A}_D \cup \mathcal{F}_D$ of the set of frames/samples $\mathcal{F}_D$ and the set of static multimedia elements $\mathcal{A}_D$ of the multimedia document.

Definition 32 (*Set of Downloadable Elements $\mathcal{E}_\mathcal{V}$ of an Annotated Interactive Nonlinear Video*) $\mathcal{E}_\mathcal{V}$ is a set of downloadable elements of an annotated interactive nonlinear video $\mathcal{V}$, which is defined as the union $\mathcal{E}_\mathcal{V} = \mathcal{A}_\mathcal{V} \cup \mathcal{F}_\mathcal{V}$ of the set of frames/samples $\mathcal{F}_\mathcal{V}$ and the set of static multimedia elements $\mathcal{A}_\mathcal{V}$ of the multimedia document.

In the following functions, the set of downloadable elements $\mathcal{E}_\mathcal{V}$ of an annotated interactive nonlinear video and the set of downloadable elements $\mathcal{E}_D$ of a multimedia document are used interchangeable.

A **size function** s is defined in Function 3.9. It returns the size of an element by returning the length of the n-tuple of bits representing the content of the multimedia element. This function is needed to get the amount of data that has to be downloaded from the server or has to be stored in the cache for each multimedia element.

$$s : \mathcal{E}_D \to \mathbb{N}^+, e_i \mapsto s(e_i) := \begin{cases} dim(e_i) & \text{if } e_i \text{is a frame/sample} \\ dim(\pi_1(e_i)) & \text{if } e_i \text{ is an static multimedia element} \end{cases} \tag{3.9}$$

Function 3.10 returns the **priority** q of a static multimedia element by a projection on the second component of the static multimedia element pair (α_o, Λ). The higher the priority is, the lower is its number. The highest priority is "1". If no priorities are

used, all static multimedia elements are set to priority "1" and are treated with the same priority with which other elements are downloaded.

$$q : \mathcal{A}_D \to \mathbb{N}^+, a_o \mapsto q(a_o) := \pi_2(a_o) = \Lambda \tag{3.10}$$

The **duration of a continuous medium** F in seconds $l(F)$ is calculated by the division of the number of frames/samples of the continuous multimedia element $dim(F)$ by the frame/sample rate c_r at a fixed frame/sample rate. This is expressed by Function 3.11.

$$l : \mathcal{F}_D \to \mathbb{N}^+, F \mapsto l(F) := \frac{dim(F)}{c_r} \tag{3.11}$$

3.6.2 VCR Actions for Continuous Multimedia Elements

Besides play, pause, and stop, several other VCR actions are possible. It is also possible to play the medium backward with the frame rate used for playing it forward. Besides slow- and fast-forward or rewind, it is furthermore possible to jump to a certain frame/sample forward or backward in the currently played medium. Table 3.2 enlists all actions which may be considered in a single medium. For each action, first the current frame is given, and then the current frame rate is stated. Thereby, f_m is a frame in a scene. If the current frame rate is 0, the playback is currently stopped/paused, and if it is c_r, the video is playing with a constant frame rate. After the user interaction, either frame or frame rate changes. Depending on the current state of the video, some restrictions may apply, see remark column in Table 3.2.

3.6.3 Extended Interactivity and Navigation

Annotated interactive nonlinear videos (as described in Sect. 3.5.2) provide additional features besides the VCR actions described in Sect. 3.6.2. The following facts are summarized in Table 3.3. As in Table 3.2, pre- and post-interaction frame and frame rate are shown. Because scene changes may occur, also an index for the scene is given for current and new frame. We assume that the selection panels or quizzes are usually displayed after a scene ends. The user decides which scene should be displayed next either by selecting it directly in a button panel or by solving a quiz. In the latter case, the follow-up scene is chosen by the score reached in the quiz. Each score is assigned to a point range of a scene which is then selected accordingly. Jumps in the whole video which are not depending on the underlying graph structure between the scenes are selections in a table of contents or a selection in search results. When a user opens the table of contents, the video may stop and continue playing after a user selection, or it may continue playing, depending on the positioning of the table of contents (side area of the player or overlay on the video). The selected entry starts the

Table 3.2 Different intrascene VCR actions

Action	Current frame	Current frame rate	New frame	New frame rate	Remark
Stop	f_m	c_r	f_σ	0	
Pause	f_m	c_r	f_m	0	
Play forward	f_m	0	f_{m+1}	c_r	f_m not last frame of a scene
	f_m	c_r	f_{m+1}	c_r	f_m not last frame of a scene
Play backward	f_m	0	f_{m-1}	c_r	f_m not first frame of a scene
	f_m	c_r	f_{m-1}	c_r	f_m not first frame of a scene
Jump forward	f_m	c_r	f_{m+x}	c_r	$x \in \mathbb{N}^+, f_m, f_{m+x}$ in same scene
Jump backward	f_m	c_r	f_{m-x}	c_r	$x \in \mathbb{N}^+, f_m, f_{m-x}$ in same scene
Slow-forward	f_m	c_r	f_{m+1}	$c_{r_{SFW}}$	$c_r < c_{r_{SFW}}, f_m$ not last frame of a scene
Slow rewind	f_m	c_r	f_{m-1}	$c_{r_{SBW}}$	$c_r < c_{r_{SBW}}, f_m$ not first frame of a scene
Fast-forward	f_m	c_r	f_{m+1}	$c_{r_{FFW}}$	$c_r < c_{r_{SFW}} < c_{r_{FFW}}, f_m$ not last frame of a scene
Fast rewind	f_m	c_r	f_{m-1}	$c_{r_{FBW}}$	$c_r < c_{r_{SBW}} < c_{r_{FFW}}, f_m$ not first frame of a scene
Pan/tilt/zoom	f_m	c_r	f_{m+1}	c_r	User interaction to modify the presentation of the following frames

Table 3.3 Interactive and navigational actions which are possible in a single scene (intrascene) or in-between scenes (interscene)

Action	Current frame	Current frame rate	New frame	New frame rate	Remark
Selection/quiz	$f_{a,m}$	0	$f_{b,1}$	c_r	$f_{a,m}$ last frame of a scene, $f_{b,1}$ first frame of selected successor scene
TOC	$f_{a,m}$	c_r	$f_{a,m}$	0	User interaction to invoke table of contents at frame $f_{a,m}$
	$f_{a,m}$	0	$f_{a,1} \vee f_{b,1}$	c_r	Selection in overlay table of contents at frame $f_{a,m}$, jump to first frame $f_{a,1}$ of same scene or $f_{b,1}$ of other selected scene; thereby, the video resumes playing at the new position
	$f_{a,m}$	c_r	$f_{a,1} \vee f_{b,1}$	c_r	Selection in side area table of contents at frame $f_{a,m}$, jump to first frame $f_{a,1}$ of same scene or $f_{b,1}$ of other selected scene
Keyword search	$f_{a,m}$	c_r	$f_{a,m}$	0	User interaction to invoke keyword search at frame $f_{a,m}$
	$f_{a,m}$	0	$f_{a,k} \vee f_{b,k}$	c_r	Select annotation in search results at frame $f_{a,m}$, jump to frame $f_{a,k}$ of same scene or $f_{b,k}$ of other scenes where selected annotation is displayed
	$f_{a,m}$	0	$f_{a,1} \vee f_{b,1}$	c_r	Select scene in search results at frame $f_{a,m}$, jump to first frame $f_{a,1}$ of same scene or $f_{b,1}$ of other scenes where selected annotation is displayed

playback of a scene at its beginning. A search is usually carried out during the playback of a scene. It is possible to jump to the beginning of a scene or to an annotation in a scene. Interactive functions like pan, tilt, and zoom have no influence on the order of the displayed frames or the frame rate. They may rather increase the download volume, because higher resolutions of single frames or other camera positions are needed at client side [38].

3.7 Conclusion

In this chapter, we propose formal definitions of temporal models as well as other functions that are important for multimedia synchronization, scheduling, and management in applications. Thereby, we focus on the most important existing temporal models, namely, point-based, event-based, and interval-based temporal models which have been proposed and described in previous related work. We summarize the descriptions given in related literature, show advantages and disadvantages of the temporal models, and point out issues that originate from these temporal models. While the temporal models have been widely used, no commonly used formal and precise way of defining them is available. This makes transforming one temporal model into another or performing calculations on one of the temporal models tedious. To overcome this issue, we present formal definitions for commonly used relationships, timing, and interactivity of each temporal model which can easily be extended with additional features. We use each definition in a small example to show its practical usage for defining presentations. After that, we give further definitions of basic functions that are useful for calculations on the temporal models and for the definition of algorithms. We also show how the proposed formalized temporal models and definitions can be used to describe possible user interaction and the following reaction of the video.

In the era of Web with technologies like HTML5, Adaptive Streaming (e.g., DASH), user-generated live video, and Netflix, these temporal models can be used to describe synchronization constraints, to describe scheduling algorithms, or to manage elements during download or streaming and while they reside in the (client) cache. The temporal models make calculations in theoretical frameworks possible which can then be transformed into real-world technologies after initial tests. This is especially useful for standards that are not finished yet or where browser implementations are still in development.

Appendix

Table 3.4 List of symbols

Symbol	Explanation
A	Symbol for the display duration of a static multimedia element
A_i	Symbol for the ith display duration
F	Symbol for a continuous media element
F_i	Symbol for the ith continuous media element
I	Symbol for an interval
I_e	Symbol for an interval for multimedia element e
R	Symbol for a relationship between two intervals
T	Symbol for a time interval
a	Symbol for an annotation
a_i	The ith annotation of a video
c	Symbol for constant values
c_r	Constant for the frame rate (normal speed)
$c_{r_{SFW}}$	Constant for the frame rate (slow-forward)
$c_{r_{SBW}}$	Constant for the frame rate (slow rewind)
$c_{r_{FFW}}$	Constant for the frame rate (fast-forward)
$c_{r_{FBW}}$	Constant for the frame rate (fast rewind)
dim	Symbol for the function returning the length of a tuple
e	Symbol for a (downloadable) element
ea	Symbol for an executable action
ee	Symbol for an element event
ep	Symbol for an event point
e_i	The ith element of set $\mathcal{E}_D$
f	Symbol for a frame/sample
f_σ	Symbol for the start frame/sample
f_ϵ	Symbol for the end frame/sample
f_i	Symbol for the ith frame/sample
$f_{i,m}$	The mth frame/sample of scene i
i	Index
j	Index
j_i	Last frame index of scene i
k	Index
l	Symbol for the duration function
n	Index

(continued)

Table 3.4 (continued)

Symbol	Explanation
p	Symbol for a scene
p_σ	Symbol for the start scene
p_ϵ	Symbol for the end scene
p_i	Scene i in set $\mathcal{P}_\mathcal{V}$
q	Symbol for the function returning the priority of an element
r	Symbol for the frame rate function
s	Symbol for the function returning the size of an element
t	Symbol for the time
t_i	The ith point in time
tl	Symbol for a timeline
w	Symbol for a button
w_j	The jth button
x	Index
α	Symbol for the content of an annotation
α_o	The content of the oth annotation in set $\mathcal{A}_D$
δ	Symbol for the transition function of the DFA
ϵ	Symbol for the end of the video
Λ	Priority of an annotation
π	Symbol for the projection function
π_i	Symbol for the projection function on the ith element of a tuple
$\pi_{i,j}$	Symbol for the projection function on the ith and jth element of a tuple
σ	Symbol for the start of the video
Σ	Input symbols of the DFA
$\mathcal{D}$	Symbol for a multimedia document
$\mathbb{N}$	Set of natural numbers
$\mathbb{N}^+$	Set of positive natural numbers (without zero)
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^+$	Set of positive real numbers (without zero)
$\mathcal{A}_D$	Set of static multimedia elements of $\mathcal{D}$
$\mathcal{A}_\mathcal{V}$	Set of static multimedia elements of $\mathcal{V}$
$\mathcal{C}_D$	Set of continuous multimedia elements of $\mathcal{D}$
$\mathcal{E}_D$	Set of (downloadable) elements of $\mathcal{D}$
$\mathcal{E}_\mathcal{V}$	Set of (downloadable) elements of $\mathcal{V}$
$\mathcal{EA}$	Set of executable actions
$\mathcal{EA}_a$	Set of executable actions for static multimedia elements
$\mathcal{EA}_F$	Set of executable actions for dynamic multimedia elements
$\mathcal{EE}$	Set of element events
$\mathcal{EP}$	Set of event points

(continued)

Table 3.4 (continued)

Symbol	Explanation
$\mathcal{F}_D$	Set of frames/samples of D
$\mathcal{F}_\mathcal{V}$	Set of frames/samples of $\mathcal{V}$
$\mathcal{I}$	Set of intervals
$\mathcal{IR}$	Set of interval relations
$\mathcal{N}_\mathcal{V}$	Set of transitions of $\mathcal{V}$
$\mathcal{P}_\mathcal{V}$	Set of scenes of $\mathcal{V}$
$\mathcal{RS}$	Set of relationships
$\mathcal{X}$	Random set of single elements
$\mathcal{X}^k$	Random set of k-tuples
$\mathcal{V}$	Symbol for an interactive nonlinear video
$\exists$	Existential quantifier ("there exists")
$\exists!$	Existential quantifier ("there exists exactly one")
$\nexists$	Existential quantifier ("there does not exist")
$\forall$	Universal quantifier ("for all")
$\cap$	Intersection of sets
$\cup$	Union of sets
$\|\circ\|$	Cardinality of a set

Definitions

Multimedia element A multimedia element is an image, a video, an audio, a text or any other type of audiovisual medium. It is the atomic object of any multimedia document.

Annotation An annotation is additional information displayed with a main medium. It consists of an anchor attaching it to the main medium and a body. The body of an annotation is a multimedia element that can be shown in a player [39].

Static multimedia element Static multimedia elements are time independent and always show the same content, like images and/or text.

Continuous multimedia element Continuous multimedia elements are time dependent showing/playing different contents over time, like videos or audios.

Hyperlinked media Hyperlinked media are multimedia elements which are linked with each other by hyperlinks (as known from hypertext). Static media may be clickable or have clickable areas. Continuous media may provide links depending on the media time.

Media synchronization Synchronization of multimedia elements requires mechanisms to prepare the media for display (i.e. pre-fetch, buffering, rendering) and to ensure that timing constraints are met.

Multimedia document A multimedia document is a *self-contained* presentation of linked and synchronized multimedia elements which allows user interaction and navigation. Usually it is about a certain topic.

Point A (time) point is a precise moment in time [41]. It is synchronized with a clock.

Event An event is something that happens or takes place [41]. It may be triggered by a clock or by a user interaction.

Interval A (time) interval is the time between start and end of a time span.

Hypermedia Hypermedia is an extension to hypertext providing multimedia facilities, such as those handling sound and video [41]. Keeping the hyperlink structure from hypertext, multimedia elements of different types are added.

Multimedia Multimedia uses a variety of artistic or communicative media that are presented in one presentation [41].

Passive multimedia Passive multimedia presentations are started and then watched with little to no interaction. Available forms of interaction are starting, pausing, and stopping the presentation.

Active multimedia Active multimedia presentations allow more interaction compared to passive multimedia presentations. They may have hyperlinks or other interactive control elements.

References

1. AFNOR Expert Group: Multimedia synchronization: Definitions and model, input contribution on time variant aspects and synchronization in oda-extensions. ISO IE JTC 1/SC 18/WG3 (1989)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
3. Appelt, W., Scheller, A.: HyperODA–going beyond traditional document structures. Comput. Stand. Interfaces **17**(1), 13–21 (1995)
4. Benbernou, S., Makhoul, A., Hacid, M.S., Mostefaoui, A.: A spatio-temporal adaptation model for multimedia presentations. In 7th IEEE International Symposium on Multimedia (ISM'05), pp. 8–pp. Dec (2005)
5. Blakowski, G., Steinmetz, R.: A media synchronization survey: reference model, specification, and case studies. IEEE J. Sel. Areas Commun. **14**(1), 5–35 (1996)
6. Blakowski, G., Hübel, J., Langrehr, U., Mühlhäuser, M.: Tool support for the synchronization and presentation of distributed multimedia. Comput. Commun. **15**(10), 611–618 (1992)
7. Boll, S., Klas, W., Westermann, U.: A comparison of multimedia document models concerning advanced requirements. Technical Report UIB-1999–01, DBIS (1999)
8. Boll, S., Klas, W.: ZYX—A Semantic Model for Multimedia Documents and Presentations, pp. 189–209. Springer, US, Boston, MA (1999)
9. Buchanan, M.C., Zellweger, P.T.: Automatic temporal layout mechanisms. In: Proceedings of the First ACM International Conference on Multimedia, MULTIMEDIA '93, pp. 341–350. ACM, New York, NY, USA (1993)
10. Buchanan, M.C., Zellweger, P.T.: Automatic temporal layout mechanisms revisited. ACM Trans. Multimedia Comput. Commun. Appl. **1**(1), 60–88 (2005)
11. Campbell, and A. N. Habermann. The specification of process synchronization by path expressions. In: Operating Systems, Proceedings of an International Symposium, pp. 89–102. Springer, London, UK (1974)

12. Chen, J.-J., Hang, H.-M.: Source model for transform video coder and its application ii. variable frame rate coding. IEEE Trans. Circuits Syst. Video Technol. **7**(2), 299–311 (1997)
13. Courtiat, J.P., De Oliveira, R.C.: Proving temporal consistency in a new multimedia synchronization model. In: Proceedings of the 4th ACM International Conference on Multimedia, MULTIMEDIA '93, pp. 141–152. ACM, New York, NY, USA (1996)
14. Drapeau, G.D.: Synchronization in the maestro multimedia authoring environment. In: Proceedings of the First ACM International Conference on Multimedia, MULTIMEDIA '93, pp. 331–339. ACM, New York, NY, USA (1993)
15. Duda, A., Keramane, C.: Structured temporal composition of multimedia data. In: 1995 Proceedings International Workshop on Multi-Media Database Management Systems, pp. 136. Aug 1995
16. Eijk, P.V., Diaz, M.: Formal Description Technique Lotos: Results of the Esprit Sedos Project. Elsevier Science Inc., New York, NY, USA (1989)
17. Euzenat, J., Layaïda, N., Dias, V.: A semantic framework for multimedia document adaptation. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence IJCAI'2003, pp. 31–36. Morgan Kauffman, Acapulco, Mexico, Aug 2003
18. Fiume, E., Tsichritzis, D., Dami, L.: A temporal scripting language for object-oriented animation. In: EG 1987-Technical Papers Eurographics Association (1987)
19. Fujikawa, K., Shimojo, S., Matsuura, T., Nishio, S., Miyahara, H.: Multimedia presentation system "harmony" with temporal and activemedia. In: Multimedia for Now and the Future, USENIX (1991)
20. Gibbs, S.J., Breiteneder, C., Tsichritzis, D.: Audio/video databases: an object-oriented approach. In: Proceedings of the 9th International Conference on Data Engineering, pp. 381–390. IEEE Computer Society, Washington, DC, USA (1993)
21. Gotthardt, K.: Grundlagen der Informationstechnik. LIT Verlag Münster, Einführungen–Informatik (2001)
22. Hirzalla, N., Falchuk, B., Karmouch, A.: A temporal model for interactive multimedia scenarios. IEEE Multimedia **2**(3), 24–31 (1995)
23. Hoepner, P.: Synchronisation der Präsentation von Multimedia-Objekten, pp. 455–464. Springer, Berlin Heidelberg (1991)
24. Illik, J.: Formale Methoden der Informatik: Von der Automatentheorie zu Algorithmen und Datenstrukturen. Expert, Reihe Technik (2009)
25. ISO/EIC: Iso/iec 13522-6:1998(en) information technology—coding of multimedia and hypermedia information. Website (1998)
26. ISO/IEC: Information technology—hypermedia/time-based structuring language (hytime) (iso/iec jtc 1/sc 34) (1997)
27. Kim, W., Kenchammana-hosekote, D., Lim, E.P., Srivastava, J.: Synchronization relation tree: a model for temporal synchronization in multimedia presentations. Technical Report (1992)
28. Kim, J.W., Kim, Y.-G., Song, H., Kuo, T.-Y., Chung, Y.J., Kuo, C.-C.J.: Tcp-friendly internet video streaming employing variable frame-rate encoding and interpolation. IEEE Trans. Circuits Syst. Video Technol. **10**(7), 1164–1177 (2000)
29. Kretz, F., Colaitis, F.: Standardizing hypermedia information objects. Comm. Mag. **30**(5), 60–70 (1992)
30. Little, T.D.C., Ghafoor, A.: Interval-based conceptual models for time-dependent multimedia data. IEEE Trans. Knowl. Data Eng. **5**(4), 551–563 (1993)
31. Little, T.D.C., Ghafoor, A.: Scheduling of bandwidth-constrained multimedia traffic. In: Proceedings of the 2nd International Workshop on Network and Operating System Support for Digital Audio and Video, pp. 120–131. Springer, London, UK (1992)
32. Little, T.D.C., Ghafoor, A.: Synchronization and storage models for multimedia objects. IEEE J. Sel. Areas Commun. **8**(3), 413–427 (1990)
33. Little, T.D.C., Ghafoor, A.: Spatio-temporal composition of distributed multimedia objects for value-added networks. Computer **24**(10), 42–50 (1991)
34. Macromind Director: Version 3.0: Overview Manual. MacroMind (1991)

35. Meixner, B., Einsiedler, C.: Download and cache management for HTML5 hypervideo players. In: Proceedings of the 27th ACM Conference on Hypertext and Social Media HT '16, pp. 125–136. ACM, New York, NY, USA (2016)
36. Meixner, B., John, S., Handschigl, C.: Siva suite: framework for hypervideo creation, playback and management. In: Proceedings of the 23rd ACM International Conference on Multimedia, MM '15, pp. 713–716. ACM, New York, NY, USA (2015)
37. Meixner, B., Kosch, H.: Interactive non-linear video: definition and xml structure. In: Proceedings of the 2012 ACM Symposium on Document Engineering, DocEng '12, pp. 49–58. ACM, New York, NY, USA (2012)
38. Meixner, B.: Annotated interactive non-linear video–software suite, download and cache management. Ph.D. thesis, Universität Passau (2014)
39. Meixner, B.: Hypervideos and interactive multimedia presentations. ACM Comput. Surv. **50**(1), 9:1–9:34 (2017)
40. Ogawa, R., Harada, H., Kaneko, A.: Scenario-based hypermedia: a model and a system. In: Rizk, A., Streitz, N., Andre, J. (eds.) Hypertext: Concepts, Systems and Applications–Proceeding the First European Conference on Hypertext, pp. 38–51. Cambridge University Press, Cambridge (1990)
41. Oxford University Press: British & world english. Website https://en.oxforddictionaries.com/ (2017). Accessed 03 May 2017
42. Pan, F., Lin, X., Rahardja, S., Lim, K.P., Li, Z.G., Wu, D.J., Wu, S.: Proactive frame-skipping decision scheme for variable frame rate video coding. In: 2004 IEEE International Conference on Multimedia and Expo 2004. ICME '04, Vol. 3, pp. 1903–1906. IEEE (2004)
43. Poole, L.: Quicktime in motion. MacWorld. **8**(9), 154–159 (1991)
44. Rousseau, F., Duda, A.: An execution architecture for synchronized multimedia presentations, pp. 42–55. Springer, Berlin, Heidelberg (1998)
45. Shepherd, D., Salmony, M.: Extending osi to support synchronization required by multimedia applications. Comput. Commun. **13**(7), 399–406 (1990)
46. Shue, J.-S., Hsieh, C.-H., Tsai, H.-S., Wang, C.-C.: Variable-rate video codec using frame adaptive finite-state vector quantization. In: 1993 IEEE International Symposium on Circuits and Systems, 1993 ISCAS '93, vol. 1, pp. 28–31. IEEE (1993)
47. Steinmetz, R.: Synchronization properties in multimedia systems. IEEE J. Sel. A. Commun. **8**(3), 401–412 (1990)
48. Vazirgiannis, M., Kostalas, I., Sellis, T.: Specifying and authoring multimedia scenarios. IEEE Multimedia **6**(3), 24–37 (1999)
49. Wahl, T., Rothermel, K.: Representing time in multimedia systems. In: Proceedings of the International Conference on Multimedia Computing and Systems, pp. 538–543. May 1994
50. Wahl, T., Wirag, S., Rothermel, K.: Tiempo: temporal modeling and authoring of interactive multimedia. In: Proceedings of the International Conference on Multimedia Computing and Systems, pp. 274–277. May 1995

Chapter 4
Time, Frequency and Phase Synchronisation for Multimedia—Basics, Issues, Developments and Opportunities

Hugh Melvin, Jonathan Shannon and Kevin Stanton

Abstract In this chapter, we provide a comprehensive overview of timing. We describe the underlying concepts that comprise timing through examples and then present a range of mature, standardised and evolving techniques to improve the so-called time awareness across the full Information and Communications Technology (ICT) infrastructure over which multimedia applications operate. Although the media synchronisation community is already acutely aware of timing issues, this chapter offers some valuable insights through its holistic approach to timing.

Keywords Timing · Time awareness · Time synchronisation protocols Time-sensitive networking

4.1 Introduction

Although the scope of media synchronisation is quite broad, encompassing a range of scenarios including intra-media, inter-media, multi-source, inter-device synchronisation, a common requirement relates to a sense of timing. The term 'timing', as applied both in this chapter and internationally, is used as an umbrella term that represents three different concepts—time, phase or frequency. In this chapter, we first define these three concepts and then describe how the term synchronisation applies to each of the three, providing examples to illustrate and distinguish between the three. We then describe the challenges that arise in implementing these timing concepts in modern Information and Communications Technology (ICT). This then raises the notion of time awareness and its key role across the full end-to-end infrastructure over

H. Melvin (✉) · J. Shannon
School of Computer Science, National University of Ireland, Galway, Republic of Ireland
e-mail: hugh.melvin@nuigalway.ie

K. Stanton
Intel Corporation, Santa Clara, USA

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_4

which multimedia applications operate. In this context, we introduce the work of the recently (2014) formed Interest Group (IG) termed Time-Aware Applications, Computers and Communications Systems (TAACCS) http://www.taaccs.org/. Although the scope of TAACCS is quite broad, and extends to the so-called Internet of Everything (IoE), the media synchronisation community, though already acutely aware of timing issues, can learn a lot from its holistic approach. We then review a range of mature techniques and more recent developments that collectively help deliver better timing and time awareness for multimedia applications. This includes a summary of time and frequency distribution protocols, such as the Network Time Protocol (NTP), Precision Time Protocol (PTP), Synchronous Ethernet, as well as recent standardisation developments in Time-Sensitive Networking (TSN). The chapter concludes by reviewing the significant challenges to achieving precision timing on multimedia end devices, addressing both software and hardware issues.

4.1.1 The Basics of Timing

In this section, we draw significantly from the ITU-R TF.686-3 for definitions—see [1] for more detail. As mentioned above, we use the word timing as an umbrella term for time, phase and frequency. Time is used 'to specify an instant (or time of day) on a selected timescale' [1]. An accurate time clock is one that is traceable to a timescale standard, such as Universal Coordinated Time (UTC) or International Atomic Time (TAI). A timescale is defined by ITU-R TF.686-3 as 'A family of time codes for a particular coordinate time that provide an unambiguous time ordering of events' [1].

TAI is an atomic timescale and differs from UTC in that the latter is discontinuous, requiring adjustments known as leap seconds to keep it aligned with variations in the rotation of the earth. Time synchronisation is defined as the 'relative adjustment of two or more sources of time with the purpose of cancelling their time differences' [1] and is especially important for multimedia applications where each-way latency measurement is required across a network of multimedia devices or where aligning of media streams is essential for a good Quality of Experience (QoE). For example, interactive media applications such as Voice over IP (VoIP) often have a tight limit of Mouth-to-Ear delay or latency, and thus measuring such delay is very important for optimising jitter buffer parameters in real-time or as part of quality assurance purposes that form part of a Service Level Agreement (SLA). Similarly, multi-source inter-media synchronisation such as Hybrid Broadcast & Broadband (HBB) services requires that the media at each source device is timestamped with a system-wide agreed timescale, so that the media streams can be correctly aligned/synchronised on the final playout device. Whilst a locally agreed timescale can be sufficient when synchronising multiple distributed nodes in a small-scale network, it is often more convenient to utilise a global timescale such as UTC. As such, so-called inter-device synchronisation (IDES) that requires all receiving devices to share a common time reference (to correctly implement seamless playout across devices) typically does this by using a global standard.

For standalone multimedia devices that do not interact with the outside world, there is typically no need to be time-synchronised.

Frequency refers to a rate of repetition (of an event) per unit of time and thus two clocks are frequency synchronised if they oscillate at the exact same rate—alternatively, a clock is defined as accurate in frequency if its rate agrees with the definition of the second, e.g. an accurate 100 Hz clock oscillates exactly 100 times in 1 s. The ITU-R TF.683 defines the term syntonisation as 'The relative adjustment of two or more frequency sources with the purpose of cancelling their frequency differences but not necessarily their phase difference'. However, the term frequency synchronisation is also widely used instead of syntonisation. Typical units to describe the extent of frequency synchronisation are parts per million (ppm), where a clock gains/loses a number of microseconds per second, or parts per billion (ppb), where clock gains/loses a number of nanoseconds per second. 1 ppm is equivalent to 0.0001%. Frequency synchronisation is a critical requirement for intra-media synchronisation in that if the clock on a media producing device is running at a different rate than the clock on the media consuming device, buffer overflow or underflow situations will occur, with resulting QoE issues, such as high delay/packet loss or discontinuities.

With regard to phase, two clocks are frequency and phase synchronised if they not only operate at the same frequency (frequency synchronised), but they both reach peak clock signal levels at the same instant. In many cases, time synchronisation to a global timescale, such as UTC, is used to implement phase synchronisation, e.g. to ensure that samples are taken at the exact same instant across geographically separated nodes in a distributed system.

4.1.2 Challenges for Timing in Information and Communications Technology (ICT) Infrastructure

At the heart of most consumer-grade multimedia systems are quartz crystal oscillators. These are used for example to maintain system time or simply to set the sample/playback rate of media. Figure 4.1 illustrates the operation of a crystal within a hypothetical computer system clock. The oscillator is manufactured to run at a certain frequency—a register is incremented/decremented for every oscillation and when it rolls over to zero, it generates a tick that generates an interrupt. This in turn calls a clock handler that adjusts a register representing, for example, UTC on the device. This process is subject to a range of errors, principle amongst them is the fact that the actual oscillator frequency may at manufacture differ from its nominal value, and second that its frequency is impacted by ageing, impurities and external factors such as temperature, pressure and voltage.

As an example, consumer-grade oscillators found in Ethernet NIC are designed to operate within ±100 ppm band, which is sufficient for the asynchronous transmission deployed. The accumulation of timing errors in crystals can result in a range of issues for multimedia devices, such as buffering and media synchronisation, that must be addressed.

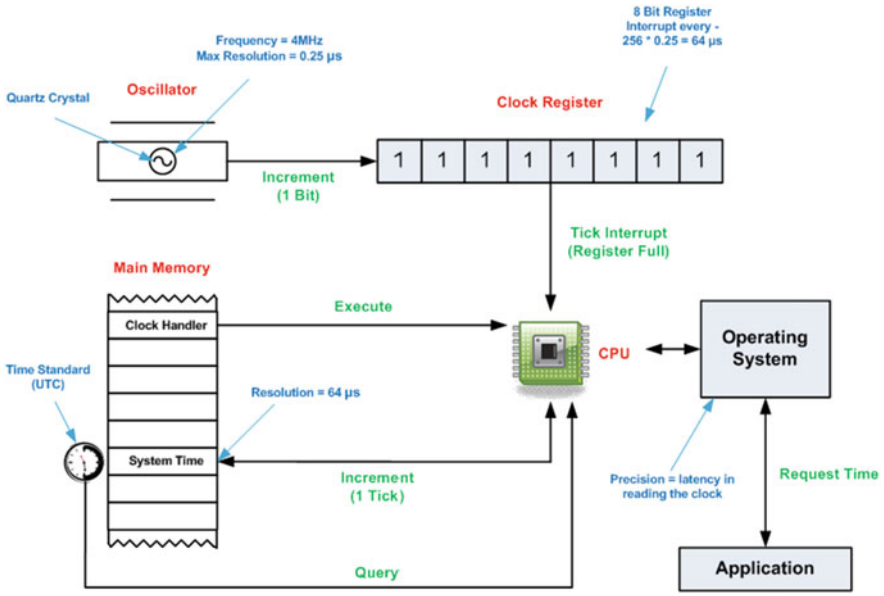


Fig. 4.1 Computer system clock architecture

With the growth in Internet connectivity, the complexity of multimedia systems has greatly increased. For Real-Time Communication (RTC) applications in particular, such as voice, video and gaming, timing issues can have a very significant impact on QoE. Taking an abstracted view of the full end-to-end infrastructure for complex distributed multimedia systems, a common problem is that the design of applications, computers and communications systems has evolved such that whilst it optimises data processing, it often degrades timing performance such that achieving and maintaining time, frequency and phase synchronisation to high levels of accuracy is very challenging. In practical terms, this results in applications, computers and networks that are designed for fairness and throughput rather than temporal determinism. As a result, whilst advances in programming languages, multicore devices with complex pipelining and cache prediction strategies, time sharing operating systems and best-effort networks all represent significant advances when measured against metrics such as throughput and fairness, they cause problems for applications with special timing needs. Whilst solutions to some of these challenges have been achieved in niche sectors—especially in so-called Hard Real-Time Systems (RTS), such as those found in critical infrastructure and key resources (CIKR), the solutions are often hardware specific, inflexible, sometimes proprietary, typically not scalable and often expensive. The challenge for so-called Soft Real-Time Systems, such as RTC applications, is that the established paradigm of fairness and throughput is unsuited to delivering temporal determinism and whilst some work has been done to address the challenges, much more is needed.

4.1.3 Time-Aware Applications, Computers and Communications Systems (TAACCS)

Time awareness is a recently coined term and can be defined as the extent to which a device, system or device/system model has an appropriate ability to sense and respond to timing signals/information. For RTC applications, we argue that to optimise QoE for the end user, the full end-to-end infrastructure needs to become more time-aware. In 2014, a US-based Interest Group termed **Time-Aware Applications, Computers and Communications Systems** (TAACCS) was formed and published a white paper, that has since been released as a NIST Technical Note [2] [http://dx.doi.org/10.6028/NIST.TN.1867]. This document takes an holistic view of time awareness—both in terms of where it stands today and where it needs to be in the future. The scope of the document is broad—it looks at time awareness in the context of the IoE with predicted billions of interconnected devices. It examines the extent to which time awareness is present across the full end-to-end infrastructure over which IoE applications will run, identifies gaps between the state of the art (SOTA) and new application-specific requirements and points in the direction of future required research to meet challenges and achieve expected IoE potential. TAACCS then identifies a range of topics where cross-disciplinary research is required so that time awareness can be achieved. These include: – the fundamental building blocks of components, such as oscillators, that are found throughout the infrastructure; – time awareness on endpoint device subsystems, such as operating systems, application software; – application development environments; – time awareness across networks; – time awareness in the cloud and finally – protocols to distribute time and frequency across such networks.

In the following section, we briefly examine these topics focusing on what relevance they hold for the multimedia community:

- *Oscillators*: These are, ultimately, the heart of a system's timing. The design usually involves a range of tradeoffs regarding performance, power and cost. A whole range of multimedia QoE problems are caused by poor oscillators and/or poor compensating strategies. There is a drive to eliminate multiple oscillators on consumer-grade devices, largely to reduce power consumption. Such a move may help to reduce potential multimedia skew impact by having a single source of timing.
- *Time Awareness on End Devices and Applications*: This will need cross-disciplinary research in the following areas:
  - Hardware and software support for predictable execution: This is a huge challenge for both, as it goes against the recent design trends for throughput and abstraction. Software includes the application software, driver software and the underlying operating system. The concept of 'Time Correctness by Design' whereby timing issues are adequately addressed at design stage would revolutionise timing performance for multimedia applications and resulting QoE. Recent work such as [3, 4] outlines some work in this

direction by developing time-aware programming models and microkernel design, respectively.
- Similarly, the work in [5] argues that whilst a lot of work has been done with regard to delivering precision timing to end devices via time transfer systems, the availability and efficient use of such precision on the devices has not kept pace with these developments, particularly when operating on virtualised hardware.
- Timing across interfaces will require standards and latency control especially when crossing network domains.
- Time Support within Media Protocols: For applications such as HBB or IDES, the need exists to map media timestamps to real UTC (or other common reference timescale) timestamps. This facilitates transporting the desired temporal relationship to end devices so that streams can be synchronised for playout, either on one device or on multiple devices (IDES). The Internet Engineering Task Force (IETF) Real-Time Transport Protocol (RTP), in conjunction with its control protocol Real-Time Transport Control Protocol (RTCP) (www.ietf.org/rfc/rfc3550.txt), provide one approach—e.g. the RTP profile for MPEG2 relates RTP timestamps to Programme Clock Reference (PCR) and RTCP Sender Report (SR) packets relate RTP to NTP [6]. The AES67 standard [7] further provides further specificity of RTP when IEEE 1588 [8] time is available. Similarly, HTTP Adaptive Streaming (HAS) solutions, such as Moving Pictures Expert Group Dynamic Adaptive Streaming over HTTP (MPEG-DASH) (www.iso.org/standard/57623.html), have timestamps embedded in media, which can be used for synchronisation. The IEEE 1722 standard [9] provides yet another option for the transport of time-sensitive media streams.

- *Time-Aware Development Environments* (*DE*): The environments that we use to specify, model and develop systems will need to evolve to support timing accuracy. This is a very big challenge, as it requires time to be built in as a correctness criterion and that the DE will work to that criteria, or at least let you know if criteria cannot be met. Essentially, a time-aware DE allows developers to specify time/frequency constraints (for a multimedia application), and the DE assists in design and construction of code to meet requirements. As such, the resulting application will satisfy both logical correctness and temporal correctness constraints. Although some work has been done in this area with the likes of Unified Modelling Language/Specification and Description Language (UML/SDL) and products such as LabView, many challenges remain.
- *Time-Aware Networks*: This can be defined in the context of the previous general definition of time awareness as a network that has an appropriate ability to sense and respond to timing signals/information. Such requirements raise significant challenges in a number of areas:

  - Network hardware and software will need new designs to both support and make use of time awareness. Significant progress has been made in this area under the umbrella term of Time-Sensitive Networking (TSN) developed by the IEEE 802.1 TSN [10] Task Group, detailed later.

  - Network performance monitoring is required both for technical and financial reasons in the form of SLAs, and requires time synchronisation to varying degrees.
  - Software-Defined Networking (SDN) and Network Function Virtualisation (NFV) aim to improve certain aspects of network performance metrics through abstraction, flexible virtualisation and centralised big-picture intelligence. Having precise time information can greatly enhance decision making for SDN and NFV.

- *Time-Aware Cloud Computing*: As the role of cloud computing becomes increasingly pervasive, there is a need to ensure that time awareness challenges arising from the growing role of cloud infrastructure in multimedia applications are considered. For example, what additional latencies and jitter, if any, does the rollout of Virtual Machines, as opposed to distinct servers, introduce?
- *Time and Frequency Distribution*: The growing need for timing synchronisation for multimedia applications, and the tighter precision with which it is needed can only be satisfied if the protocols for delivering time and frequency across networks to multiple endpoints are fit for purpose. Time transfer protocols, such as NTP and PTP, described in subsequent sections, will need to deliver time to an exponential increase in endpoints, with varying application-specific requirements for time and frequency synchronisation. The performance of NTP, described later, is very much dependent on the underlying network as well as endpoint hardware and software. PTP, also detailed later, implements packet timestamping at the MAC/PHY layer and thus eliminates much endpoint non-determinism found with NTP. However, though capable of delivering orders of magnitude better time synchronisation than NTP, maximum benefit requires care in selecting and managing network infrastructure with on-path PTP support to measure and compensate for congestion delays along the path of the PTP timing packets. Frequency distribution/transfer is a further key challenge for multimedia, especially when phase lock loop/frequency lock loop approaches are used over packet-based, nondeterministic networks. Synchronous Ethernet (SynchE) offers great potential in this regard and is very different from standard Ethernet interface in that it locks the frequency of each physical transmitter within the whole network, but comes at significant financial cost. Combining PTP with SynchE provides excellent time and frequency transfer, but is often cost prohibitive, and thus currently deployed only in utility networks such as Telecommunications and Power Systems.

One final message that is core to the TAACCS philosophy is that research and advances in these areas will yield more innovative Time-Aware Applications. Essentially, as timing, and for multimedia, we are concerned largely with time and frequency, becomes available to greater levels of precision, this can trigger a more innovative use of timing, and will further stimulate the development of new multimedia application scenarios.

4.2 Timing Synchronisation Techniques

This section first describes some general concepts relating to unidirectional and bidirectional time synchronisation as well as definitions of some clock terminology, such as skew and drift. It then provides operational details of a range of actual protocols used to distribute time synchronisation. The section concludes with a short section describing frequency synchronisation techniques.

4.2.1 General Concepts

If two computing systems, hereafter referred to as node A and node B, connected via a communication link wish to synchronise their clocks, then they must exchange some information regarding the state of their clocks. Thus, node A might transmit a message containing a timestamp to node B. Node B can then set its clock to the value of that timestamp. This scenario is illustrated in Fig. 4.2. Here, node A is referred to as the *reference* and node B is termed the *host*. This represents one of the most basic synchronisation techniques and is referred to as *unidirectional synchronisation* by Romer et al. [11]. Due to its simplicity, however, it is prone to multiple sources of synchronisation error.

First, node B does not accommodate for the time it takes the message to traverse the entire communication link between A and itself. This time is termed the *traversal time*. Thus, if node B sets its clock to the value received in the message, it will have a time error equivalent to the traversal time of the message. Node B could accommodate for the traversal time if it was known but, in many cases, it is nondeterministic. This is due to sources of nondeterministic latencies that exist along the communication path. They are categorised by Kopetz and Ochsenreiter [12] as the

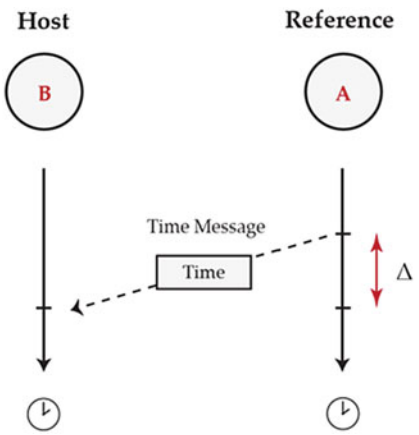


Fig. 4.2 Unidirectional synchronisation

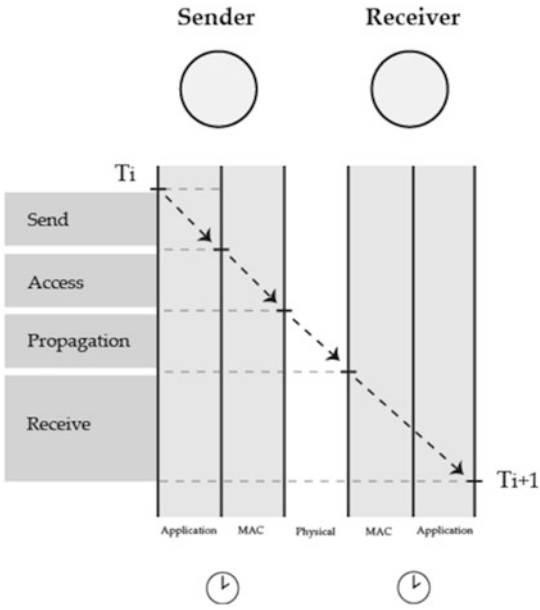


Fig. 4.3 Components of message latency

send time, *access time*, *propagation time* and *receive time* and are illustrated in Fig. 4.3.

Whilst all of these components may be nondeterministic, much of this nondeterminism can be eliminated by timestamping message transmission and reception events at lower levels in the communication hierarchy. In its most basic form, the unidirectional synchronisation technique is not effective at meeting the accuracy requirements of many time-sensitive applications. In fact, this technique is only appropriate if the accuracy requirements of the host are coarser than the maximum message latency between the host and reference.

A second more effective synchronisation technique is *round-trip synchronisation*, which is illustrated in Fig. 4.4. Here, node B, the host, initiates the communication exchange by transmitting a *request message* to node A, the reference node. Node B records the transmission time, T_i, of this request message. The message traverses the link and arrives at A at which point A records the reception time, T_{i+1}, of the message. Node A then constructs a *response message* that it transmits back to B. This process requires that A records the transmission time of the response message, T_{i+2}, and places the timestamps T_{i+1} and T_{i+2} into this message. Node B subsequently receives the response message, records its reception time T_{i+3}, and uses the four acquired timestamps to determine the *round-trip delay* (δ) of the message, using Eq. 4.1. By eliminating the processing time at A and subtracting the downlink delay from the uplink delay, an approximation for B's time error relative to A can be determined. This error represents the *phase offset* (θ) between A and B and is calculated using Eq. 4.2.

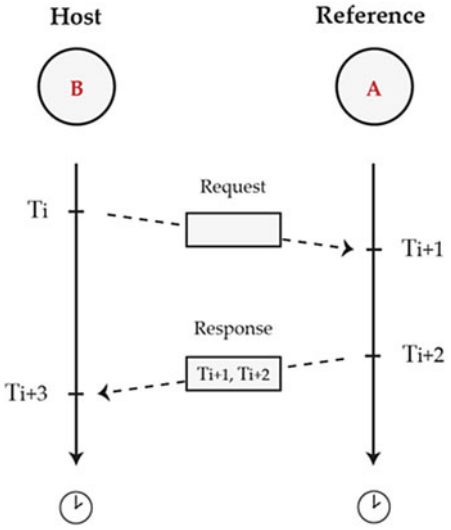


Fig. 4.4 Round-trip synchronisation

$$\delta = (T_{i+3} - T_i) - (T_{i+2} - T_{i+1}) \tag{4.1}$$

$$\theta = [(T_{I+1} - T_i) + (T_{i+2} - T_{i+3})]/2 \tag{4.2}$$

Round-trip synchronisation can, in certain circumstances, be an effective method for determining the traversal time of a message, thus, improving the estimate of a host's offset from its reference. Nevertheless, it makes one core assumption that turns out to be generally untrue in real-world scenarios, that is, the latency of a request message is equal to the latency of the corresponding response message over anything but the most trivial network path. The issue with this assumption is illustrated in Fig. 4.5. In the scenario depicted in Fig. 4.5, the latency of the response message exceeds that of the request message by a magnitude denoted by Δ. If the nodes are in fact synchronised, then this time difference results in an estimated offset error of $-\Delta/2$. Thus, although this technique outperforms unidirectional synchronisation, it is also subject to the negative effects of nondeterministic message latencies, although to a lesser extent. As we will see later, PTP defines a method whereby intermediate network nodes measure the actual delay experienced by these messages in each direction, effectively eliminating network jitter and delay asymmetry as sources of time synchronisation error.

4.2.2 Clock Skew and Drift

The synchronisation techniques detailed in the previous section provide a means for a host node to determine its phase offset from a reference node. If both nodes employ extremely accurate and stable oscillators, then it is sufficient to perform the

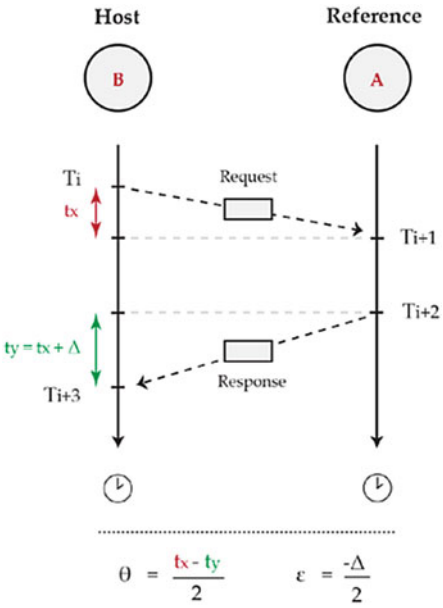


Fig. 4.5 Asymmetric delay

synchronisation process only once. In reality, however, clocks will have a frequency error that may change over time necessitating multiple synchronisation rounds.

A frequency error that exists between a host and a reference results in *clock skew* (λ). Clock skew is defined as the rate of change of a host's time with respect to a reference's time. A host may determine its clock skew relative to a reference by performing two synchronisation rounds. The skew is then calculated as the difference between the two estimated offset values (θ_i, θ_{i+1}) divided by the *synchronisation interval* (τ) (see Eq. 4.3). The synchronisation interval, τ, represents the time interval between two consecutive synchronisation rounds.

$$\lambda_i = (\theta_i - \theta_{i+1})/\tau \tag{4.3}$$

Whilst two synchronisation rounds is the minimum required to determine a host's skew, the accuracy of the estimated skew value depends on a number of factors. These include the time interval between synchronisation rounds and the magnitude of time error that is caused by nondeterministic message latencies. If the time interval between synchronisation rounds is relatively short, then message latency related errors typically dominate. This is generally true in the case of packet switched networks that span large geographical areas and are composed of multiple intermediary nodes that subject packets to varying latencies. If, however, the time interval between synchronisation rounds is too long, then the stability of a clock becomes a factor, since the clock's frequency offset may change within the interval and contribute to a greater proportion of the time error.

The quality of a skew estimate can be improved by analysing multiple offset estimates using a technique termed *least squares linear regression*. Linear regression provides a means to postulate a linear relationship between a host's time and reference's time. This relationship can be expressed using Eq. 4.4.

$$T_r = \theta + \lambda T_h \tag{4.4}$$

In Eq. 4.4, T_r denotes the time at the reference node and T_h denotes the time at the host node. The benefit of using linear regression with multiple data points is that it is more resistant to erroneous offset estimates. Consequently, the calculated skew value is generally more accurate.

The second derivative of a node's time with respect to a reference's time is referred to as *clock drift* (φ). Clock drift is the direct result of oscillator instability and represents the rate of change of a host's skew with respect to a reference's time. Clock drift can be determined with two or more skew estimates as shown in Eq. 4.5.

$$\varphi_i = (\lambda_i - \lambda_{i+1})/\tau \tag{4.5}$$

Oscillator frequency variations are the result of crystal ageing, oscillator circuitry noise, and numerous environmental factors such as temperature changes. Thus, with respect to synchronisation, an accurate value for a host's clock drift at a particular point in time can be difficult to determine. Consequently, most synchronisation algorithms operate on the assumption that clock drift remains constant between synchronisation rounds. Thus, by frequently updating the clock skew, the negative effects of clock drift can be indirectly alleviated. With an estimate of its clock's skew and knowledge of the maximum permitted time error, ε_{max}, a host can bound the synchronisation interval as expressed in Eq. 4.6.

$$\tau \leq \varepsilon_{max}/\lambda \tag{4.6}$$

4.3 Time Synchronisation Protocols

Time synchronisation techniques form the cornerstone of most time-sensitive applications, in particular, distributed network applications. As outlined in the previous subsections, such techniques aim to ensure that the timescale of a host adheres to some reference timescale. This reference scale can be sourced locally via highly precise clocks or remotely. In the latter case, network connectivity facilitates synchronisation. The potential accuracy provided by the latter technique is highly influenced by the characteristics of the host, the reference device and the communication link that connects them. The communication link generally proves to be the greatest source of error. Thus, some techniques employ numerous communication exchanges and statistical analysis to alleviate this problem. Nonetheless, the use of a time synchronisation technique alone is generally inadequate in terms of

meeting the accuracy requirements of time-sensitive applications. Thus, additional methods are required that work in unison with a particular synchronisation technique.

The combination of such methods and techniques typically form what is termed a *time synchronisation protocol*. Whilst time synchronisation techniques typically detail the sequence and order of communications between a host and reference, as well as the core data exchanged between them, specific message structure and other operational details are not specified. The actual implementation of a synchronisation technique in combination with the message structure, and additional operations and methods, form a time synchronisation protocol.

Time protocols are typically designed to target specific network types. For example, *NTP*, developed by Mills and standardised in [13], is designed to operate over large, dynamic and variable latency *packet switched networks* (*PSNs*). It does so effectively by employing sophisticated statistical techniques to minimise errors introduced by such networks. Conversely, *PTP,* standardised in [8], is designed for privately managed and well-controlled packet switched networks (PSN) that employ specialised hardware and, therefore, can provide much greater accuracies. Each protocol performs well in its targeted environment. The accuracy requirements of a time-sensitive application will ultimately dictate which protocol is employed and which network(s) it should be deployed over.

Both NTP and PTP are designed to operate within traditional PSNs. Such networks typically span large geographical areas and consist of devices that have considerable processing, memory and energy resources. Of course, there exists other types of networks where resources are limited and the geographical areas they span may be relatively small. One such type of network is a *Wireless Sensor Network* (*WSN*). Although a WSN's composite nodes are relatively powerful for their size, they pale in comparison to a typical PSN node. To elaborate, a typical WSN node's microcontroller operates in the low MHz range and may possess only several hundred kilobytes of short-term memory. In comparison, a typical PSN node's CPU will operate in the GHz range, and the node may possess several gigabytes of short-term memory. Consequently, alternative time protocols have emerged to deal with the limited capabilities of WSN nodes. These include *Reference Broadcast Synchronisation* (*RBS*), by Elson et al. [14]; the *Timing-sync Protocol for Sensor Networks* (*TPSN*), by Ganeriwal et al. [15] and, most notably, the *Flooding Time Synchronisation Protocol* (*FTSP*), by Maroti et al. [16]. Such protocols operate in a fashion so as to limit processor, memory and, more importantly, energy use whilst still providing a high degree of time accuracy.

The following sections detail the design of the most notable time synchronisation protocols mentioned above, namely, FTSP, NTP and PTP, and highlight the methods by which these protocols eliminate or mitigate potential sources of time error.

4.4 Flooding Time Synchronisation Protocol (FTSP)

The *Flooding Time Synchronisation Protocol* (*FTSP*), by Maroti et al. [16], is a good example of a protocol that employs the unidirectional synchronisation technique described in Sect. 4.2. It is currently the de facto time synchronisation protocol employed by WSNs. It comes bundled with TinyOS [17], a popular open source embedded operating system designed for low-power wireless devices. An analysis of the protocol gives one insight into how the protocol overcomes the limitations of the unidirectional synchronisation technique.

4.4.1 Overview

The FTSP protocol operates by organising a network into an ad hoc synchronisation tree within which an elected root acts as the source of time. The initial phase of FTSP involves the election of a root node. The root election process is based on a simple algorithm whereby the node with the lowest assigned ID is elected the root. The election of the root is followed by synchronisation rounds that are initiated by the root and occur at periodic intervals denoted by τ. The choice of τ is dictated by the accuracy requirements of a WSN application and the state of a WSN's operating environment. If an environment subjects a node's clock to drift then lower values of τ result in more accurate estimates for that node's offset. This is true because the node receives time messages more frequently, and this allows it to update its estimate of its clock skew more regularly.

As illustrated in Fig. 4.6, synchronisation messages contain four key fields: the *timestamp* field, the *rootID* field, the *nodeID* field and the *seqNum* field. The *timestamp* field represents the sender's notion of time at the point of transmission; the *rootID* field represents the address of the root as recognised by the sender; the *nodeID* field represents the address of the sender and the *seqNum* field holds a sequence number that is incremented solely by the root at the beginning of each synchronisation round.

A synchronisation round begins when the root broadcasts a synchronisation message. This message is received by all nodes within direct communication range of the root. The recipient nodes estimate their offset and skew and use these estimates to adjust their timescale. They subsequently broadcast their own messages and place adjusted timestamps within the *timestamp* field. Thus, the root's timescale is effectively flooded through the network. In order to manage redundant messages and ensure only the most recent messages are utilised by nodes, FTSP dictates that a node can only utilise the contents of a time message if it contains a lower *rootID* value or higher *seqNum* value than those received in the prior message. This mechanism, in addition to managing redundant messages, can also improve the synchronisation accuracy of those nodes located further away from the root. This is true because messages that arrive sooner encounter less delay en route through

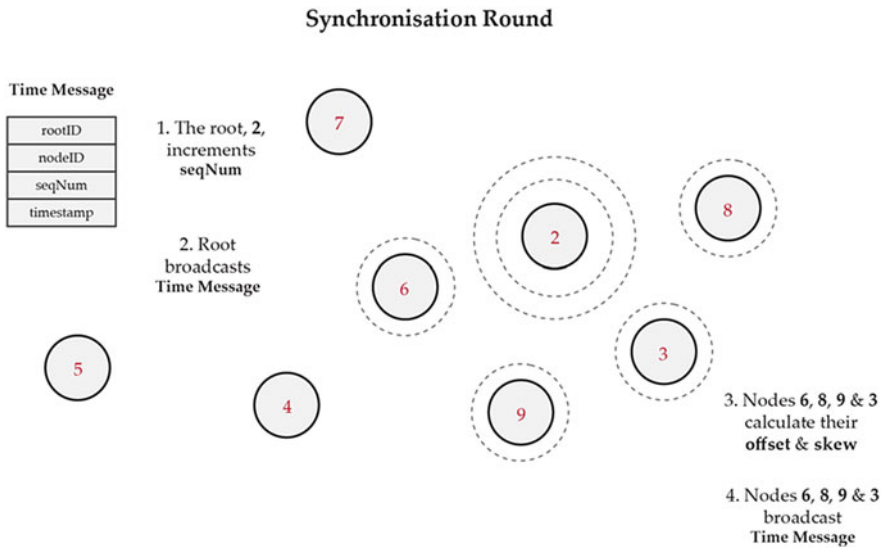


Fig. 4.6 FTSP operation

intermediate nodes and, therefore, more likely produce less error due to nondeterministic message latencies.

4.4.2 Synchronisation Errors

FTSP employs unidirectional synchronisation and, thus, is inherently susceptible to time errors that result from the propagation time of messages. This, however, as detailed by Maroti et al. [16], is shadowed by other larger delays that result from the MAC layer timestamping process. Traditional FTSP employs MAC layer timestamping, that, although an improvement on application layer timestamping, is still vulnerable to errors associated with nondeterministic message delays.

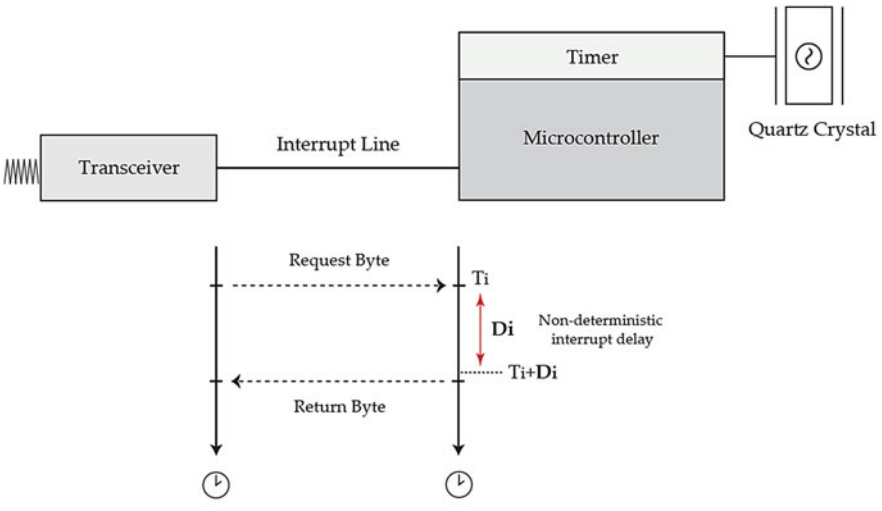
Maroti et al. [16] identify the greatest sources of error in communicating a message as the interrupt handling time and the encoding time. The FTSP method of reducing the errors associated with the process of transmitting a message entails the use of timestamps recorded at each byte boundary of a message. The assumption is made that a message is handed to a transceiver in a byte-orientated fashion. The transceiver signals that it is ready to receive subsequent bytes via interrupt requests. Each interrupt request is handled by passing the next byte of the message to the transceiver and generating a timestamp. An FTSP sender records these timestamps and normalises them by taking an appropriate multiple of the nominal byte transmission time from each one, that is, the time it takes to transmit all bytes up to and

including the byte associated with the timestamp. Thus, the transmission time is accounted for.

Interrupt delays vary and depend on the code being processed by the microcontroller when an interrupt is generated. Some code sections disable interrupts that increase this delay. FTSP deals with interrupt delays by using the minimum of the recorded timestamps. The minimum timestamp provides a basis to deduce the interrupt delay associated with all other timestamps, thus, allowing them to be corrected. Consequently, those errors associated with the interrupt handling time are eliminated with high probability.

Figure 4.7 illustrates this mechanism. In Fig. 4.7, the transceiver requests a byte from the microcontroller via an interrupt request at time T_i. The request is processed after a delay of D_i after which a timestamp is generated and a byte is returned to the transceiver. The value of D_i depends on the characteristics of the code being processed by the microcontroller at that time and, thus, varies for each request. In the table in Fig. 4.7, the timestamps for each byte associated with a four byte message are presented. Here, the request for the third byte results in the lowest interrupt delay of five ticks and, thus, produces the most accurate timestamp, T_3. Naturally, the identification of the minimum timestamp is performed after all timestamps have been normalised, since the value of D_i is unknown.

Ultimately, FTSP mitigates most of the errors associated with message delays with the exception of propagation delay, since it uses the unidirectional approach described in Sect. 4.2. Typically, propagation time contributes to a very small



Ti	T1	T2	T3	T4	. . .
Di	10	23	5	8	

Fig. 4.7 Interrupt delay

proportion of error (less than 1 μs for up to 300 m) whilst the interrupt handling time can be orders of magnitude higher.

In order to deal with clock skew, FTSP employs linear regression. Each node contains a regression table that holds the reference points related to the last N valid messages received. A node's skew value is updated each time a new valid message is received. Subsequently, the oldest reference point contained in the regression table is shifted out and the new one, associated with the new message, is shifted in. The protocol also dictates that a node may only broadcast synchronisation messages when it has at least M entries in its regression table. This rule ensures synchronisation stability throughout the network.

4.5 The Network Time Protocol (NTP)

The *Network Time Protocol* (*NTP*), by Mills [13, 18–21], provides a noteworthy example of a protocol that employs the round-trip synchronisation technique described in Sect. 4.2.

It is one of the oldest Internet protocols in operation today, having been in operation for over 25 years. It is designed to synchronise the clocks of nodes connected over dynamic PSNs that subject packets to varying latencies. Multiple paths may exist between two particular nodes and any of these paths may become congested, or fail, at any time. Thus, the route a stream of packets takes from one node to another may not remain fixed and, therefore, the latency of packets can vary over time.

In view of the fact that NTP employs round-trip synchronisation and this technique assumes that the packet latencies between two nodes are symmetric, NTP's design incorporates techniques to mitigate the effect of asymmetric delays. The protocol uses redundant time references together with a suite of statistical algorithms to achieve this. The use of redundant time references increases the diversity of paths and, thus, increases the likelihood of acquiring high quality time data, especially if there are no common network links to the different references. In addition to this, the identification and elimination of low-quality references are made possible. The statistical algorithms employed by NTP are used to filter the acquired data and choose the highest quality time references from which the final offset estimation is produced.

NTP hosts are organised into a hierarchical structure termed an NTP subnet. An example of an arbitrary subnet is illustrated in Fig. 4.8. Hosts at each level of the hierarchy are assigned a stratum number that represents their distance from the root of the hierarchy, which itself has a stratum of zero. Stratum 0 references consist of extremely accurate time sources, such as Global Navigation Satellite System (GNSS) clocks, e.g. GPS (Global Positioning System), radio and atomic clocks. Hosts that synchronise with stratum 0 references are classified as stratum 1 references, and those that synchronise with stratum 1 references are classified as stratum

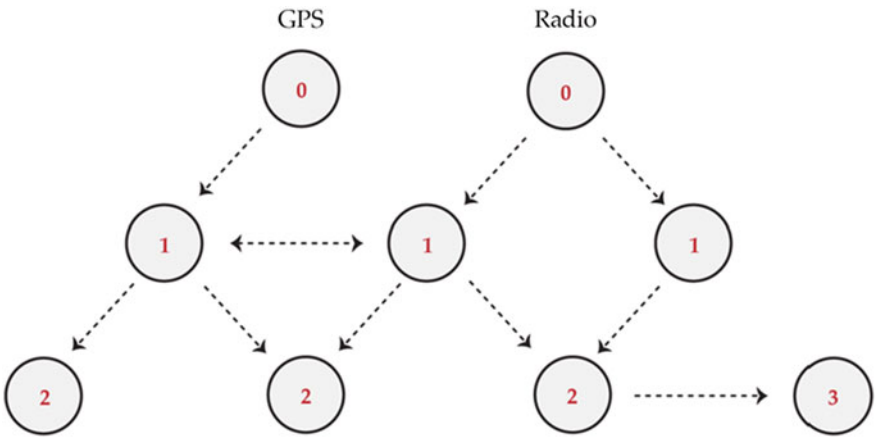


Fig. 4.8 NTP hierarchy (stratums)

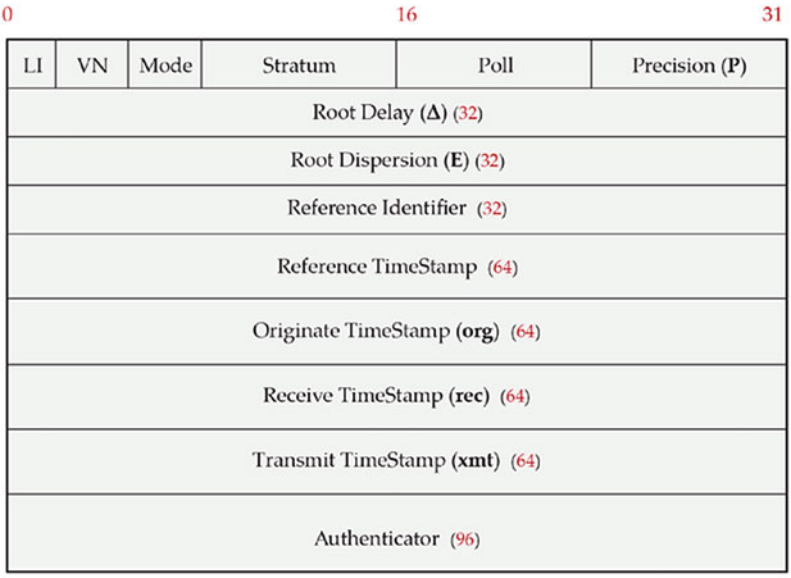
2 references and so on. This structure, in addition to eliminating cyclical dependencies, provides a means of comparing the quality of references.

4.5.1 Packet Structure and Processing

The NTP protocol operates at the application layer of the OSI (Open Systems Interconnection) model. It uses the transport layer protocol User Datagram Protocol (UDP) to distribute time. The NTP packet structure is illustrated in Fig. 4.9. The interpretation of the packet fields varies depending on the operating mode of the NTP hosts that are interacting. The primary operating mode is *client/server* mode, and it represents the most intuitive one, as it follows the classical *remote procedure call* (*RPC*) paradigm. The packet structure of an NTP reply packet sent from an NTP server in response to a request from an NTP client is also illustrated in Fig. 4.9. The four fields of note are as follows:

- *Reference Timestamp*—Holds the time the server last corrected its clock.
- *Originate Timestamp*—Holds the local time of the client the moment it sent the request packet (represents T_i at the client in Fig. 4.9).
- *Receive Timestamp*—Holds the local time of the server the moment it received the request packet (represents T_{i+1} at the server).
- *Transmit Timestamp*—Holds the local time of the server the moment it sent this reply packet (represents T_{i+2} at the server).

On receipt of a reply message from the server, the client uses Eqs. 4.1 and 4.2 to determine the packet's *round-trip delay* (*RTD*) (δ) and its *clock offset* (θ), respectively.



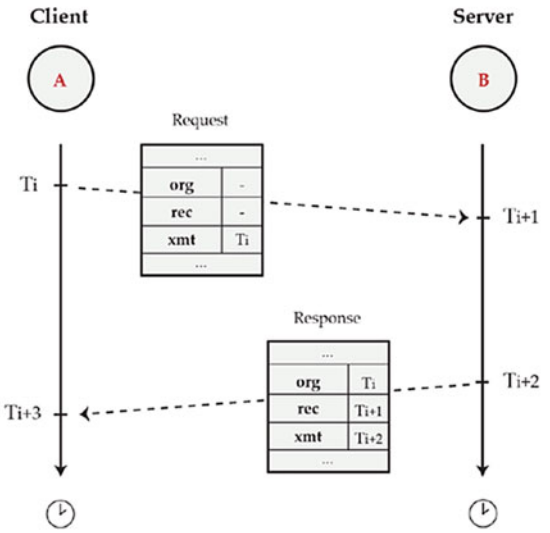


Fig. 4.9 NTP packet structure and exchange

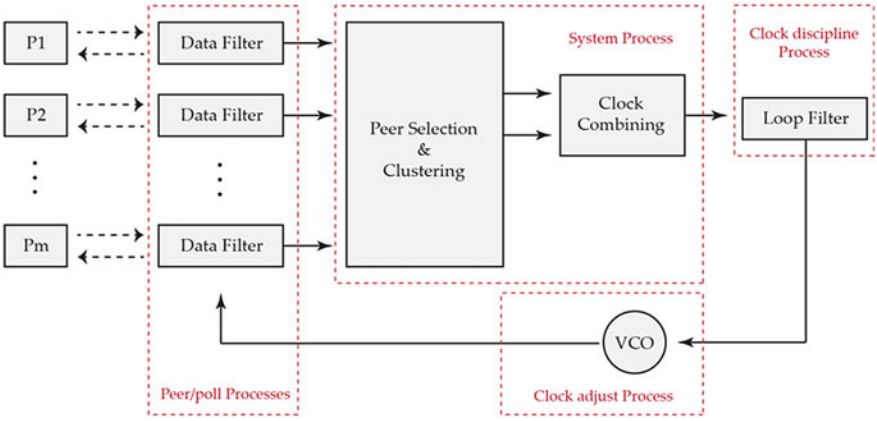


Fig. 4.10 NTP processes and flow

4.5.2 NTP Process Flow

In relation to the NTP daemon/service that resides on a host, its composite algorithms are organised into processes that interact with each other in a well-defined manner. The algorithms, their associated processes and their interactions are illustrated in Fig. 4.10.

The *peer/poll processes* are responsible for receiving and transmitting NTP packets from a host to a peer. The *peer process* is also responsible for processing packet contents that result in offset and round-trip delay estimates, collectively termed state variables. A peer and associated poll process are established for each peer. These processes together with their associated state variables form what is termed an *association*. An association is classified according to the duration of its lifetime that in turn is dependent on the operational mode of the peer and host. An association that exists indefinitely is classified as a *persistent* one, whereas one that exists briefly is classified as either *pre-emptable* or *ephemeral*. To clarify, in the case of a host and peer operating in *client/server mode*, the client establishes a persistent association for the server, since it must continually poll it in order to stay synchronised. In contrast, the server, on receipt of an NTP request packet, generates an ephemeral association that only exists until the request is fulfilled. This is so because a server does not need to maintain state regarding the condition of a client.

The *system process* and its associated *selection*, *clustering* and *combining* algorithms are responsible for pruning the data passed to it from one or more peer process(es). The selection algorithm's core responsibility is the identification and elimination of data associated with erroneous peers, also termed *falsetickers*. The clustering algorithm prunes the remaining data further by identifying the most accurate data and eliminating the remainder. If at this point data associated with more than one peer remains, then the final estimate of a host's offset is calculated using the combining algorithm. The final offset estimation is then passed to the

clock discipline process that together with the *clock adjust process* adjust the clock in small appropriate increments insuring a continuous monotonic clock, that is, a non-decreasing clock.

A basic of overview of these processes follows. A more detailed description of each process is presented by Mills in [20].

4.5.3 Data Filter Algorithm

The data filter algorithm represents a core component of the peer process. Its primary function is the identification and selection of an accurate offset and RTD pair sample (δ, θ) from a list of *n* contiguous samples associated with a particular peer.

The data filter algorithm's design is based on the characteristics of typical PSNs. Such networks are designed to alleviate packet congestion through the use of extra resources and sophisticated routing algorithms. Thus, whilst it is unlikely that a packet experiences significant delay in one direction, it is very unlikely that it also experiences a significant delay in the opposite direction. This can be verified by plotting a RTD (δ) versus offset (θ) scattergram for a particular peer. When the RTD (δ) and offset (θ) samples associated with a particular peer are plotted over time they form a scattergram, the shape of which highlights the characteristics of the path between a host and a particular peer. In general, it resembles a *wedge*, as illustrated in Fig. 4.11, and is termed a wedge scattergram by Mills [20].

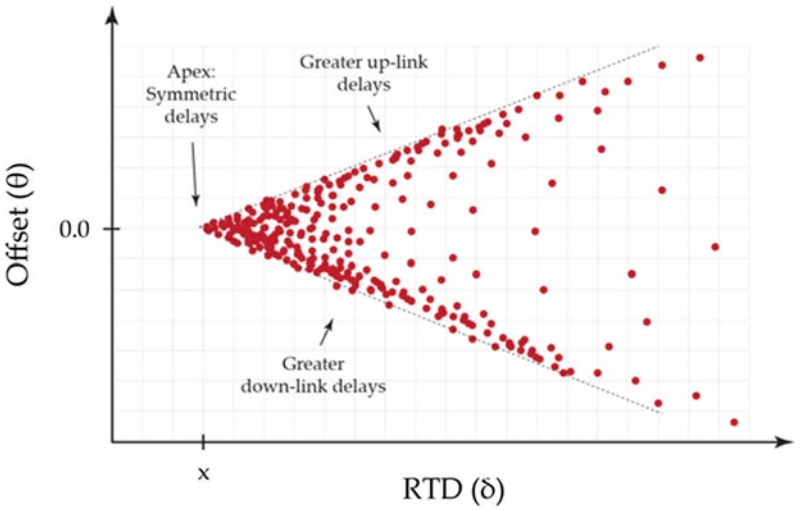


Fig. 4.11 NTP wedge scattergram [20]

With respect to Fig. 4.11, one may observe that the majority of points are located at the top, bottom and apex of the wedge, which highlights the aforementioned characteristics of PSNs. Those points located at the bottom of the wed indicate that the associated NTP request packets are subjected to greater delays than the corresponding reply packets. Correspondingly, those points located at the top of the wedge indicate that the associated NTP reply packets are subjected to greater delays than the corresponding request packets. Consequently, those points located at the apex of the wedge scattergram represent the most accurate estimates of the host's clock offset, as they are associated with symmetric network delays. Those points are also associated with the lowest RTDs, which suggest that when presented with a sequence of RTD and offset pair samples (δ, θ), the pair with the lowest RTD should be utilised, as it represents the most accurate data.

The data filter algorithm strives to determine such a sample and the final output of the algorithm is a tuple of the form $(\delta_m, \theta_m, \varphi_m, \varepsilon_m)$ that corresponds to the lowest RTD sample (i.e. δ_m and θ_m), the jitter, and dispersion, respectively, for a particular peer m. The *peer dispersion* (ε) and *peer jitter* (φ) represent important quality metrics that are used by subsequent algorithms to further prune and refine samples. The *peer dispersion* is initialised with the resolution of the host's clock and then increased at a constant rate to mimic skew. The *peer jitter* represents the *Root Mean Square* (*RMS*) of the differences between each of the offset samples associated with a particular peer and the most accurate offset sample of that peer (Fig. 4.12).

These variables are generated by each of the m peer processes associated with each of the m peers. They are subsequently referred to as *peer variables* and used by succeeding NTP algorithms.

4.5.4 Selection Algorithm

The goal of the selection algorithm is to identify inaccurate peers. It uses the *peer variables* associated with each peer to identify inaccurate peers.

A properly configured NTP client will employ numerous references/servers located across distinct network paths in order to synchronise its clock. The use of redundant references is beneficial in that data that originates from one or more incorrect server/s can be more easily identified assuming the majority of references are correct. The selection algorithm is responsible for separating incorrect peers, termed f*alsetickers*, from correct peers, termed *truechimers*. It achieves this by constructing confidence intervals for each of the m peers. Subsequently, it determines the smallest intersection interval within which at least m-f of the peers' associated offsets lie.

The confidence interval of a peer represents a range of values within which the true offset associated with a peer must be located. In relation to a peer m, this interval is equal in magnitude to twice the value of the *root distance*, Λ_m, associated with that peer, and the midpoint is equal to θ_m. This interval is represented by the

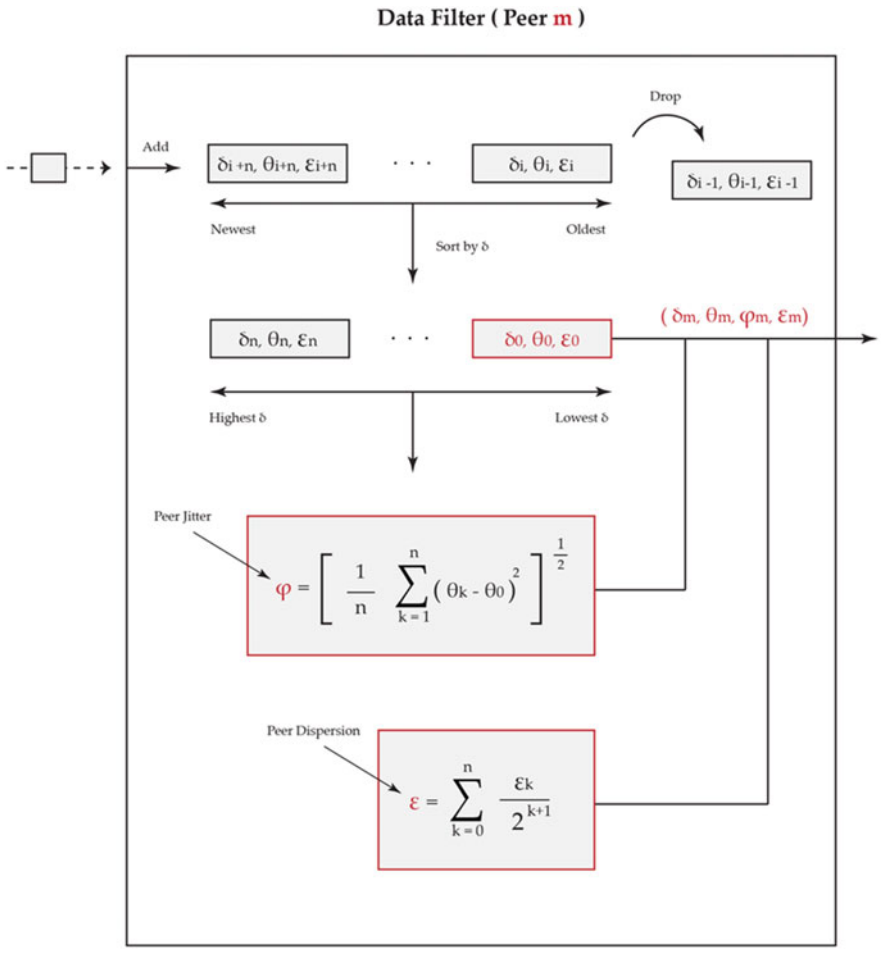


Fig. 4.12 Data filter algorithm

expression $[\theta_m - \Lambda_m, \theta_m + \Lambda_m]$. The root distance Λ associated with a peer represents the maximum time error of that peer relative to the root of the hierarchy.

Since the confidence interval bounds the true offset of a host relative to a peer, the confidence interval of a particular peer that does not intersect with the majority of the intervals associated with the remaining peers is most likely a falseticker.

In the case of m peers where up to f falsetickers are permitted and $f < m/2$, the NTP selection algorithm identifies the smallest intersection of m-f confidence intervals that contains at least m-f midpoints. This intersection interval is bounded by the lower limit, l, and the upper limit, u, and can be represented by the expression $[l, u]$. This is illustrated in Fig. 4.13, which depicts the confidence intervals associated with four peers. The selection algorithm produces an intersection interval that contains the midpoints θ_0, θ_1, and θ_2. The data associated with

peer *3* lies outside this interval, and so the peer is considered a falseticker and removed from the group.

4.5.5 Clustering Algorithm

The goal of the clustering algorithm is to identify the most accurate peer/s amongst the remaining peers.

In order for the clustering algorithm to identify this particular peer, it must use appropriate quality metrics. The first of these metrics is the *peer jitter* (φ), which was explained above. In essence, it represents the magnitude of variation of the offset samples associated with a particular peer and, therefore, is a good measure of the quality of data associated with that peer. The second metric utilised is the *selection jitter* (φ_s), which represents the RMS of the differences between a particular peer's associated offset and the remaining peers' associated offsets. The final metrics employed are the *stratum* (s) and *root distance* (Λ) of the peer. These metrics are used to produce a sort metric denoted by the symbol λ. The sort metric for a particular peer m is calculated using Eq. 4.7, where Λ_{max} represents a bias factor termed the *maximum distance*.

$$\lambda_m = \Lambda_{max} s_m + \Lambda_m \tag{4.7}$$

Equation 4.6 is designed to give preference to those peers with lower stratum numbers. This is logical, since those peers with lower stratum numbers are typically

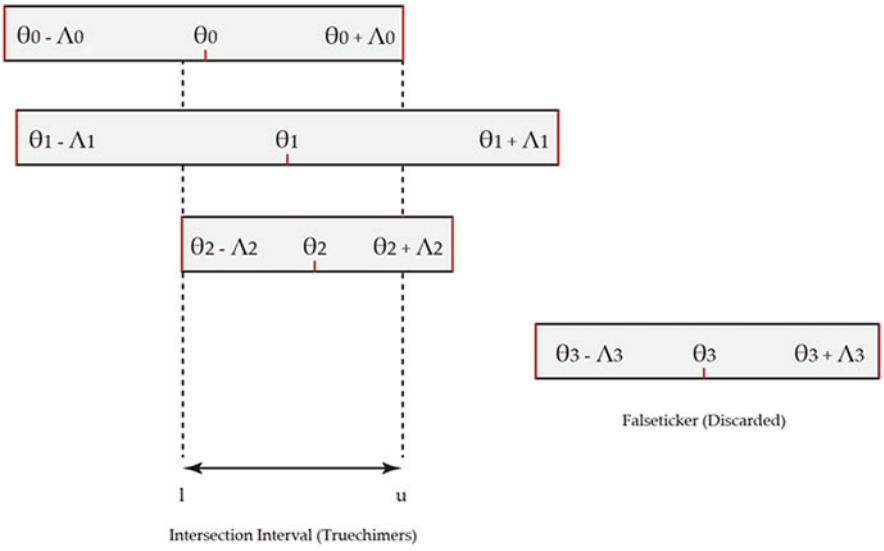


Fig. 4.13 NTP selection algorithm

more accurate than those with higher ones, given the structure of the NTP hierarchy. Of course, this is not always the case, and so the root distance, Λ, of a peer is also utilised to handle such scenarios. Thus, in the case of two peers i and j with stratum numbers *1* and *2* respectively, peer j will not precede peer i unless i's root distance is greater than the maximum distance, Λ_{max}, and j's root distance, Λ_j, combined.

The details of the algorithm are outside the scope of this work, suffice to say that the three aforementioned metrics, that is, the peer jitter, selection jitter and sort metric are used to prune the collection of peers until the most accurate peer/s is/are identified.

4.5.6 Combining Algorithm and Clock Discipline Algorithm

In the case of multiple surviving peers, the combining algorithm is employed. The combining algorithm is based on an observation that it is possible to obtain a more accurate estimate of a host's offset, if the offset of multiple surviving peers are averaged according to a suitable weighing scheme. This weighing scheme is based on the *root distance* (Λ) of a peer whereby those peers with lower values of Λ contribute more to the final estimation of the offset Θ. The final offset, Θ, is subsequently used by the clock discipline algorithm, which forms the heart of the clock discipline process. It operates as a feedback control system that uses the offset value produced by preceding algorithms to calculate phase and frequency corrections that are subsequently used to control a *Variable Frequency Oscillator* (*VFO*).

4.5.7 Asymmetry Corrections

In relation to the previous outline of the various techniques and algorithms employed by NTP, it should be clear that the protocol is designed around the assumption that asymmetric round-trip delays are typical of the networks it is deployed on. As such, the suite of algorithms at the core of the protocol strives to mitigate the errors that might result from asymmetric round-trip delays. This strategy, however, only proves effective if NTP is configured to employ multiple time references that are connected via distinct paths. If too few references are employed or multiple references share a common path, then NTP's performance may be degraded quite significantly. Thus, whilst NTP is quite an effective time synchronisation solution, its performance is highly dependent on the way it is configured.

4.6 Precision Time Protocol (PTP)

Another notable example of a time protocol that employs the round-trip synchronisation technique described in Sect. 4.2 is the *Precision Time Protocol* (*PTP*). Before discussing PTP, it is worth outlining the reasons for its development since its alternative, the well-established NTP, appears to be an effective time protocol in and of itself. NTP is designed to meet the accuracy requirements of distributed applications that are connected over wide area networks. NTP provides a sufficient degree of synchronisation to meet the requirements of a majority of these applications. When accuracies as low as 1 ms are required, NTP may also be employed but, ideally, the link that connects an NTP server to the devices that require synchronisation should not introduce delays that might negatively impact the protocol's performance. Thus, the link should be a managed one.

There are other applications that require synchronisation levels far beyond what NTP can deliver. The IEEE 1588 standard, otherwise known as the *PTP* [8, 22, 23], is designed to fill the niche that alternative synchronisation protocols cannot, by providing a feasible means of time synchronising numerous interconnected computing devices to each other over a network to within microseconds of each other.

4.6.1 Overview

PTP was designed for systems that require microsecond to sub-microsecond accuracies. To date, such systems can be found mostly in industrial, utility and communication sectors.

In contrast to NTP and with regard to the accuracies it is designed to achieve, PTP makes some core assumptions about the state of the network it operates over. The first of these is associated with asymmetry. Similar to NTP, the protocol, at its core, employs a variant of the round-trip synchronisation technique and, thus, asymmetric two-way packet delays can impact its performance. Thus, PTP expects that the underlying network is managed and, thus, assumes that the network and its components are selected and configured to minimise asymmetry. In addition to this, PTP expects that the traffic patterns on the network are controlled, so that traffic variation and, consequently, timing jitter, are minimised. Ideally, the network should be configured such that PTP messages are prioritised and, in addition, where possible, internetworking devices should be replaced by PTP-aware devices, such as *transparent clocks* and *boundary clocks*.

The PTP protocol itself is a distributed one that self-organises nodes into a *master–slave hierarchy*, the root of which is termed the *grandmaster clock*. The grandmaster clock acts as the time reference for every clock within a PTP domain. The network elements that participate in the PTP synchronisation process are categorised into one of five PTP device types: ordinary clocks, boundary clocks, end-to-end transparent clocks, peer-to-peer transparent clocks and management

nodes. Each PTP device type implements various features of the protocol. The role that a network element plays within a network typically dictates the type of PTP device it becomes within the PTP synchronisation hierarchy.

The devices in a PTP system interface with the network via ports. When the protocol constructs the synchronisation hierarchy, it places each port on each PTP device into one of three primary states: *slave*, *master* or *passive*. Thus, although a device typically contains a single physical clock, it is not the device itself that enters the slave, master or passive state but its ports, each of which is associated with a distinct *protocol engine* and a dataset that describes it. During the synchronisation process, a slave port determines its clock offset with respect to its associated master port using two-way PTP message exchanges.

4.6.2 PTP Synchronisation

Each slave port in a PTP device communicates with a master port on another PTP device via an independent communication link. This communication link may or may not contain other network elements. If it does, then these elements may introduce varying message latencies. If these elements are 'PTP aware', then they possess specialised PTP hardware/software that allows them to measure and then correct for any message latencies they are responsible for.

The goal of a port in the slave state is to determine the propagation delay of PTP messages that traverse the link between it and its associated master port. Knowledge of this propagation delay permits the port to determine the offset between its local clock and its master's local clock. To accomplish this, the standard describes two mechanisms that can be used, namely the *delay request–response mechanism* and the *peer delay mechanism*. Devices that employ the delay request–response mechanism use a variant of the round-trip synchronisation technique in order to acquire the four timestamps $(T_1 - T_4)$ necessary to calculate their clock offset. Devices that employ the peer delay mechanism use a separate sequence of PTP messages in order to determine one-way link delays at each port (explained later). Using this mechanism, timestamps recorded at either end of the link can be corrected using the known link delay, thus, eliminating the need for a slave port to acquire all four timestamps $(T_1 - T_4)$.

4.6.3 Link Propagation Delay

PTP devices that employ the *peer delay mechanism* can determine the propagation delay of its ports' associated communication links, where a communication link represents a connection between two distinct PTP ports. Every port on a PTP device that employs the mechanism uses it to determine a delay measurement for its associated link. Ports that share the same link, such as a master port on one device

and a slave port on another, use the mechanism to calculate their own independent value for the link propagation delay.

The mechanism itself employs two core PTP message types termed the *Pdelay_Req* and *Pdelay_Resp* messages, and in cases where particular hardware is not available, the *Pdelay_Resp_Follow_Up* message. The *Pdelay_Req* and *Pdelay_Resp* messages fall into a category of messages termed *event messages* that trigger the generation of timestamps at transmission and reception. Thus, the purpose of a PTP event message is to capture time information. The *Pdelay_Resp_Follow_Up* message falls into another category of messages termed *general messages,* which are used to communicate information, which in this case, is timestamp information.

The link delay measurement process is illustrated in Fig. 4.14. A port that wishes to determine the link propagation delay between it and another port that shares the same link, termed its *link peer*, initiates the communication exchange by transmitting a *Pdelay_Req* event message via the event interface. In Fig. 4.14, port 1 and port 2 represent two ports that share a link. Port 1 wishes to determine the link delay between it and port 2 and, thus, initiates the communication exchange. This transmission triggers the generation of a timestamp denoted T_1 that represents the transmission time at port 1. The destination port, port 2, on receipt of the *Pdelay_Req* message via the event interface generates the timestamp T_2 that represents the reception time of the *Pdelay_Req* message. In response, port 2 transmits a *Pdelay_Resp* event message back to port 1. This generates the transmission timestamp T_3 at port 2 and the reception timestamp T_4 at port 1.

In the particular case depicted in Fig. 4.14, at this stage in the process, port 1 only has access to timestamps T_1 and T_4. It must acquire either T_2 and T_3, or the difference between T_2 and T_3 in order to calculate the link delay. The acquisition of this information can be accomplished in one of the three ways, depending on the capabilities of the hardware:

- The first approach, which is illustrated in Fig. 4.14, has port 2 communicate the difference between T_2 and T_3 back to port 1 in a *Pdelay_Resp_Follow_Up* message.
- The second approach has port 2 communicate the difference between T_2 and T_3 back to port 1 in the *Pdelay_Resp* event message.
- The third approach has port 2 return timestamp T_2 in the *Pdelay_Resp* event message and timestamp T_3 in a *Pdelay_Resp_Follow_Up message*.

Once port 1 has acquired all four timestamps, it can calculate the link delay and, subsequently, correct timestamps acquired using the synchronisation process (next section).

It must be noted that the peer delay mechanism requires that a single *Pdelay_Req* message is received by and responded to by a single port—which is typically ensured by operating Pdelay messages over a point-to-point links, such as full-duplex Ethernet.

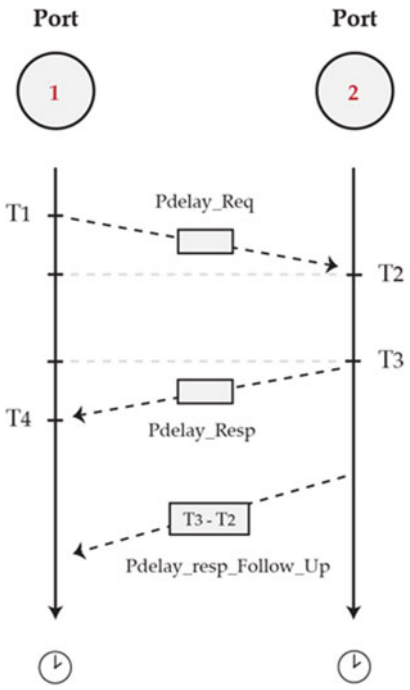


Fig. 4.14 PTP peer delay mechanism

Other less severe errors can be introduced if the frequency of port 1's associated clock differs substantially from that of port 2's associated clock. In order to reduce such errors, port 2 should respond to a *Pdelay_Req* event message with a corresponding *Pdelay_Resp* message as quickly as possible after its reception.

4.6.4 Synchronisation Process

The typical PTP synchronisation process is illustrated in Fig. 4.15. The process itself employs messages termed *Sync*, *Delay_Req* and *Delay_Resp* messages. In cases where particular hardware capabilities are not available, a message termed a *Follow_Up* message is employed. Each port in a PTP device uses the process independently of other ports and, thus, is responsible for constructing and communicating its own PTP messages. *Sync* and *Delay_Req* messages, which belong to the category of event messages, are used to generate the necessary timestamps. *Delay_Resp* and *Follow_Up* messages, which belong to the category of general messages, are used to communicate timestamps recorded by a master port back to a slave port.

The synchronisation process is illustrated in Fig. 4.15. The master port initiates the synchronisation process by transmitting a *Sync* message to the slave. This message is

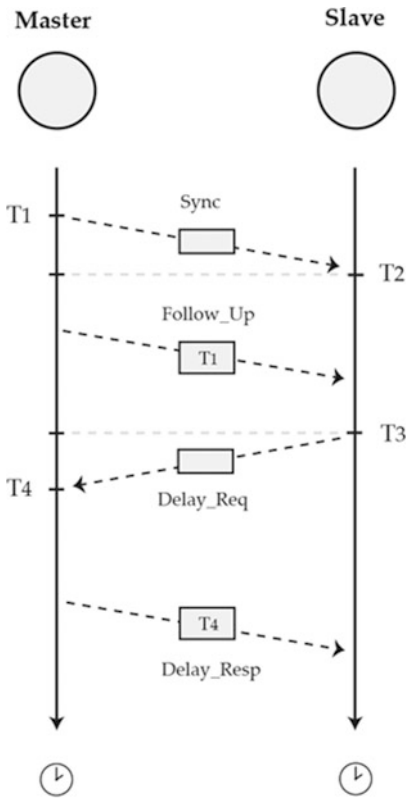


Fig. 4.15 PTP synchronisation process

transmitted via the event interface that generates the timestamp T_1. The *Sync* message is received via the event interface at the slave triggering the generation of the timestamp T_2. The slave must have access to T_1 in order to accurately calculate its local clock's offset. The acquisition of T_1 can be accomplished in one of the two ways:

- The timestamp T_1 can be placed into the *Sync* message before transmission by the master, known as one-step operation. This requires specialised hardware.
- A *Follow_up* general message can be used by the master to communicate T_1 to the slave (as illustrated in Fig. 4.15), known as two-step operation.

If the slave and master employ the peer delay mechanism then at this stage in the process, the slave has enough information to determine its clock offset. Hence, it will use the link propagation delay measurements it acquired via the peer delay mechanism to correct T_2 and, thus, calculate the clock offset.

If the slave and master employ the delay request–response mechanism, then the slave must acquire more time information. The slave transmits a *Delay_Req* event message that triggers the generation of timestamp T_3. The master on receipt of the *Delay_Req* message generates the timestamp T_4. This timestamp must be

communicated back to the slave and is done so via a *Delay_Resp* message. At this point, the slave has acquired the four timestamps ($T_1 - T_4$) necessary for it to compute its clock offset.

4.6.5 Asymmetry Corrections

In relation to the previous outline of both the PTP *peer delay mechanism* and *request–response mechanism,* it must be noted that these mechanisms rely on the assumption that delays between communicating ports are symmetric. Any asymmetry will produce incorrect offset estimations. Naturally, a communication link between a master and slave can have various network elements that may introduce such asymmetry. These elements, however, can be configured to proactively eliminate such asymmetry. For example, 'PTP-aware' devices, such as *end-to-end transparent* clocks and *peer-to-peer transparent* clocks, proactively correct for delays that they directly subject PTP event messages to. In essence, this is accomplished by measuring the residence time of PTP event messages as they traverse the various communication layers of the device and including this within the message.

In addition to errors resulting from asymmetry, other less severe errors can be introduced if the frequency of two communicating port's associated clocks differ. Such errors can be mitigated if interacting ports respond promptly to each other's requests. However, a more effective method involves clock synthesis. Clock synthesis in this context refers to the process of synchronising the frequency of one clock relative to another. The PTP standard describes a method to allow PTP devices, such as transparent clocks, to discipline their local clocks to match the rate of their master clock. This is accomplished by having the transparent clock analyse the timestamps sent by its master. By comparing the rate of change of its local time with respect to its master's local time, the ratio of one rate to the other can be estimated. This ratio of rates can then be used to adjust the frequency of the transparent clock's local clock (directly or via timestamp adjustments), thus, synthesising it to the master's clock. This whole process operates closed loop, since the adjusted clock's timescale is used in subsequent estimations.

The final and most obvious factor that has the greatest impact with regard to clock error is the point within the communication hierarchy that timestamps are generated. The generation of a timestamp occurs when a specific point in an event message crosses a particular boundary in one of the communication layers. This point is termed the *timestamp point*. The standard itself specifies a timestamp point in an event message for each of the communication layers. In order to achieve the maximum precision, timestamps should be recorded at the physical layer.

4.7 Frequency Synchronisation Over Packet Networks

As outlined earlier, frequency synchronisation (or syntonisation) of media clocks between multimedia endpoints ensures that the rate with which media is produced and consumed is equal, thus avoiding buffer overflow/underflow problems. In broadcast networks with fixed latencies, frequency synchronisation is easily achieved through use of frequency/phase lock loops. Over best-effort IP packet networks where latencies are nondeterministic, frequency synchronisation is much more challenging and techniques such as adaptive clock recovery are needed. Conventional Ethernet networks that are ubiquitous in today's infrastructure deploy a physical layer system whereby the receiving clock on the ingress interface that receives incoming bits and the transmitting clock that transmits bits on the egress interface are independently operating to within ±100 ppm, thus there is a lack of frequency synchronisation in the network. The protocol is thus designed at physical layer to cope with such potential frequency differences. For domains and applications that require frequency to be distributed, such as telecommunications networks, Synchronous Ethernet (known as SynchE) has been developed. Essentially, this physical layer mechanism locks the egress clock to the ingress clock. Thus, the source frequency is propagated through each hop, essentially locking the whole network together. This benefit is realised however at significant expense as it requires hardware upgrades and thus far, SynchE is typically deployed only in utility telecommunications networks. SynchE is standardised by the ITU-T, along with IEEE in a range of recommendations [24].

4.8 Time-Aware Networks: IEEE Time-Sensitive Networking (TSN) and Audio–Video Bridging (AVB) Standards

As outlined in the introduction, there has been some significant work to tackle some of the many challenges to making the full ICT infrastructure more time-aware. In this section, we examine recent developments in the standards world that aim to improve the degree of time awareness in networks.

Ethernet (defined formally by IEEE 802.3 [25]) was originally developed to operate over shared media where collisions demanded the retransmission of data by the Ethernet transmitter. Ethernet was enhanced in the 1990s, in collaboration with the bridging group (IEEE 802.1), to add support for full-duplex, store-and-forward, switched LANs, where Ethernet collisions were entirely eliminated and the theoretical link utilisation could be routinely achieved in both directions simultaneously. In addition to enormous increases in Ethernet PHY rates (100 Mbps, 1 Gbps, 10 Gbps, 100 Gbps, etc.), further enhancements defined specifically for real-time media streams were recently published. Real-time traffic could previously be prioritised by bridges and routers, but network delays due to congestion could neither

be determined beforehand nor controlled during operation. As these amendments to the IEEE 802 standards were created by the 802.1 TSN (formerly the AVB or Audio–Video Bridging) Task Group [26], they are often referred to collectively as the TSN (formerly the AVB) standards. Note that IEEE 802 standards are available for free 6 months after publication through the *IEEE Get* program at http://standards.ieee.org/about/get.

4.8.1 Time Synchronisation

The first addition to standard Ethernet required to meet the demands of high quality media networking was accurate time synchronisation. As outlined earlier, when two or more networked devices sample or render audio, if the timing relationship between them is not precisely controlled, the resulting audio channels can drift apart or exhibit modulation that negatively affects the fidelity of the sound.

The TSN standard that addresses the time synchronisation problem is IEEE Std. 802.1AS™ [26], a profile of PTP. In addition to specifying features for configuration-free accuracy assurance and fast reconfiguration, 802.1AS specifies the use of timing measurement protocol of IEEE Std. 802.11™-2012 (i.e. Wi-Fi)—an important consideration for consumer media. Predictable time accuracy is achieved by requiring each switch/bridge/router/Wi-Fi Access Point to participate explicitly in the transfer of time. It also adds explicit detection of networking equipment that does not support the capability, routing time instead on another path if available.

Typically, accuracy of time delivered to the network interface of each timing slave is more accurate than one microsecond, allowing even microphone array beamforming applications to be distributed across multiple distinct networked devices. As another extension beyond the default 1588 profile, if the Grand Master (GM) as defined in Sect. 4.6 disappears, the new GM communicates its frequency offset and time offset (if any) with respect to the previous GM, allowing slave devices to quickly lock their media or other application-specific clock.

Wi-Fi, being of particular interest for media synchronisation due to its prevalence in consumer markets, deserves some additional explanation here. In Fig. 4.16 we illustrate the method of passing of time between Station A and Station B, together with various timestamp measurement errors that occur within the Wi-Fi devices and within the wireless channel. The existence and correction of asymmetries within each device are accommodated in a manner identical to those with 802.3/Ethernet. Essentially, differences between Tx and Rx timestamps within each system are compensated for, with any remaining timestamp uncertainty contributing to the overall time error budget. Immunity to transmission rate and frame length is achieved by defining the packet timestamp point at the beginning of the frame.

Uncertainty within the wireless channel is due primarily to multipath effects of the signal propagation and reflections. The error in propagated time caused by asymmetry in the wireless channel (or timestamp asymmetry) is equal to one half of

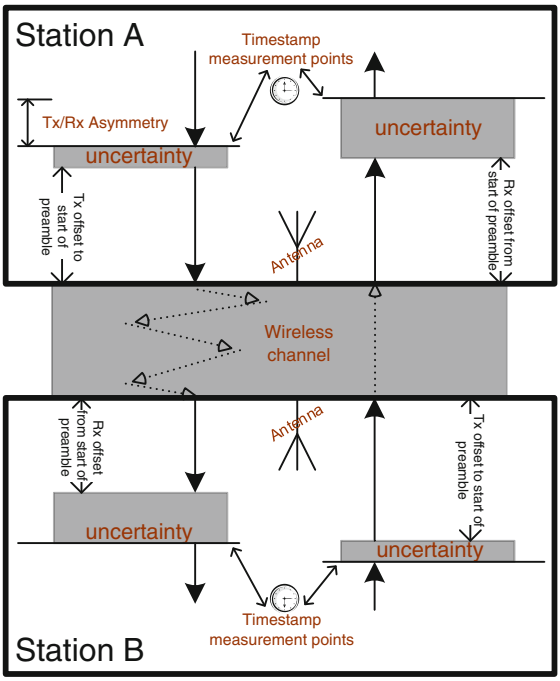


Fig. 4.16 Handling asymmetries for 802.1AS-based synchronisation over 802.11/ Wi-Fi links

the asymmetry. As an example, if signal reflections result in an asymmetry of 30 m, the resulting time error is 0.5 * (30 * 3.33 × 10^{-9}) s = 500 ns. Additional work in 802.11-RevMC [27] is underway to detect the line-of-sight arrival of the signal, in order to achieve absolute time accuracies of the order of a few nanoseconds for the purpose of indoor location [28].

Enhancements to 802.1AS provide GM redundancy and also path redundancy for the timing packets, so that the loss of a single link or a single GM does not result in loss of timing. Also, each GM may provide two or more independent time sources simultaneously—an important capability for industrial applications which use a very stable local time source for machine operation and another, synchronised to UTC, for workflow coordination and event timestamping.

4.8.2 Ensuring Timely Delivery

Together with accurate time synchronisation, real-time media streaming networks should ensure that media packets arrive in time for the contents to be processed and rendered. In networks with sufficiently light traffic and well-behaved endpoints, packet prioritisation can be sufficient to achieve timely delivery in most cases. Modern networks provide substantially better service for time-sensitive media (and other) streams through additional enhancements beyond time synchronisation from

802.1. These temporal Quality of Service (QoS) enhancements fall into two categories:

1. Admission Control: Signalling protocols that allow endpoints to request admission to a class of service that receives special treatment for a time-sensitive stream.
2. Traffic Shaping: Algorithms for networking equipment that deliver the temporal QoS for admitted streams.

Admission Control and Scheduling Protocols

Any general network that provides guaranteed service must have the ability to perform admission control—to reject requests once some threshold has been reached or if the request would cause network resources to be exceeded. One such distributed, peer-to-peer reservation protocol deployed in AV networks today is the Stream Reservation Protocol (SRP) published in IEEE Std. 802.1Q™-2015 [29].

A talker initiates an end-to-end SRP reservation across the network by advertising a stream with certain characteristics, including frames per second, maximum frame size, etc. Each bridge/switch in the LAN takes note of the characteristics of the reservation request and passes it downstream through the network. Upon receiving such an advertisement, one or more listeners may choose to subscribe to that stream by sending a listener ready message. As the listener_ready information propagates back towards the talker, each bridge has the option of declining to admit the stream due to bandwidth or other resource constraints. In this way, a multicast forwarding tree is created in each bridge automatically, and the listener(s) and talker are eventually informed of the outcome of the reservation request. Admitted streams then receive special treatment in the network devices, resulting in timely delivery of media packets.

Centralised SDN-based network provisioning techniques directly discover and configure the forwarding database and traffic shaping parameters in each device along the path between talkers and listeners to achieve the same end-to-end guarantees (instead of the SRP protocol). Support for this is being standardised in IEEE p802.1Qcc [29].

Latency Control Methods

The data rate of media streams is typically more regular than other web traffic. When low worst-case latency streaming of media is required, retransmission of data is not feasible. Now that Ethernet is full-duplex, collisions are eliminated. Thus, network congestion in the Ethernet switch/bridge/router becomes the dominant source of packet loss. Once a stream reservation is successful, the network applies special transmit selection algorithms to the admitted stream at each network device between talker and listener. The first such transmit selection standard from 802.1 is Forwarding and Queuing Time-Sensitive Streams (FQTSS), standardised in IEEE Std. 802.1Q™-2015. FQTSS eliminates congestion of admitted isochronous

streams by pacing admitted streams using the aggregate bandwidth admitted on each transmitting port.

Additional tools for improving worst-case latency include pre-emption of non-time-sensitive frames and creation of a global schedule for time-sensitive streams. Pre-emption (IEEE Std. 802.1Qbu™/IEEE Std. 802.3Qbr™) [27/23] eliminates the necessity for a time-sensitive frame to wait for a non-time-sensitive frame. Specifically, the best-effort frame currently being transmitted is pre-empted, the time-sensitive frame to be inserted and the best-effort frame to be resumed. One level of pre-emption is supported, but a best-effort frame may be pre-empted multiple times by multiple time-sensitive frames.

A global schedule for time-sensitive frames is akin to a schedule applied to traffic lights such that each subsequent traffic light turns green as an emergency (time-sensitive) vehicle approaches. The time-scheduled traffic shaper is defined in IEEE Std. 802.1Qbv™ [24]. With this approach, endpoints first make an admission request, including temporal requirements, typically to a central controller. Together with centralised admission control, this standard allows a high-level network scheduler to compute and download a transmit schedule for each link in a TSN-capable network. Thus, if the talker injects the time-sensitive packet within the right time window, it flows through the network without any head-of-line blocking from best-effort frames, just as a set of locomotives and their cars could theoretically traverse a continent without stopping, so long as all tracks are properly scheduled and potentially interfering trains held back.

TSN Robustness Features

Any assurance of low worst-case latency requires the talkers injecting time-sensitive traffic to be well behaved. If things go wrong, e.g. due to a malfunction of a talker, excessive traffic marked as time-sensitive (i.e. exceeding the amount of admitted traffic) might disrupt other time-sensitive flows. To provide immunity against such failures, 'Ingress Policing', defined in 802.1Qci [29], allows detection of bad behaviour by comparing the traffic injected by each talker to the amount of traffic admitted, punishing the misbehaving talker by dropping the excess frames.

Even with all of these features, random bit errors due to cosmic and other events can cause a frame to be corrupted and subsequently dropped. For situations where such an error would be catastrophic, a network-wide redundancy capability is defined in 802.1CB [29]. This works by replicating time-critical traffic (perhaps multiple times) and sent along maximally disjoint and redundant paths between the talker and listener such that delivery is immune to single points of failure of the network. Duplicate frames are then eliminated where redundant paths join together.

Time-Sensitive Media Streaming Protocols

The above capabilities provide robust transport of media and other time-sensitive data. They do so in a way that is independent of the protocol used for encapsulating media or other time-sensitive traffic for transmission from the talker to the listener. Several media streaming protocols are used to carry the audio samples and video

frames in a way that can take advantage of the aforementioned 802.1 synchronisation and timeliness standards.

Whilst also able to support best-effort networks, the AES67 standard [7] defines interoperability requirements for use of these capabilities by RTP streams, including a direct one-to-one mapping of the PTP clock to the media clock and an RTP header extension. The IEEE 1733 standard, which addressed these requirements, did so using a new RTCP payload format, but has been deprecated.

The Audio–Video Transport Protocol (AVTP), defined in IEEE Std. 1722™-2016 [9], encapsulates various media and other payloads and provides some useful in-band information, such as whether the PTP GM is available. This standard defines the encapsulation of International Electrotechnical Commission (IEC) 61883 (www.iec.ch/) and other newer media formats typically into Ethernet frames and provides for an indirection between the PTP clock and the media clock.

Additional specifications for TSN-based systems are available from Avnu Alliance (http://avnu.org/) which certifies implementations for interoperability and conformance with the TSN/AVB and related standards—similar to the role the Wi-Fi Alliance fills for the 802.11 standards.

4.9 Time Awareness on Endpoints

The issues that arise throughout multimedia systems with respect to time awareness are how to get time and the time-sensitive data to/from the application accurately and with low latency, respectively. In this section, we examine issues of how to schedule software tasks, timestamp events, cross-timestamp disparate clocks within an endpoint and trigger events with sufficient time accuracy. We describe each of these in the following with a focus on media applications, though the principles extend to other time-sensitive applications.

From Fig. 4.17 [30], we see that there can be a large number of software layers through which a typical function call must pass, each of which can add nondeterministic delays. If we have delivered accurate time across a network and then wish to read this network time, which typically resides in a counter close to the physical layer of the Ethernet device in the system—far from the CPU, errors can arise. As such, the value ultimately returned by a system time call, such as the POSIX *clock_gettime*, is no longer ‘now’ but ‘soon thereafter’. Modern x86 Central Processing Units (CPUs), therefore, provide a TimeStamp Counter (TSC) in the CPU itself and a native instruction to ReaD the TSC (RDTSC) that provides direct and immediate access to the TSC. But a counter that reports the number of cycles since reset is not sufficient for synchronisation of media streams. Thus, coefficients are maintained by the operating system to convert the current TSC value into the corresponding UTC time. These linearity coefficients may be computed based on timestamp pairs, or cross-timestamps, where the TSC counter and the UTC time are captured as close to simultaneously as possible. Over time, the coefficients are refined and adjusted to track changes in the frequency reference that drives the TSC [31].

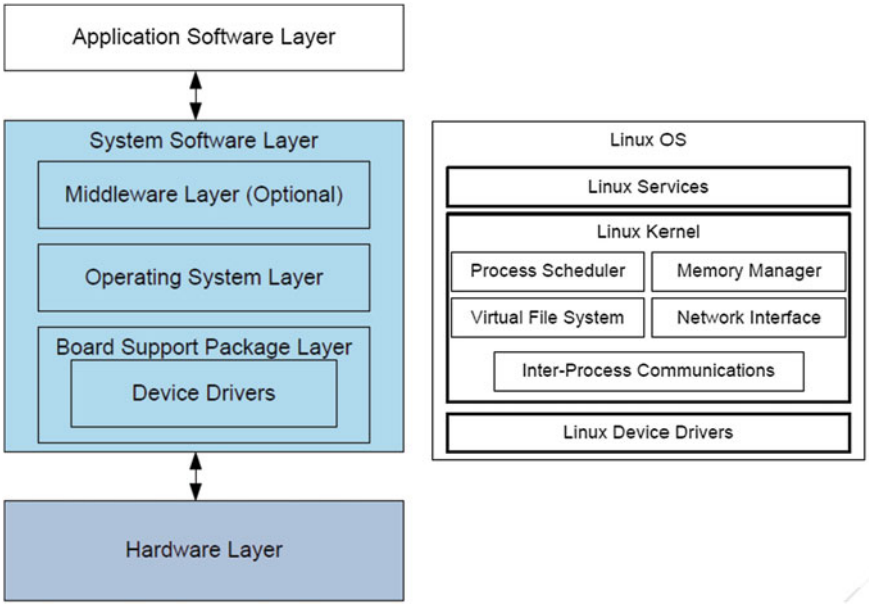


Fig. 4.17 Software layers [30]

Since improved time synchronisation can improve QoE when rendering (or recording) audio and video, it is important to consider that a reduction in the uncertainty of the delay when reading the network time or media position results in commensurate improvement in the linear coefficients, which results in improvement in translation between TSC and UTC or between UTC and the current media position. One such improvement in cross-timestamp accuracy is the PCIe Precision Time Measurement (PTM) protocol, illustrated in Fig. 4.18, which provides the system's time to the Ethernet or Wi-Fi device with better than one microsecond error [32]. As seen in the figure, the Ethernet, or Local Area Network (LAN) device maintains its estimate of the PTP time in a hardware counter. On demand of software, the PTM protocol is initiated, resulting in the simultaneous capture of both system and the PTP timestamps in a cross-timestamp pair, which may be read by software at its leisure. The same method may be used for other I/O devices that contain their own custom clock, including an audio device, as seen in Fig. 4.18.

Likewise, when multiple systems are employed to render multiple channels of a single audio programme, it is necessary to both start rendering audio at the right time on each system and to maintain the proper render rate, accounting for the unique parts-per-million (ppm) offset of the local crystal frequency reference on each system relative to the source. Typically, the render time is communicated in a timestamp point referencing a shared timeline, e.g. as an explicit 802.1AS presentation timestamp field in the media packet header, as in AVTP, or implicitly as a sample number with respect to a prearranged epoch, as specified for RTP in AES67.

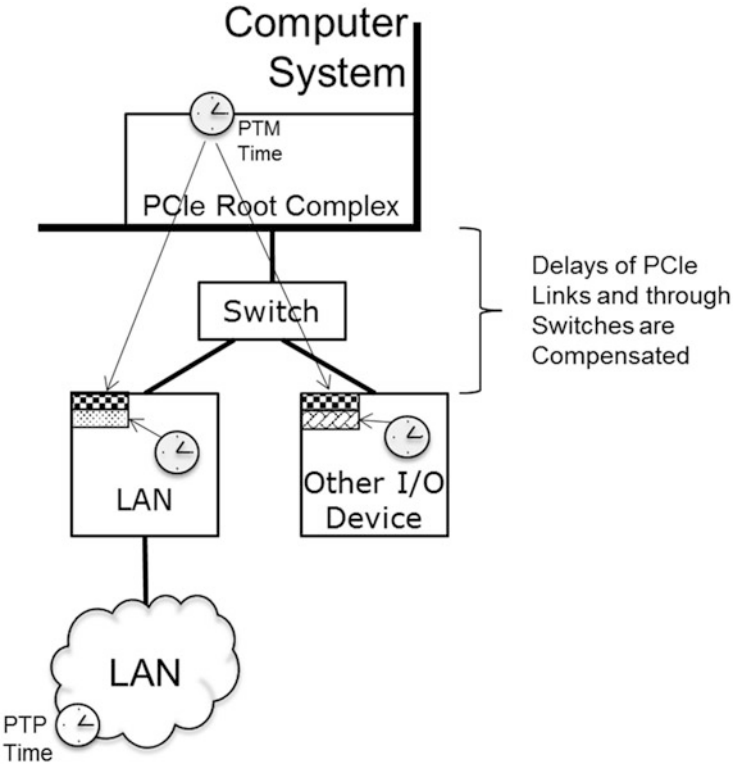


Fig. 4.18 Using PCIe PTM to cross-timestamp system and PTP time

Upon receiving a media frame corresponding with a desired render time, the issue of realising the specific render time arises. Typical MPEG transport systems assume either a single renderer or renders separated by distances large enough that a single person would be able to observe only one renderer at a time, and therefore the only synchronisation requirement is contained within each renderer independently and inter-render phase alignment of the media is typically not addressed. For example, in order for seven Wi-Fi-attached speakers in a home media room to render the seven inter-related audio channels as part of a '5.2 Surround Sound' system—or the related situation arising in professional audio such as Line Array Speakers [33]—audio phase and frequency must be mutually aligned in order to avoid unpleasant artefacts. These situations demand both a shared common timeline and frequency reference between all renderers—such as the PTP profile provided in 802.1AS—and the desired render time with respect to that timeline, explicitly or implicitly, for each audio sample or video frame.

Regardless of their quality, no two clocks agree exactly on the length of one second and their difference varies with temperature, voltage, etc. Thus, two renderers will naturally render audio at slightly different rates, summing to a difference

of perhaps a few samples per second if corrective action is not taken, since the actual average frequency of common crystal oscillators can each differ from the nominal frequency by tens of PPM. Some endpoints allow the audio clock to be disciplined to match the actual average frequency of the source, e.g. by employing a voltage-controlled crystal oscillator (VCxO). But the adjustment to the VCxO requires the system to know whether it is rendering too fast or too slow with respect to the others, and to what degree. Given the availability of 802.1AS time, this can be accomplished through cross-timestamping the network and audio counters.

If fine adjustment of the audio clock is unavailable, the number of samples must be adjusted regularly in order to keep multiple renderers synchronised. Sometimes called 'resampling', the original audio signal is processed to create a string of samples with the same spectral content, but with a slightly different sample rate. Also called Micro Sample Rate Conversion (Micro SRC), the resulting waveform can be made to compensate for any PPM error in the audio clock. For example, if the stream is currently rendering locally at 40 PPM faster than desired (based on timestamps on the common 802.1AS timeline, for example), extra samples must be inserted—about two per second. Typically, this is accomplished by passing a ratio (2/sample rate) and a set of audio samples to the Micro SRC function, which resamples the audio data and returns the appropriate number of samples, in our example, a sequence of samples that is 40 PPM longer than the input sample sequence.

4.10 Conclusions

This chapter has provided a big-picture view of timing with a particular focus on multimedia synchronisation. Starting with some basic definitions of the three timing concepts of time, frequency and phase, it then outlined the challenges of delivering precision timing in modern Information and Communications Technology (ICT) systems. It summarised the work of the TAACCS group that is examining the core issue of time awareness across the full ICT infrastructure, albeit aimed at the broader domain of Internet of Everything (IoE). The chapter then reviewed some of the core synchronisation techniques for time and less so frequency, as well as advances in Ethernet Time-Sensitive Networking (TSN). Finally, it examined endpoint timing issues relating to both hardware and software and detailed a number of initiatives and techniques for precise media rendering. Whilst time/phase and frequency are always a serious consideration for anyone working in media synchronisation, it is important to consider how multifaceted the challenge is and the relationship between the various facets. Essentially, a more integrated and holistic approach will help better deliver existing applications and help conceive new ones.

Definitions

Time an instant (or time of day) on a selected time-scale.

Timescale a family of time codes for a particular coordinate time that provide an unambiguous time ordering of events.

Time synchronisation relative adjustment of two or more sources of time with the purpose of cancelling their time differences.

Frequency refers to a rate of repetition (of an event) per unit of time and thus two clocks are frequency synchronised if they oscillate at the exact same rate.

Syntonisation the relative adjustment of two or more frequency sources with the purpose of cancelling their frequency differences but not necessarily their phase difference.

Time Awareness the extent to which a device, system or device/system model has an appropriate ability to sense and respond to timing signals/information.

References

1. Glossary and Definitions of Time and Frequency Terms, Recommendation ITU-R TF.686-3 (12/2013)
2. Weiss, M., Eidson, J., Barry, C., Broman, D., Goldin, L., Iannucci, B., Lee, E.A., Stanton, K.: Time-Aware Applications, Computers, and Communication Systems (TAACCS). NIST Technical Note 1867. NIST, National Institute of Standards and Technology, http://dx.doi.org/10.6028/NIST.TN.1867, U.S. Department of Commerce, Feb 2015
3. Zhao, Y., Liu, J., Lee, E.A.: A programming model for time-synchronized distributed real-time systems. In: 13th IEEE Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07, Apr 2007, pp. 259–268. http://chess.eecs.berkeley.edu/pubs/325.html
4. Zou, J., Matic, S., Lee, E.: PtidyOS: a lightweight microkernel for Ptides real-time systems. In: Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE (2012)
5. Stanton, K.: Semantics and Software APIs for the Efficient Use of Time in Time Sensitive Systems Using General Purpose Compute Hardware. WSTS, Calif (2015)
6. Beloqui, L., Boronat, F., Montagud, M., Melvin, H.: Understanding timelines within MPEG standards. IEEE Commun. Surv. Tutorials **18**(1), 368–400 (2016)
7. A. E. Society: AES67-2015: AES Standard for Audio Applications of Networks—High-Performance Streaming Audio-Over-IP Interoperability. AES, New York (2014)
8. IEEE: Standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002), pp. c1–269 (2008). https://doi.org/10.1109/ieeestd.2008.4579760
9. IEEE: IEEE 1722—Layer 2 Transport Protocol Working Group for Time-Sensitive Streams. http://grouper.ieee.org/groups/1722/
10. IEEE 802.1. http://www.ieee802.org/1/
11. Romer, K., Blum, P., Meier, L.: Time Synchronization and Calibration in Wireless Sensor Networks, pp. 199–237. Wiley, Hoboken, NJ (2005)

12. Kopetz, H., Ochsenreiter, W.: Clock synchronization in distributed real-time systems. IEEE Trans. Comput. **C-36**(8), 933–940 (1987). ISSN 0018-9340. https://doi.org/10.1109/tc.1987.5009516
13. Mills, D.L.: Internet time synchronization: the network time protocol. IEEE Trans. Commun. **39**(10), 1482–1493 (1991). ISSN 0090-6778. https://doi.org/10.1109/26.103043
14. Elson, J., Girod, L., Estrin, D.; Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 147–163 (2002)
15. Ganeriwal, S., Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks. In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, New York, NY, USA, pp. 138–149. ACM (2003). ISBN 1-58113-707-9. https://doi.org/10.1145/958491.958508
16. Maroti, M., Kusy, B., Simon, G., Ledeczi, A.: The flooding time synchronization protocol. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys '04, New York, NY, USA, pp. 39–49. ACM (2004). ISBN 1-58113-879-2. http://doi.acm.org/10.1145/1031495.1031501
17. TinyOS WG. Tinyos, Apr 2016. http://www.tinyos.net/
18. Mills, D.L.: Improved algorithms for synchronizing computer network clocks. IEEE/ACM Trans. Netw. (TON) **3**(3):245–254 (1995)
19. Mills, D.L.: Computer Network Time Synchronization: The Network Time Protocol, 1st edn. CRC Press (2006). ISBN 978-0849358050
20. Mills, D.L.: Network time protocol (version 3) specification, implementation and analysis (1992)
21. Mills, D.L.: Precision synchronization of computer network clocks. ACM SIGCOMM Comput. Commun. Rev. **24**(2):28–43 (1994)
22. Eidson, J.C.: Measurement, Control, and Communication Using IEEE 1588 (Advances in Industrial Control). Springer (2006). ISBN 978-1846282508
23. Eidson, J.C., Lee, K.: IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In: Sensors for Industry Conference, 2002. 2nd ISA/IEEE, pp. 98–105. IEEE (2002)
24. ITU-T Rec. G.8261, ITU-T Rec. G.8262, ITU-T Rec. G.8264
25. IEEE. http://standards.ieee.org/about/get/802/802.3.html
26. Time Sensitive Networking Task Group. http://www.ieee802.org/1/pages/tsn.html
27. IEEE: Status of IEEE 802.11 TGmc. http://www.ieee802.org/11/Reports/tgm_update.htm
28. Never Lost Indoors—The Promise of Wi-Fi® Location. http://www.wi-fi.org/beacon/rolf-de-vegt/never-lost-indoors-the-promise-of-wi-fi-location
29. IEEE 802.1 TSN. http://www.ieee802.org/1/pages/tsn.html
30. Frank, A., Weisberg, P.: Operating Systems—Structure of Operating Systems
31. Eidson, J.C., Stanton, K.B.: Timing in cyber physical systems: the last inch problem. In: International Symposium on Precision Clock Synchronization for Measurement Control and Communications (ISPCS), pp. 19–24 (2015)
32. P. SIG: Precision Time Measurement. https://pcisig.com/sites/default/files/specification_documents/ECN_PTM_Revision1a_31_Mar_2013.pdf. Accessed 09 2016
33. Olsen, D.: Time Accuracy Requirements in Audio Networks (2007)

Part II
Applications, Use Cases, and Requirements

Chapter 5
Simultaneous Output-Timing Control in Networked Games and Virtual Environments

Pingguo Huang and Yutaka Ishibashi

Abstract In this chapter, we make a survey of techniques for simultaneous output-timing control, which adjusts the output timing of media streams among multiple terminals in networked games and virtual environments. When media units (MUs, each of which is the information unit, such as a video frame and a voice packet for media synchronization) are transmitted over non-guaranteed Quality of Service (QoS) networks like the Internet, the receiving times of each MU at the terminals may be different from each other owing to network delays and delay jitters. Therefore, for example, the fairness among players may be damaged in networked games, and collaborative work may not be done efficiently among users in virtual environments. It is important for multiple players/users to play/do networked games/collaborative work while watching the same displayed images simultaneously. To solve the problems, the simultaneous output-timing control, such as media synchronization control and causality control is needed. In this chapter, as the control, we mainly handle the group (or inter-destination) synchronization control, which is a type of media synchronization control, the adaptive Δ-causality control, and the dynamic local lag control. We also discuss the similarities and differences among the three types of control. Generally, the group synchronization control or adaptive Δ-causality control can be employed to keep the fairness and/or the consistency in good conditions among multiple terminals in networked games and virtual environments, and the dynamic local lag control is used for sound synchronization in networked virtual ensembles. However, the interactivity may seriously be deteriorated under such types of control. Therefore, we introduce prediction control to improve the interactivity. As a result of Quality of Experience (QoE) assessment, we demonstrate that the prediction control improves the interactivity and there is the optimum prediction time according to the network delay. Finally, we discuss the future directions of simultaneous output-timing control in networked games and virtual environments.

P. Huang (✉)
Seijoh University, Aichi 476-8588, Japan
e-mail: huangpg11@gmail.com; huangpg@seijoh-u.ac.jp

Y. Ishibashi
Nagoya Institute of Technology, Nagoya 466-8555, Japan

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_5

Keywords QoS control · Simultaneous output-timing control
Networked game · Virtual environment

5.1 Introduction

Recently, distributed interactive real-time applications, such as networked real-time games (i.e., competitive work) and collaborative work in networked virtual environments become more and more popular along with the development of communication technology and virtual reality technology [1]. In the applications, the virtual environments are shared with multiple users via networks, and the users can collaboratively and/or competitively interact with objects in shared virtual environments. Therefore, users can greatly immerse themselves in the games and the environments [2]. Since multiple users collaboratively/competitively conduct work in a shared world at the same time, it is necessary to output the shared information (for example, position information of objects and avatars of users) simultaneously at all the terminals. Normally, the information is managed by a server or multiple servers in the client–server model and by peers in the P2P model. When the information is transmitted over a non-guaranteed QoS [3] network like the Internet, the receiving times at the terminals (destinations) may be different from each other owing to network delays and delay jitters. This means that some destinations may have already received information while the other destinations may have not received the information. Then, since users (or players) at different destinations may watch different displayed images at the same time, the fairness among the users may be damaged in networked games and the efficiency of collaborative work may deteriorate. Thus, it is important for multiple users to do competitive work/collaborative work while watching the same displayed images simultaneously. To solve the problems, the simultaneous output-timing control is needed.

There are many studies focusing on simultaneous output-timing control [4], such as media synchronization control [5] and causality control [6]. In this chapter, we explain the current status of research on simultaneous output-timing control in networked games and virtual environments, and we mainly handle the group (or inter-destination) synchronization control [5, 7, 8], which is a type of media synchronization control, the adaptive Δ-causality control [9], and the dynamic local lag control [10]. The three types of control are typical examples of simultaneous output-timing control and can widely be used in many applications. We also discuss the similarities and differences among the three types of control.

In this chapter, first, we introduce the necessity of simultaneous output in networked games and virtual environments. Next, we explain techniques for the simultaneous output-timing control, introduce three types of control, and clarify the similarities and differences among them. However, the interactivity may seriously be deteriorated under the control. Then, we introduce prediction control to improve

the interactivity which is degraded by the simultaneous output-timing control. Finally, we discuss the future directions of the simultaneous output-timing control in the networked games and virtual environments.

5.2 Necessity of Simultaneous Output

As described in Sect. 5.1, in most of the networked games and virtual environments, multiple users share a virtual space with each other. Normally, the information is managed by a server or multiple servers in the client–server model and by peers in the P2P model. Therefore, as shown in Fig. 5.1, when a source terminal (server or peer) transmits *media units* (*MUs*), each of which is the information unit for media synchronization [11], to two destinations, the MUs may not arrive at the two destinations simultaneously, and the users may watch different displayed images. This is because there exist network delays and delay jitters. Thus, in networked games, users at the destinations which receive MUs earlier may be in an advantageous position; that is, the fairness among the users may be seriously deteriorated. For example, in a networked shooting game, when users try to shoot target objects, MUs including target information have already arrived at some destinations and users at the destinations may be ready to shoot, but the MUs may not arrive at the other destinations and users at the destinations may still not find the targets. Therefore, the fairness among the users may be damaged. Also, in collaborative work in networked virtual environments, it may be difficult for users to do work collaboratively since they watch different displayed images.

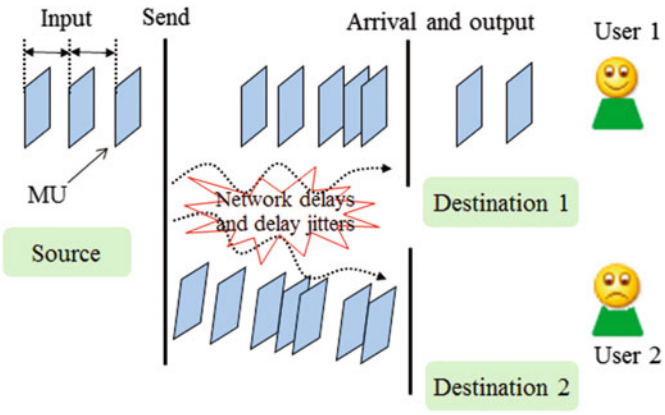


Fig. 5.1 Example of influence of network delay and jitters among different destinations

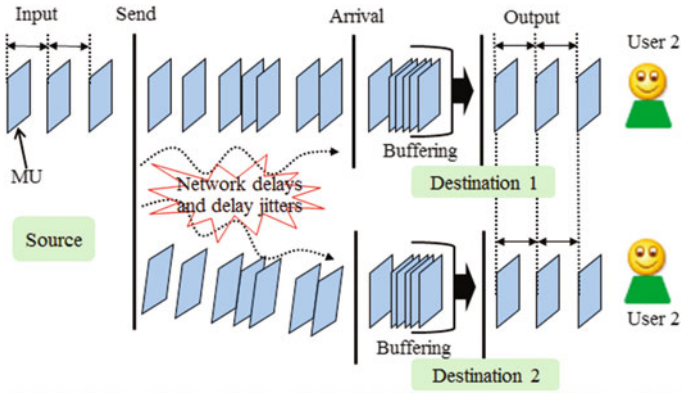


Fig. 5.2 Example of simultaneous output-timing control

In order to solve the problems, we need a simultaneous output-timing control which outputs each MU at all the destinations at the same time. As shown in Fig. 5.2, under the control, when each destination receives MUs, it buffers the MUs if the MUs arrive earlier than the output times, and each MU is output simultaneously at multiple destinations.

5.3 Simultaneous Output-Timing Control

In this section, we explain techniques used for simultaneous output-timing control, explain three types of simultaneous output-timing control, and clarify the similarities and differences among the control.

5.3.1 Techniques for Control

A number of techniques employed for simultaneous output-timing control have been proposed so far. We explain the techniques based on the four items: clocks, techniques at sources and those at destinations [12], and methods to determine the output timing of MUs at all the destinations. The first item denotes whether clocks are globally synchronized or locally available. In order to achieve simultaneous output at all the destinations, some techniques should be employed at both sources and destinations. Also, all the destinations should use the same method to determine the output timing.

(1) Clocks

Most types of the control assume that the clock ticks at the sources and destinations have the same advancement and the current local times are also the same, i.e., *globally synchronized clocks* [5], which make simultaneous output-timing control simpler. We can use the globally synchronized clocks by adjusting the clocks by employing some method such as Network Time Protocol (NTP) [13] or Global Positioning System (GPS).

(2) Techniques at sources

At sources, synchronization information such as timestamp and the sequence number is attached to each MU [14]. The timestamp indicates the generation time of the MU. If MUs are generated periodically, only the sequence number can be used instead of the timestamp.

(3) Techniques at destinations

When each destination receives MUs, the MUs are stored in the destination buffer to compensate for network delay and delay jitter, and then the destination outputs them according to the synchronization information. Therefore, each MU waits for output from the buffer when it arrives earlier than the time at which it should be output (see Fig. 5.2).

(4) Methods to determine output timing

In order to output each MU simultaneously among multiple destinations, the output timing of the MU at all the destinations has to be the same. There are mainly two methods to determine the output timing. One is a distributed method, and the other is a centralized method [15]. In the distributed method, each destination transmits information about the output timing of MUs at the destination to all the other destinations, and determines output timing called the *reference output timing* [10], to which all the destinations should adjust their own output timings, according to the received information from the other destinations by using the same method. Each destination adjusts its output timing to the reference output time gradually. In the centralized method, there is a single manager to gather the output timings from all the destinations, and the manager determines the reference output timing and multicasts the information about the reference output timing to all the destinations. When each destination receives the information about the reference output timing, it gradually adjusts its output timing to the reference output timing [15].

5.3.2 Dynamic Local Lag Control

In the conventional local lag control [16], each source buffers local information for a constant time called the *local lag*, which is denoted by Δ (≥ 0) here, in order to be

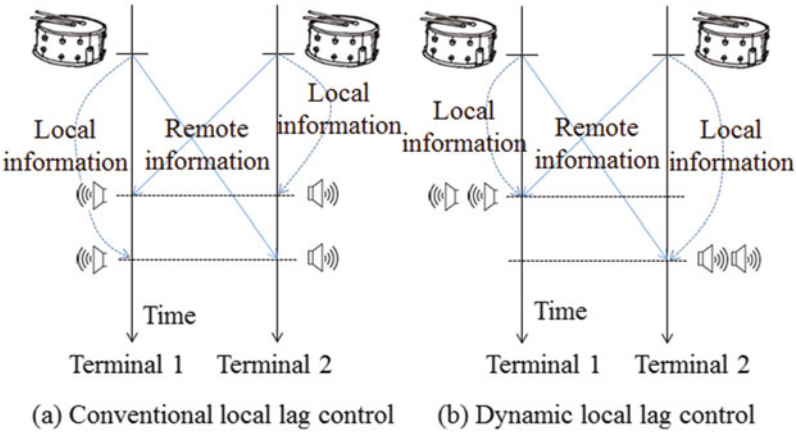


Fig. 5.3 Examples of conventional local lag control and dynamic local lag control

synchronized with destinations (see Fig. 5.3a). In the dynamic local lag control, the value of Δ is dynamically changed according to the network delay from the other terminal to the local terminal [17] (Fig. 5.3b).

From Fig. 5.3, we can see that in the conventional local lag control, Δ is set to the same value as the network delay from the local terminal to the other terminal; on the other hand, in the dynamic local lag control, Δ is set to the same value as the network delay from the other terminal to the local terminal, and the control also changes the value dynamically according to the network delay. When the local lag is set to the same value as the network delay, high quality of synchronization among destinations can be achieved. In Fig. 5.3a, a user of each terminal hears sounds separately; on the other hand, the user in Fig. 5.3b hears the sounds simultaneously. However, the dynamic local lag control degrades the interactivity since the control buffers local information for a constant time.

In [17] and [18], the authors investigate the effects of the dynamic local lag control in a networked haptic drum system, in which two users at different places can play the drum set together with the same rhythms at the same tempi. They also enhance the control so that two or more users can play the networked haptic drum system [19].

5.3.3 Group Synchronization Control

For group synchronization control, there are mainly three schemes. One is the master–slave destination scheme [5], another is the synchronization maestro scheme [7], and the other is the distributed control scheme [8].

In the master–slave destination scheme, the destinations are grouped into a master destination and slave destinations. The master destination carries out only intra-stream synchronization control over a master stream (and slave streams) and interstream synchronization control over the slave streams [5]. The master destination multicasts a control packet that includes the output time of its first MU of the master stream to the slave destinations. In addition, when *the target output time* [11], which denotes an instant at which each destination should output each MU under the group synchronization control when there exists network delay jitter, of the master destination is modified, the master destination notifies all the slave destinations of the modification by distributing a control packet which has the amount of the time which has been modified and the sequence number of the MU at which the target output time has been changed. It should be noted that the target output time can be modified when a large network delay occurs or when network delays become smaller [11]. When each slave destination receives the control packet, the slave destination gradually adjusts its output timing of MUs to the output timing of the master terminal.

In the synchronization maestro scheme, as shown in Fig. 5.4, there are M sources, N destinations, and a synchronization maestro (i.e., a manager described in Sect. 5.3.1). Each destination notifies the synchronization maestro of the information about its output timing. When the synchronization maestro receives the information about the output timing from each destination, it determines the reference output timing [7] and multicasts the information about the reference output timing to all the destinations. Each destination gradually adjusts its output timing to the reference output timing. Note that each source may also be a destination; in this case, the source also outputs each MU after buffering and transmits the MU to the other

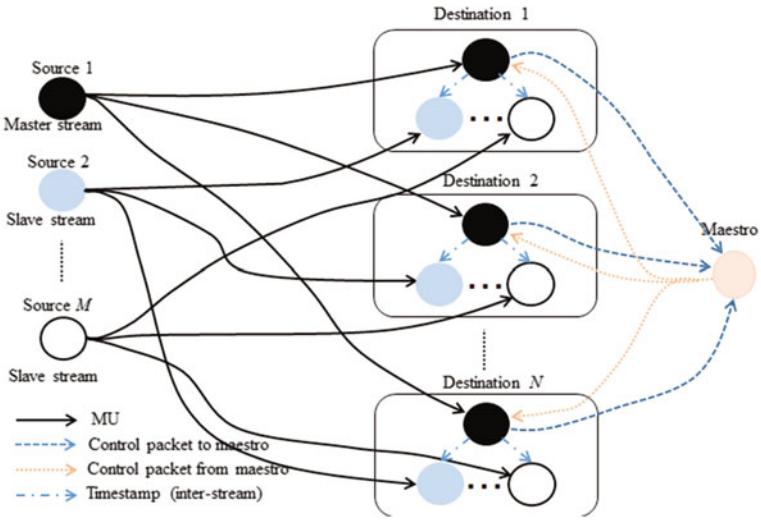


Fig. 5.4 Model of synchronization maestro scheme

destinations. Before output of the MU, the source buffers the MU for a constant time (called the local lag) according to the reference output timing.

The main difference between the distributed control scheme and the synchronization maestro scheme is in how to determine the reference output timing. In the distributed control scheme, each destination notifies the other destinations of the information about its output timing. The destination determines the reference output timing based on the notifications by using the same method as the other destinations and gradually adjusts its output timing to the reference output timing.

In [9], the authors propose a media synchronization scheme with adaptive Δ-causality control (we will explain the control in Sect. 5.3.4), and employ the synchronization maestro scheme as group synchronization control. They investigate the effects by implementing the proposed scheme in a networked shooting game. In the game, two users fight with each other, and each user fires shots at the other user's fighter while moving his/her own fighter to the right or to the left, and he/she avoids shots fired by the other user by shielding his/her own fighter under buildings. Experimental results show that the scheme improves the fairness among users and reduces the inconsistency rate which is defined as the ratio of the number of computer data MUs each of which is received by only one terminal to the total number of computer data MUs. However, when the network load is heavy, the causality and the interactivity may be disturbed. In [20], the authors investigate the effect of the proposed scheme in [9] by comparing the effects of the synchronization maestro scheme and distributed control scheme in the networked shooting game. Experimental results show that the two schemes have their own advantages and disadvantages. The synchronization maestro scheme (a centralized control scheme) is superior to the distributed control scheme in terms of the causality, and the distributed control scheme outperforms the centralized control scheme from the points of view of the consistency, the fairness, and the interactivity.

In [15], the synchronization maestro scheme and distributed control scheme are handled in a networked virtual environment where users have a conversation with each other while moving as avatars. In [21] and [22], the authors investigate effects of the synchronization maestro scheme by experiment in which two users collaboratively raise and move an object in networked virtual environments with haptics. Experimental results show that the scheme can improve the efficiency of collaborative work. In [23], the effect of the synchronization maestro scheme is also investigated in a networked real-time game. In the game, each of two users lifts and moves his/her object (a rigid cube) to contain a target. When the target is contained by either object of the two users, it disappears and then appears at a randomly selected position in a virtual space. The two users compete on the number of eliminated targets with each other. Experimental results show that the scheme can improve the fairness in the networked real-time game. The effects of the synchronization maestro scheme are also clarified in a networked real-time game with collaborative work [24]. In the game, two groups each of which consists of two users play the networked real-time game in a 3-D virtual space. The two users in

each group collaboratively lift and move an object which is assigned to the group to contain the target. The two groups compete on the number of eliminated targets with each other. The effects of the synchronization maestro scheme are further investigated in a first-person shooting game [25]. In the game, two avatars fire bullets while moving in a 3-D virtual space and hit the bullets to each other. QoE (Quality of Experience) assessments [26] show that the scheme can maintain the fairness high and there is the optimum value of Δ_H (defined as the maximum value of Δ which equals to the target output time of an MU in the group synchronization control minus the generation time of the MU) to balance the interactivity and fairness.

In [27], the authors assess the effects of the distributed control scheme in a networked game and a networked virtual environment. They deal with a name-guessing task like fastest fingers first as competitive work and networked rock-paper–scissors as collaborative work. Assessment results show that the fairness among users can be maintained high in the networked game and the synchronization quality of networked collaborative work can be improved by the scheme.

5.3.4 *Adaptive Δ-Causality Control*

The Δ-causality control keeps the causal relationships (i.e., temporal order) among events [6]. As shown in Fig. 5.5, in the Δ-causality control, each MU has a time limit which is equal to the generation time of the MU plus Δ seconds for the preservation of the real-time property, and the MU is output at the time limit. If the MU is received after the time limit, it is discarded as an obsolete MU. The causality among the output MUs can be kept. In the adaptive Δ-causality control, the value of Δ in the Δ-causality control is dynamically changed according to the network load (i.e., the network delay and delay jitter) [20]. However, it should be noted that changing the value of Δ may disturb the causality among MUs.

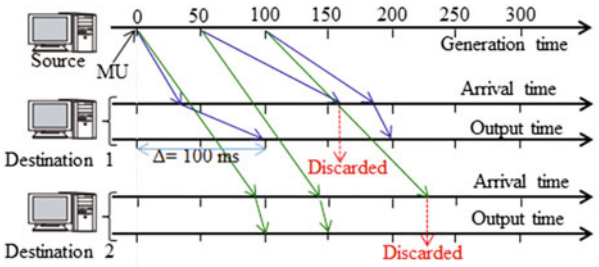


Fig. 5.5 Example of MU output timing under Δ-causality control

Note that, as in group synchronization control, each source may also be a destination which also outputs each MU transmitted to the other destinations. In this case, each source also buffers MUs for a constant time Δ s (called the local lag).

As described in Sect. 3.3, in [9] and [20], the authors also employ the adaptive Δ-causality control in a networked shooting game and found that the control is effective for the game.

In order to achieve a further amelioration of consistency, an adaptive Δ-causality control with adaptive dead reckoning is proposed in [28]. Experimental results show that the proposed control can improve the consistency among users, and when the value of Δ is larger than 100 ms, the interactivity is degraded. In [29], the authors implement the Δ-causality control scheme with adaptive dead reckoning in a networked air hockey game with haptics, and experiment results show that the fairness can be kept high.

In [30], the authors propose an adaptive scheme with the adaptive Δ-causality control and prediction control. The adaptive Δ-causality control is exerted for the conservation of causality. When there exist no MU for the output in the buffer, the prediction control predicts the current positions of objects by using the latest two output MUs. They compare the proposed scheme with other six schemes in a networked racing game. Experiment results show that the proposed scheme keeps the consistency among users and the interactivity high.

5.3.5 Similarities and Differences Among Three Types of Control

As described above, the group synchronization control or adaptive Δ-causality control can be employed to keep the fairness and/or the consistency in good condition among multiple terminals in networked games and virtual environments, and the dynamic local lag control is used for sound synchronization in networked virtual ensembles. The adaptive Δ-causality control is very similar to the distributed control scheme for group synchronization control which adjusts the reference output timing by Δ. The difference between the two types of control is that in the distributed control scheme, the streams are grouped into a master stream and slave streams, and intra-stream and interstream synchronization control is carried out; in the adaptive Δ-causality control, the output time of each MU is only related to the generation time of the MU and Δ, and if the MU is received after the time limit, it is discarded even though there is not MU for output.

The adaptive Δ-causality control can preserve the causal relation among MUs; generally, the causality control tries to output MUs in the generation order of the MUs. However, the causality control may not always output MUs simultaneously at all the destinations. On the other hand, achieving the group synchronization perfectly can lead to the simultaneous output at all the destinations and preserve causality with synchronization.

If the networks delays between all the terminals are the same, the dynamic local lag control can perfectly achieve group synchronization and causality among destinations. Therefore, the dynamic local lag control is also similar to the adaptive Δ-causality control. The difference is that the Δ value of the dynamic local lag control is determined according to the network delay from the other terminal to the local terminal, and the Δ value of the adaptive Δ-causality control is determined by the network delay in the opposite direction.

5.4 Improvement of Interactivity by Prediction Control

As described in Sect. 5.3, in order to output MUs simultaneously at all the destinations, it is necessary to buffer the MUs at the destinations and keep the same output timing under the simultaneous output-timing control. This means that the control degrades the interactivity owing to buffering. Prediction is one of the methods which can improve the interactivity [31].

In [32], the authors propose group synchronization control with a prediction for work using haptic sense. The control adjusts the output timing among multiple terminals and keeps the interactivity high. It outputs position information by predicting the future position later than the position included in the last-received MU by a fixed amount of time, and it also advances the output time of position information at each local terminal by the same amount of time. The proposed control is implemented in two types of work (remote haptic drawing instruction [33] and collaborative haptic play with building blocks [34]). According to the system model of the two types of work, the distributed control scheme is adopted for group synchronization for the remote haptic drawing instruction, and the synchronization maestro scheme for the collaborative haptic play with building blocks. The effects of the proposed scheme are investigated by QoE assessment. In this section, we present assessment results in the collaborative haptic play with building blocks.

As shown in Fig. 5.6, in the collaborative haptic play with building blocks, two users pile up building blocks 1, 2, 3, and 4 collaboratively to build a dollhouse which consists of 26 blocks (see Fig. 5.6) in a 3-D virtual space. The effects of the proposed control are investigated by QoE assessment subjectively and objectively. In the assessment, a network emulator NIST Net [35] which is used instead of the network in Fig. 5.6 generates an additional delay according to the Pareto-normal distribution for each packet transmitted between the two terminals. For the subjective assessment, each subject is asked to practice on the condition that there is no additional delay and the group synchronization control is not carried out. Then, the subject is asked to give a score from 1 through 5 (5: Imperceptible; 4: Perceptible, but not annoying; 3: Slightly annoying; 2: Annoying; 1: Very annoying) according to the degree of deterioration on the condition that there exist additional delays and the group synchronization control with prediction is performed. We obtain the

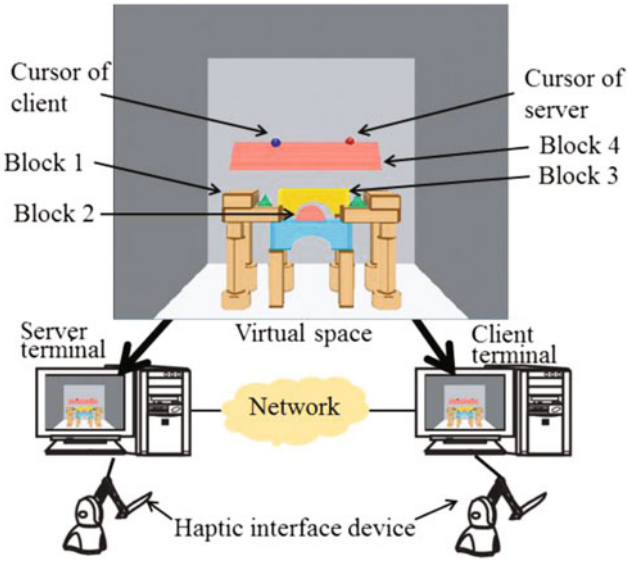


Fig. 5.6 System configuration of collaborative haptic play with building blocks

Mean Opinion Score (MOS) [36] by averaging scores given by all the subjects. For objective assessment, we employ the average operation time which is defined as the average time from the moment the play is started until the instant all the blocks are piled up. This parameter denotes the work efficiency of subjects.

In the QoE assessment, the prediction time T_{predict} is set to 0, 10, 20, 30, and 50 ms, and the average additional delay is set to 50 and 100 ms. When the average additional delay is set to 50 ms, the standard deviation is changed from 0 ms to 20 ms at intervals of 5 ms; when the average additional delay is 100 ms, the standard deviation is changed from 0 ms to 40 ms at intervals of 10 ms.

Since the MOS values at the client terminal were almost the same as those at the server terminal, we here show only the MOS values at the server terminal versus the standard deviation of the additional delay when the additional delay is 50 ms and 100 ms in Figs. 5.7 and 5.8, respectively. The objective assessment results when the additional delay is 50 ms and 100 ms are shown in Figs. 5.9 and 5.10, respectively. In the figures, we also plot the 95% confidence intervals.

From Fig. 5.7, we see that the MOS values of $T_{\text{predict}} = 10$, 20, and 30 ms are larger than that of $T_{\text{predict}} = 0$ ms (only the group synchronization control is carried out), and that of $T_{\text{predict}} = 20$ ms is larger than those of the other prediction times. However, the MOS value of $T_{\text{predict}} = 50$ ms tends to be less than that of $T_{\text{predict}} = 0$ ms. From Fig. 5.8, we also notice that the MOS value of $T_{\text{predict}} = 20$ or 30 ms is the largest. Therefore, we can say that the proposed control is effective and there exists the optimum value of the prediction time, and the optimum value depends on the standard deviation of the additional delay and the average additional delay.

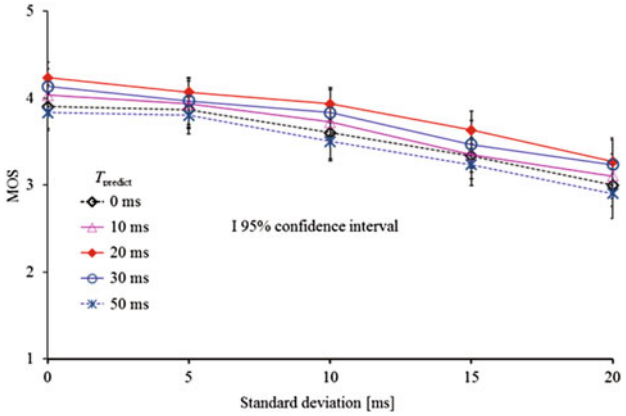


Fig. 5.7 MOS at server terminal versus standard deviation of additional delay (average additional delay: 50 ms)

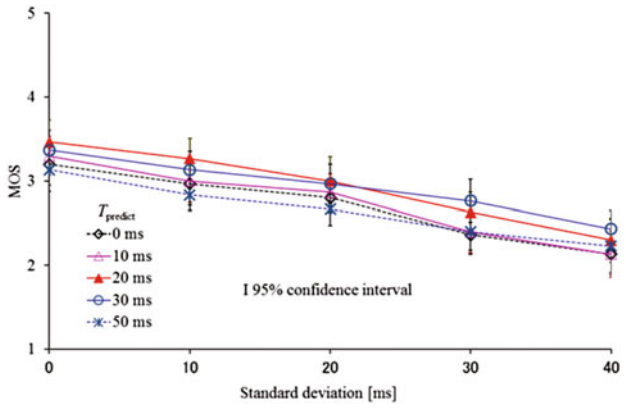


Fig. 5.8 MOS at server terminal versus standard deviation of additional delay (average additional delay: 100 ms)

In Fig. 5.9, we see that the average operation time of $T_{\text{predict}} = 20$ ms is the shortest, and in Fig. 5.10, the average operation time of $T_{\text{predict}} = 20$ or 30 ms is shorter than those of the other prediction times. In the two figures, we can see similar tendencies to those in Figs. 5.7 and 5.8, respectively; note that the magnitude relation of the average operation time is opposite to that of MOS value.

We found that the proposed control is also effective for the remote haptic drawing instruction [33], but the optimum value of the prediction time is different from that in the collaborative haptic play with building blocks. Therefore, we can say that the proposed control is effective and there is the optimum value of the

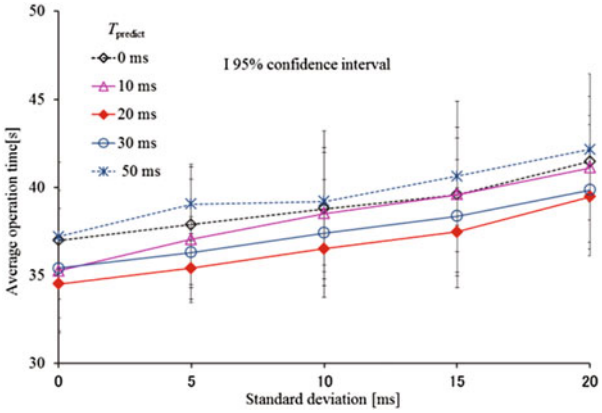


Fig. 5.9 Average operation time versus standard deviation of additional delay (average additional delay: 50 ms)

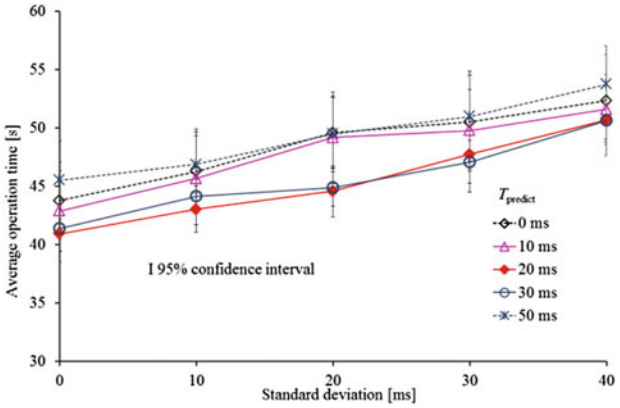


Fig. 5.10 Average operation time versus standard deviation of additional delay (average additional delay: 100 ms)

prediction time, and the optimum value depends on the standard deviation of the network delay, the average network delay, and the type of work.

In [37], the authors propose the adaptive Δ-causality control with prediction in order to keep the causality, consistency, and interactivity high. The proposed control outputs the position information by predicting the future position later than the output time of received position information by a fixed amount of time and advances the output time of position information at the local terminal by the same amount of time. By handling the proposed control in a networked air hockey game using haptic media, they found that the proposed control can keep the interactivity

high and there exists the optimal value of the prediction time. Therefore, in [38], the authors propose the adaptive Δ-causality control with dynamic control of prediction time, which dynamically change the prediction time according to the network delay. They investigate the effects of the control in the networked air hockey game using haptic media, and experiment results show that the control is effective.

As for the dynamic local lag control, in [39], the authors propose the dynamic local lag control with dynamic control of prediction time, which dynamically changes the prediction time according to the network delay in the joint performance of the networked haptic drum system. Experiment results show that the proposed control is effective.

5.5 Future Directions of Simultaneous Output-Timing Control

There exist many types of QoS control, for example, simultaneous output-timing control, traffic control, error control, CPU load control, and so on. If we perform these types of control independently, the effects may be insufficient. Let us consider when the traffic control and CPU load control are employed independently in a video transmission system. When the network load becomes heavy, the traffic control starts to reduce the average bit rate of the video. At the same time, if the CPU load is low, the CPU load control starts to increase the average bit rate. As a result, the two types of control become ineffective. On the other hand, when the network load and CPU load are heavy, the traffic control reduces the average bit rate of video and the CPU load control also decreases the average bite rate; the resultant average bit rate may be too small. Therefore, we need to perform the *integrated QoS control* (called the *adaptive QoS control* in [40]). In the integrated QoS control, the network load condition and CPU load condition are grouped into several levels. According to measured results of network load and CPU load conditions (levels), different combinations of traffic control and processing load control are employed for distributed virtual environments.

Also, it is necessary to take account of the human perception as in [41] and [42] when we consider new control. This purpose is to carry out QoS control so that users can hardly perceive any degradation of quality. That is, as in [41] and [42], we can employ three types of synchronization error ranges for simultaneous output-timing control: The *imperceptible range* (a range of the synchronization error in which almost all users cannot perceive the error), *allowable range* (a range of the synchronization error in which users can allow the error), and *operation range* (a range of the error which should be kept usually). In [41], if the synchronization error goes out the imperceptible range, the control tries to put the error into the operation range. In [42], if the synchronization error is outside the allowable range, the control tries to reduce the error gradually until the error enters

the imperceptible range and if the error is within the allowable range, the control accepts the error.

Thus, we need to study how to integrate multiple types of QoS control efficiently by taking account of human perception.

5.6 Conclusions

In this chapter, we make a survey of simultaneous output-timing control and classify the techniques based on the clocks, techniques at sources and those at destinations, and the methods to determine the output timing. We explain three types of simultaneous output-timing control: Group synchronization control, adaptive Δ-causality control, and dynamic local lag control. We also discussed the similarities and differences among the three types of control. Furthermore, we introduced the prediction control to improve the interactivity which is degraded by the simultaneous output-timing control, and investigated effects of the control in networked games and virtual environments. Finally, we also discussed the future direction of the simultaneous output-timing control.

We need to continue to make a survey of the control since the research area is still growing and new control will be proposed in the future.

Acknowledgements The authors thank Prof. Hitoshi Watanabe of Tokyo University of Science for his valuable discussions.

Definitions

Quality of Service (QoS) QoS is the quality of service which is provided from a layer to its upper layer in network layer model and it is defined as how much the service is faithful to the ideal situation.

Quality of Experience (QoE) QoE is also called the user-level QoS. QoE is the quality which is perceived subjectively and/or experienced objectively by end-users.

Mean Opinion Score (MOS) MOS is a scale of the rating scale method and is obtained by averaging scores of all the subjects for each stimulus. MOS is one of subjective QoE measures. In the rating scale method for example, the five-grade quality scale or impairment scale is employed.

Non-guaranteed QoS network A network which does not guarantee the Quality of Service (QoS).Therefore when packets are transmitted over the network, large network delays, delay jitter, and packet loss may occur.

References

1. Smed, J., Kaukoranta, T., Hakonen, H.: A review on networking and multiplayer computer games. Technical Report 454. Turku Center for Computer Science, Apr 2002
2. Chiueh, T.: Distributed systems support for networked games. In: IEEE 6th Workshop on Hot Topics in Operating Systems, pp. 99–104, May 1997
3. ITU-T Rec. I. 350: General aspects of quality of service and network performance in digital networks, including ISDNs, Mar 1993
4. Montagud, M., Boronat, F., Stokking, H., Brandenburg, R.V.: Inter-destination multimedia synchronization: schemes, use cases and standardization. Multimedia Syst. **18**(6), 459–482 (2012)
5. Ishibashi, Y., Tsuji, A., Tasaka, S.: A group synchronization mechanism for stored media in multicast communications. In: Proceedings of the IEEE INFOCOM, pp. 693–701, April 1997
6. Yavatkar, R.: MCP: a protocol for coordination and temporal synchronization in multimedia collaborative applications. In: Proceedings of the ICDS, pp. 606–613, June 1992
7. Ishibashi, Y., Tasaka, S.: A group synchronization mechanism for live media in multicast communications. In: Conference Rec. IEEE GLOBECOM, pp. 746–752, Nov 1997
8. Ishibashi, Y., Tasaka, S.: A distributed control scheme for group synchronization in multicast communications. In: Proceedings of the ISCOM, pp. 317–323, Nov 1999
9. Ishibashi, Y., Tasaka, S., Tachibana, Y.: Adaptive causality and media synchronization control for networked multimedia applications. In: Conference Rec. IEEE ICC, pp. 952–958, June 2001
10. Sithu, M., Ishibashi, Y., Fukushima, N.: Effects of dynamic local lag control on sound synchronization and interactivity in joint musical performance. ITE Trans Media Technol. Appl. Spec. Sect. Multimedia Transm. Syst. Serv. **2**(4), 299–309 (2014)
11. Ishibashi, Y., Tasaka, S.: A synchronization mechanism for continuous media in multimedia communications. In: Proceedings of the IEEE INFOCOM, pp. 1010–1019, April 1995
12. Ishibashi, Y., Tasaka, S.: A comparative survey of synchronization algorithms for continuous media in network environments. In: Proceeding of the IEEE LCN, pp. 337–348, Nov 2000
13. Mills, D.L.: Internet time synchronization: the network time protocol. IEEE Trans. Commun. **39**(10), 1482–1493 (1991)
14. Ishibashi, Y., Tasaka, S., Hasegawa, T.: The virtual-time rendering algorithm for haptic media synchronization in networked virtual environments. In: Proceedings of the CQR, pp. 213–217, May 2002
15. Ishibashim, Y., Tomaru, K., Tasaka, S., Inazumi, K.: Group synchronization in networked virtual environments. In: Proceedings of the IEEE ICC, pp. 885–890, May 2003
16. Mauve, M., Vogel, J., Effelsberg, W.: Local lag and timewrap: providing consistency for replicated continuous application. IEEE Trans. Multimedia **6**(1), 47–57 (2004)
17. Sithu, M., Ishibashi, Y., Fukushima, N.: Dynamic local lag control for sound synchronization in joint musical performance. In: Proceedings of the NetGames, Dec 2013
18. Sithu, M., Ishibashi, Y., Fukushima, N.: Effects of dynamic local lag control on sound synchronization and interactivity in joint musical performance. ITE Trans. Media Technol. Appl. **2**(4), 299–309 (2014)
19. Sithu, M., Ishibashi, Y., Fukushima, N.: Enhancement of dynamic local lag control for networked musical performance. In: Proceeding of the IEEE GCCE, Oct 2014
20. Ishibashi, Y., Tasaka, S.: Causality and media synchronization control for networked multimedia games: centralized versus distributed. In: Proceedings of the NetGames, pp. 42–51, May 2003
21. Ishibashi, Y., Hasegawa, T., Tsaka, S.: Group synchronization control for haptic media in networked virtual environments. In: Proceedings of the Haptics, pp. 106–113, Mar 2004
22. Kaneoka, H., Ishibashi, Y.: Effects of group synchronization control over haptic media in collaborative work. In: Proceedings of the ICAT, pp. 138–145, Nov/Dec 2004

23. Ishibashi, Y., Kaneoka, H.: Group synchronization for haptic media in a networked real-time game. IEICE Trans. Commun. **E89-B**(2), 313–319 (2006)
24. Hashimoto, T., Ishibashi, Y.: Group synchronization control over haptic media in a networked real-time game with collaborative work. In: Proceeding of the NetGames, Oct 2006
25. Ida, Y., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE assessment of interactivity and fairness in first person shooting with group synchronization control. In: Proceeding of the NetGames, Nov 2010
26. ITU-T Rec. P. 10/G. 100 Amendment 1: New appendix I—definition of quality of experience (QoE), Jan 2007
27. Hosoya, K., Ishibashi, Y., Sugawara, S., Psannis, K.E.: Effects of group synchronization control in networked virtual environments with avatars. In: Proceedings of IEEE/ACM DS-RT, pp. 119–127, Oct 2008
28. Ishibashi, Y., Hashimoto, Y., Ikedo, T., Sugawara, S.: Adaptive—causality control with adaptive dead-reckoning in networked games. In: Proceedings of NetGames, pp. 75–80, Sept 2007
29. Kusunose, Y., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE comparison of competition avoidance methods for management of shared object in networked real-time game with haptic media. In: Proceedings of ICAT'11, Nov 2011
30. Ikedo, T., Ishibashi, Y.: An adaptive scheme for consistency among players in networked racing games. In: Proceedings of FMUIT'06, pp. 154–157, May 2006
31. Tsumaki, Y., Yokohama, M.: Predictive motion display for acceleration based teleoperation. In: Proceedings of IEEE ICRA, pp. 2927–2932, May 2006
32. Huang, P., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE assessment of group synchronization control scheme with prediction in work using haptic media. IJCNS **5**(6), 321–331 (2012)
33. Asano, T., Ishibashi, Y., Kameyama, S.: Interactive haptic transmission for remote control system. In: Proceedings of IEEE ICME, pp. 2113–2116, July 2006
34. Huang, P., Ishibashi, Y., Fukushima, N., Sugawara, S.: Collaborative haptic play with building blocks. In: Proceedings of ACM SIGCHI ACE, Oct 2009
35. Carson, M., Santay, D.: NIST Net-A linux based network emulation tool. ACM SIGCOMM Comput. Commun. Rev. **33**(3), 111–126 (2003)
36. ITU-R BT. 500-12: Methodology for the subjective assessment of the quality of television pictures. International Telecommunication Union, Sept 2009
37. Kusunose, Y., Ishibashi, Y., Fukushima, N., Sugawara, S.: Adaptive delta-causality control scheme with prediction in networked real-time game using haptic media. In: Proceedings of APCC, pp. 800–805, Oct 2012
38. Hara, Y., Ishibashi, Y., Fukushima, N., Sugawara, S.: Adaptive delta-causality control scheme with dynamic control of prediction time in networked haptic game. In: Proceedings of NetGames, Nov 2012
39. Sithu, M., Ishibashi, Y., Fukushima, N.: Effect of dynamic local lag control with dynamic control of prediction time in joint haptic drum performance. In: Proceedings of ICCA, pp. 343–350, Feb 2014
40. Ishibashi, Y., Nagasaka, M.: An adaptive QoS control scheme of avatars in distributed virtual environments. J. Inst. Image Inf. Tele. Eng. **60**(11), 1835–1839, Nov 2006 (in Japanese)
41. Sannomiya, H., Osada, J., Ishibashi, Y., Fukushima, N., Sugawara, S.: Inter-stream synchronization control with group synchronization algorithm. In: Proceeedings of IEEE GCCE, pp. 520–524, Oct 2013
42. Ishibashi, Y., Kanbara, T., Tasaka, S.: Inter-stream synchronization between haptic media and voice in collaborative virtual environments. In: Proceedings of ACM Multimedia, pp. 604–611, Oct 2004

Chapter 6
Automated Video Mashups: Research and Challenges

Mukesh Kumar Saini and Wei Tsang Ooi

Abstract The proliferation of video cameras, such as those embedded in smartphones and wearable devices, has made it increasingly easy for users to film interesting events (such as public performance, family events, and vacation highlights) in their daily lives. Moreover, often there are multiple cameras capturing the same event at the same time, from different views. Concatenating segments of the videos produced by these cameras together along the event time forms a *video mashup*, which could depict the event in a less monotonous and more informative manner. It is, however, inefficient and costly to manually create a video mashup. This chapter aims to introduce the problem of automated video mashup to the readers, survey the state-of-the-art research work in this area, and outline the set of open challenges that remain to be solved. It provides a comprehensive introduction to practitioners, researchers, and graduate students who are interested in the research and challenges of automated video mashup.

Keywords Automated video mashup · Video clips synchronization · Video quality analysis · Cinematography rules · Mashup quality evaluation

6.1 Introduction

Users often record video clips of interesting events and activities using their smartphones, wearable devices, or standalone cameras to preserve the memory and share the experience with others. Because of the ease of recording, storing, and sharing, one can often find a large number of videos recorded of the same popular event. It

M. K. Saini
Indian Institute of Technology Ropar, Rupnagar, India
e-mail: mukesh@iitrpr.ac.in

W. T. Ooi (✉)
National University of Singapore, Singapore, Singapore
e-mail: weitsang@nus.edu.sg

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_6

is, however, boring and time-consuming to view all these videos. A video mashup is a video that is produced by concatenating video segments cut from video clips recorded at the same event by different cameras. The concatenation can produce a mashup that is either time-synchronized, i.e., follows the same timeline as the event, or asynchronous, i.e., longer or shorter than the actual event. In this chapter, we mainly focus on time-synchronous mashups. Most components of the framework, however, apply to asynchronous mashup as well.

In a professional video production, video mashups are often edited and produced by skilled human editors, directors, and producers. We are, however, concerned with the problem of *automatically* producing a time-synchronized video mashup from a large number of user-generated videos with little or no human intervention. Done properly, the automatically produced video mashup will (i) be more entertaining to watch, as it produces a video depicting the event from different angles at different times, instead of monotonously from a single angle; (ii) be more informative, as when part of the scene that is interesting is occluded, the video mashup can show a video clip recorded from a non-occluded camera; and (iii) save time and effort, compared to having a video editing professional who manually watches and analyses a large number of video clips to produce a mashup.

This chapter takes a tutorial approach to provide a comprehensive introduction to the building blocks of a video mashup system, particularly targeted toward practitioners, researchers, and graduate students. Section 6.2 introduces the problem of video mashup to readers with an overview of a typical video mashup system and the problem formulation. Survey of the state-of-the-art research work in this area is provided from Sects. 6.3 to 6.7. Finally, Sect. 6.8 outlines the set of open challenges that remain to be solved.

6.2 Overview of Video Mashup

A video mashup is a video that consists of segments (also called shots) taken from two or more video clips. In this chapter, we use the term video clip to refer a continuous recording of a single device. A typical example of a video mashup is a professional pop music video where multiple video shots are put together to be viewed along the music piece [1]. In most cases, the given video clips record the same event from different perspectives, e.g., a dance performance or a football game. Two video clips recorded from two significantly different angles may depict different scenes even if they are recording the same event. Normally, at an event, however, there is a dominant audio source (e.g., dance music or sport announcement). As such, there is a strong correlation in the audio tracks of the recorded video clips. Nonetheless, if the event occurs over a larger geographical area, or there is no dominant audio source, the video clips may contain different audio tracks.

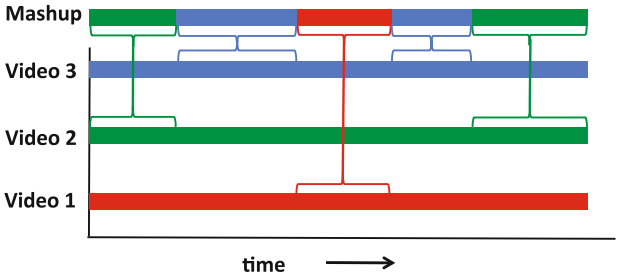


Fig. 6.1 Time-synchronized video mashup

A video mashup can be either time-synchronized or asynchronous. In asynchronous mashups, the mashup timeline generally does not correspond to the event time, i.e., the mashup can be longer or shorter than the actual event. *Video summarization* is a typical example of an asynchronous mashups [2]. The subject topic of this chapter is time-synchronized mashup in which the video clips overlap in time and the mashup timeline corresponds to the event time. An example of a time-synchronized mashup is given in Fig. 6.1. In this example, there are three video clips represented as red, green, and blue color bars. The top multicolor bar represents the video mashup, which is created by concatenating segments taken from the video clips.

How effective a mashup is would depend on the application scenario. A mashup might be created for surveillance purposes—for instance, for a security operator to examine an event that has happened through multiple security cameras. In this case, the mashup would be effective if it maximizes the amount of information gained by the security operator [3]. Similarly, a mashup of videos depicting a sport event would be effective if it captures the main actors and events of the game such as scoring a goal or kicking the ball [4]. In the case of live concerts, the mashups are created such that they are aesthetically pleasing and interesting to the viewer [5].

The timing requirement for producing mashup differs by application scenarios as well. For use cases where the mashup needs to be broadcast live, the mashup needs to be done in near real time, with tolerable delay in order of seconds. In such cases, the mashup takes input from live video cameras and has to be done online, i.e., using only information from the past [6]. In other cases, where the mashup is produced from pre-recorded videos (such as user-generated videos from social websites), the mashup can be produced offline [7], possibly using multiple passes through the videos to analyze the content and the context. Generally, offline mashup is easier and produces better quality mashup as more information available. Table 6.1 summarizes three example application scenarios given above.

In addition to the application scenario, the recording device could also vary, e.g., smartphones and camcorders [5], social cameras [8], or static cameras [4]. The

Table 6.1 Three application scenarios, their corresponding quality attributes, and online/offline requirements

Scenario	Quality attributes	Online/Offline
Sports video broadcast	Important actions and activities of the players	Either
Surveillance and monitoring	Informative coverage and details of all targets	Either
Live concerts broadcast	Interesting and aesthetically pleasant overview of the event	Online

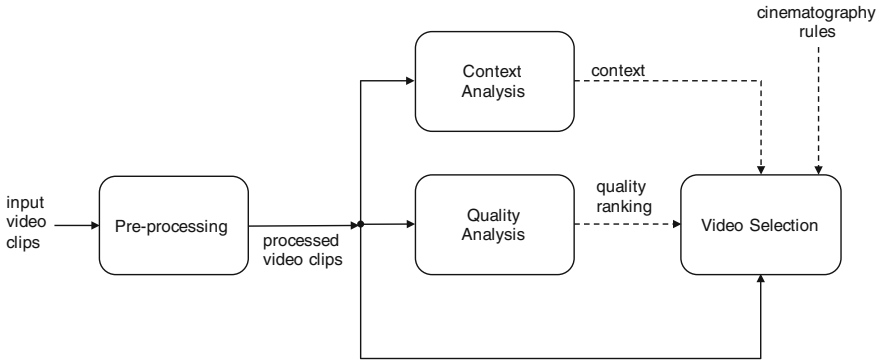


Fig. 6.2 Block diagram of a typical mashup system

mashup framework may vary according to the mashup criteria and type of input videos. For example, in the case of smartphone videos, we need to determine the shakiness of the input videos and filter out the unstable videos, whereas surveillance videos are mostly stable and the main selection criterion is the object appearance.

A block diagram of a typical mashup framework is given in Fig. 6.2. The videos are first preprocessed to align over a common timeline and the same format (i.e., resolution and frame rate). Videos need to be on a common timeline so that the content is in sync when we switch from one video clip to another. Similarly, video clips should be converted to a common format to ensure a smooth transition. While the chosen timeline depends on the earliest and latest available video clips, the format is often determined based on the smallest resolution and lowest frame rate. The preprocessed videos are analyzed to assess the video quality (shakiness, occlusion, etc.) and to detect the recording context of the camera (angle, position, etc.). Note that at a given time, generally there are multiple videos that meet the quality requirements. Cinematography rules are applied on these videos according to the contextual parameters to choose the final video clip.

We can formulate the problem of time-synchronized video mashup creation as follows. The input to the problem is a set of k video clips $V_1, \ldots, V_k$, which, without loss of generality, can be assumed to be of equal length and equal frame rate. In the preprocessing step, a video that is shorter can be padded with empty frames at the beginning or at the end of the video; video with lower frame rate can be filled with interpolated frames using motion interpolation.

A video clip V_i can be abstractly defined as a sequence

$$V_i = \left\langle (f_{i,t_0}, a_{i,t_0}), (f_{i,t_1}, a_{i,t_1}), ..., (f_{i,t_n}, a_{i,t_n}) \right\rangle,$$

where $f_{i,t}$ is a video frame that consists of a 2D array of pixels captured at time closest to t; a_{i,t_j} is the chunk of audio samples captured between time t_{j-1} and t_j from video V_i; and t_0, t_1, ..., t_n are continuous time samples being sampled at the same rate as the frame rate of the video.

The output *mashup video* $V = \left\langle (g_{t_0}, b_{t_0}), (g_{t_1}, b_{t_1}), ..., (g_{t_n}, b_{t_n}) \right\rangle$ is computed with the following general function σ:

$$(g_t, b_t) = \sigma((f_{1,t}, a_{1,t}), (f_{2,t}, a_{2,t}), ..., (f_{k,t}, a_{k,t})), \tag{6.1}$$

which takes as input the set of frames and audio chunk captured at the same time t from the k input video clips, and outputs a single frame g_t and its associated audio b_t. The term *time-synchronized* in the problem refers to the property that the same t is used in the output of σ as well as the set of inputs. We say the video source at time t (denoted $vid(t)$) for V is i if $g_t = f_{i,t}$ and the audio source for V at time t (denoted $aud(t)$) for V is i if $b_t = a_{i,t}$. A *video cut point* refers to t where $vid(t-1)$ is not equal to $vid(t)$.

Note that the mashup video contains an output frame for each time t as long as there is a non-empty frame in one of the input videos. This property is unlike the *video summarization* problem, where the output summarized video is typically shorter than the inputs, since subsequence of input frames may not be included in the output.

We next survey the existing work in the area, i.e., how σ selects g_t and b_t from a given pool of k videos. We organize Sects. 6.3–6.6 according to the building blocks shown in Fig. 6.2: preprocessing, quality analysis, context modeling, cinematography rules, and overall system. In addition, we will also discuss various techniques and datasets used to evaluate mashup frameworks in Sect. 6.7.

6.3 Preprocessing

The time-synchronized video mashup formulation requires that the input videos V_1, ..., V_k are temporally aligned. This can be viewed as shifting the frames captured from different devices such that the frames captured at the same time are labeled with

the same subscript t as $f_{i,t}$. Temporal alignment is also required to find scene/event related clips from a larger set of videos [9]. In addition, for aesthetic reasons, the video clips should be of the same format in terms of frame rate and resolution. In a studio setting, video clips are generally aligned and recorded in the same format. A majority of user-generated video clips, however, are recorded by amateur users with heterogeneous devices. For such video clips, we need a preprocessing step to prepare them for the mashup.

A naive approach to synchronize video clips is to use their recording timestamps, which is based on the clock on the device. It is, however, not uncommon to find devices with clocks that are out-of-sync. The audio samples and video frames recorded by the same device at the same time instance, on the other hand, are guaranteed to have the same recording timestamp, since they are based on the same clock.[1] Hence, a common approach to temporally align the video is to use the content (either audio, visual, or both) to synchronize the video clips across devices. Consequently, a number of content-based alignment techniques have been developed for video clip synchronization. The basic assumption these methods take is that all cameras are capturing the same environment, and therefore, the captured content can be matched across cameras for time alignment.

Sinha and Pollefeys [10] match dynamic object silhouette sequence to calculate the synchronization offset. The method is designed for surveillance camera networks and assumes that there are common objects visible in the video clips. In the case of mobile cameras, it is hard to detect accurate silhouettes of the objects. A feature tracking based method is proposed by Meyer et al. [11] to synchronize mobile camera videos. As the method matches feature trajectories across cameras, it is assumed that long feature trajectories are available in the videos. Similarly, in [12, 13], the authors track SIFT features in a video to obtain feature trajectories. All these methods are able to achieve a subframe alignment accuracy.

The visual processing based methods of video synchronization assume that the same object is visible in all video clips to be synchronized. This assumption fails very often in real scenarios where amateur users record an event using their mobile devices. In such cases, audio is particularly useful in synchronizing cameras with non-overlapping views.

The simplest approach to synchronize two audio tracks is cross-correlation. Hasler et al. [14] calculate cross-correlation of audio signals from two cameras to detect the time shift. A peak in the cross-correlation tells us the time shift between video clips. Kammer et al. [15] explore three features, spectral flatness, zero-crossing rate, and spectral energy; and finds that the spectral flatness feature produces the best synchronization. While cross-correlation works well in controlled environments, it is computationally expensive and performs poorly in outdoor environments where signal degradation is common.

[1] Note that the audio and video samples captured at the same time are *generated* at a different time at the source due to the difference in the speed of light and the speed of sound. Humans, however, have learned to compensate for the difference in normal settings.

Table 6.2 A given set of video clips can be synchronized using audio-based features or video-based features

Video-based methods		Audio-based methods	
Object/silhouette trajectory [10]	Mainly used with static surveillance cameras	Cross-correlation [14, 15]	Works in controlled environment and computationally expensive
Feature tracking [11, 13]	Works for static as well as mobile cameras	Audio fingerprinting [9, 18]	Good in outdoor scenarios where signal degradation is common

Audio fingerprinting is a more common approach to synchronize audio tracks. Shrestha et al. [16] use an audio fingerprinting method to align video clips; audio fingerprints are a sequence of 32-bit words, each representing 11.6 ms segment of audio [17]. These words are obtained by spectral–temporal analysis of audio. The time offset is calculated by matching the audio fingerprints of two audio tracks in terms of Hamming distance and bit error rate (BER). Similarly, in [9, 18], the authors calculate an audio fingerprint of each clip using short-term frequency spectrum. Guimaraes et al. [19] align videos using features of the associated audio. The alignment algorithm is based on perceptual time–frequency analysis [20]. In one of the recent approaches, Bano and Cavallaro [21] match audio chroma features across video clips.

A limitation of audio-based synchronization is that the sound source is assumed to be at the same distance from both the cameras whose videos are being synchronized. This assumption is generally false and introduces an error in the alignment. Such errors can be compensated if the location of sound source and cameras are known [14].

Almost all video synchronization methods are for offline applications, i.e., they assume the availability of full video in advance. In online applications, video synchronization is not required because the video clips are processed *as they are captured*. Table 6.2 lists four broad categories of video alignment methods. Because each work is evaluated on a different dataset with different performance measures, it is difficult to compare them objectively. Still, most state-of-the-art synchronization methods are subframe accurate.

There is almost no research work on how to choose an optimal format for a given set of video clips. In [22], all the videos are resampled at 25 frames per second and rescaled to a specific resolution (but did not mention how they choose the resolution). Wu et al. [23] resample the audio at 8 kHz and video at 25 frames per second. The video frames are resized to 640×360. The duration of mashup is determined based on the earliest and latest available video.

(a) Occlusion **(b)** Tilt

Fig. 6.3 **a** Occlusion of the stage area by other audience. **b** Camera tilt due to arm movement

6.4 Quality and Context Analysis

6.4.1 Quality Analysis

Different video clips may vary in quality. The same video clip may also have varied quality over time. To create a good quality mashup, therefore, we need to periodically check the quality of video clips and choose good quality segments. At any given time, video quality can be measured with respect to multiple characteristics given in Table 6.3. Some characteristics are determined at the frame level while others are determined at the video level. The frame-level characteristics mainly depend on the quality of the camera (sharpness and exposure), the quality of the compression algorithm (blockiness), and the position of the camera with respect to the light source (burned pixels, infidelity, and BRISQUE [22]) [6, 7, 23, 24].

Occlusion and tilt measure frame quality at a semantic level. Occlusion occurs when the region of interest (e.g., the stage area) is blocked by a person or an object [6]. Figure 6.3a shows an example frame in which a person's head is blocking stage area. Similarly, tilt occurs while recording using a handheld camera and the camera gets tilted because of tired recording arm [6, 8, 24]. Figure 6.3b shows an example of tilted video recording. In [26], the authors study how shakiness and occlusion are perceived by the users. It is found that both shakiness and occlusion affect the perceived quality negatively; the impact may vary for different genres; and shakiness and occlusion influence more negatively when they affect the RoI of the scene. With limited frame rate, the quick movement may also result in motion blur as shown in Fig. 6.4.

Shakiness and jerkiness artifacts are mainly found in mobile recorded videos because of the sporadic camera motion [23]. These characteristics are mostly independent of the content of the video. Nguyen et al. [6] separately determine the video quality and view quality. Video quality is measured in terms of blur, blockiness, burned pixels, illumination, and contrast; whereas view quality is measured in terms of shakiness, occlusion of the performance area, and camera tilt. Wilk et al. [24] exploit the gravity sensor found in the smartphones to measure shakiness.

Table 6.3 Metrics for video quality

Metrics	How is it measured?	Description
Brightness, illumination, and exposure	Frame analysis [6, 7, 23, 24]	The image should not be too bright or too dark
Sharpness (Blur)	Frame analysis [6, 7, 23]	The image should be sharp with least amount of blur
Contrast	Frame analysis [6]	High contrast is perceived to be good
Burned pixels	Frame analysis [6]	As few saturated pixels as possible
Blocking artifact	Frame analysis [6, 7]	Caused by high video compression ratio
BRISQUE [22]	Frame analysis [22]	This is a no-reference image quality measure
Infidelity	Frame analysis [23, 25]	Infidelity represent low contrast and colourlessness of the image
Occlusion	Frame analysis [6, 26]	Occurs when objects blocks the camera while recording
Tilt or orientation	Frame analysis [6, 8, 23], gravity sensor [24]	Occurs when the camera is not held upright during recording (e.g., due to a tired arm)
Shakiness	Video analysis [6–8, 22, 23], gravity sensor [24]	Caused by random and sudden camera motion
Jerkiness	Video analysis [23]	Jerkiness is also caused due to camera motion
Object and activity attributes	Video analysis [4, 27–30]	High number of objects, bigger bounding box, large amount of activity, best view, frontal face orientation, etc.
RoI	Smartphone compass signal [31], video analysis [24]	Cameras pointing directly toward the RoI are better

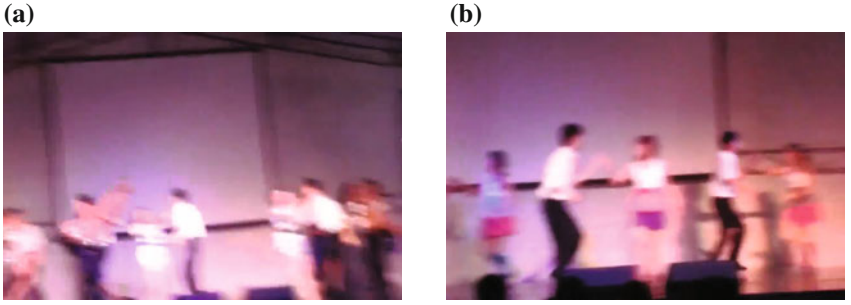


Fig. 6.4 Example of shaky frames resulting from quick arm movement

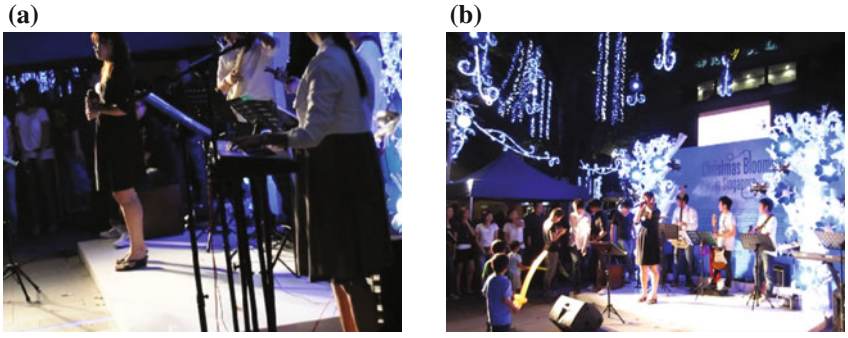


Fig. 6.5 From surveillance perspective, video **a** gives a better view than **b** as it shows more people with frontal faces

For surveillance applications, the cameras are generally static and focused. Therefore, the selection is mainly based on which camera provides the best view of the object or the activity [30]. The best view is defined in terms of object attributes such as the size of the bounding box, face orientation, body, pose, and object activity. In Fig. 6.5, the right image is better than the left from surveillance perspective because it shows more number of frontal faces. Daniyal and Cavallaro [27], Daniyal et al. [28] measure the view quality in terms of the number of objects, the size of objects, trajectory, location, etc. Activity and event score are also used to select the best view. In addition to object size, Gorshon et al. [29] also consider the motion direction in the view quality. In [4], each object is assigned an importance score by the user. The total score is the weighted sum of the quality of all objects in the view.

In many scenarios, it is not clear what objects or activities are interesting. For such scenarios, Region of Interest (RoI) is determined dynamically. Vihavainen et al. [31] measure the quality of a video based on how pointed the camera is toward the RoI, while RoI is determined based on the compass sensor signal of multiple cameras. Similarly, Arev et al. [8] calculate a joint attention map based on the intersection of the field of view of individual cameras. The view quality is measured in terms of the camera position in the 3D joint attention map. Wilk et al. [24] determine RoI by calculating the intersection of the field of view of individual cameras. The area where most cameras intersect is chosen as RoI.

6.4.2 Context Analysis

There are generally more than one video clips at any given time that meet the quality requirements. The mashup methods choose one of these video clips according to heuristics and cinematography rules. These rules require a knowledge of recording context. Recording context mainly includes the attributes of the camera, user, and

Table 6.4 List of context attributes that have been used by mashup methods. The context attributes can be static (for fixed cameras) or dynamic (for mobile cameras)

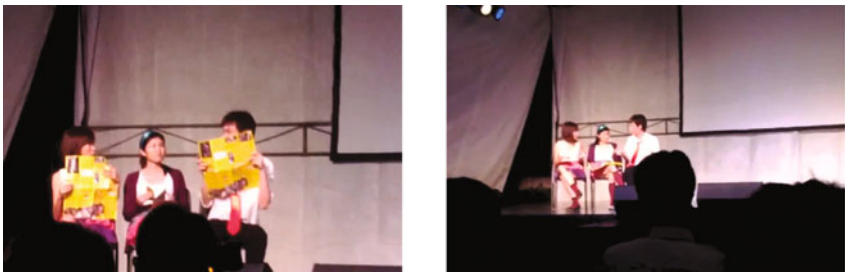
Attribute	Type	Description
History [5, 22]	Dynamic	Certain number (2–5) of previously selected video shots form history context
Site (e.g., surveillance site) [28]	Static	An environmental attribute that defines area being captured by cameras
Performance area [5]	Static	Performance area becomes the default RoI
Events [27, 31]	Dynamic	In many applications (such as sports), an event provides context for video selection
Recording angle [5]	Dynamic/Static	The angle with which a user is capturing the performance area of the RoI
Distance [5]	Dynamic/Static	The distance between the user and the performance area
Zoom level [5, 8]	Dynamic	The zoom level with which the camera is capturing the scene
Position/location	Dynamic/Static	The user's location in general in the recording environment (site)
Audio tempo [7]	Dynamic	The variation in the music tempo have been considered good points for deciding cut points
Sudden visual change [7, 31]	Dynamic	An indirect measure of change in event or activity
Silence/speech/music periods [22]	Dynamic	These attributes are used to choose suitable cut point
Instrument change [22]	Dynamic	Another attribute for to decide suitable video cut point

recording environment. Table 6.4 lists important context attributes that have been used in mashup frameworks.

There are two types of context attributes: static and dynamic. The static context is generally related to the site information and camera location (in the case of static cameras). In [28], the surveillance site layout is taken as static context information. The dynamic context is the video selection history, recording angle, zoom, event, etc. Saini et al. [5] take the distance of the camera from the stage and the recording angle as the dynamic context. In Fig. 6.6, the videos are assigned context *left, center, and right*, respectively, from left to right. Similarly, in Fig. 6.7 the left video is assigned

(a) Left **(b)** Right **(c)** Center

Fig. 6.6 Examples of a dynamic context. Videos are assigned left, center, and right context, respectively



(a) Near **(b)** Far

Fig. 6.7 Examples of a dynamic context. Video **a** is assigned *near* context; video **b** is assigned *far* context

Near context and the right video is assigned *Far* context. In addition, the recent video selection history also builds a context which is used to select diverse video views.

In [27], an occurrence of *important events* provides context for camera selection. The scenario considered is a basketball game and an attempt on the basket is considered an important event, which is detected by observing motion vectors. Vihavainen et al. [31] record electronic compass signals along with the video to obtain recording context. The authors use compass readings to calculate RoI and camera movements (horizontal and rotational). Camera panning instants are used to detect events. It is assumed that a camera pan operation is the result of the start of an interesting event; hence, this instant can be used to switch views. Wang et al. [4] also use the ball position in a soccer field as the main context for selecting best view. Shrestha et al. [7] take the audio tempo and visual appearance as context to choose suitable switching points. Similarly, Bano and Cavallaro [22] analyze audio to build up the context for choosing camera switching point. The authors use root-mean-squared value and spectral entropy of audio to detect silence/speech/music periods, and spectral centroid to detect instrument change in the music [32].

Analysis of video quality and recording context is not a completely solved problem. This problem area, however, has received much attention of the computer vision and multimedia research communities, and great strides have been made.

6.5 Cinematography Rules

In film-making, a number of video clips of a scene are recorded and a human editor picks segments from these video clips to make the final movie. The editors follow a number of cinematography rules while creating a movie [33]. For instance, one of the fundamental rules is that one should first look at the content of the video, and then the context, which means first find good quality videos and then apply formation rules. Mashup frameworks have explored incorporating a number of such rules to improve the quality of the mashups produced.

Saini et al. [5] learn cinematography rules from a set of popular professionally edited videos of live performances. The example videos are first divided into shots (or segments). Then, each shot is classified based on distance from the stage area (near–far) and recording angle (left–right–center). Finally, a hidden Markov model (HMM) is trained from the example videos with distance and angle as states and shot length as an observation. With this HMM, the method generates a continuous sequence of states (distance and angle) and shot lengths to be used in the mashup. Note that a separate HMM should be trained for each genre of videos.

Saini et al. [5] learn shot transition rules from the existing videos, Arev et al. [8] take guidelines from the literature [33]. The following cinematography rules are considered:

- Avoid jump cuts: Users perceive a jump in the video if we transition to a camera that records from almost the same perspective (angle and distance).
- 180 rule: Do not transit to a camera that is more than 180 degree apart from the current camera. It may create confusion to the viewer.
- Zoom: Depending on the main subject/area view, the videos are divided into a wide shot, long shot, medium shot, close-up, and extreme close-up. The videos with diverse zoom are selected avoiding too many or too large jumps.
- Shot duration: Minimum and maximum shot durations are defined. The method prevents cut before minimum duration and promotes after the maximum duration.
- Cut-on action: The method tries to switch to another camera when the action is at its peak.

Another important rule implicitly used is shot diversity. It is believed that a mixture of different types of shots is pleasing to the viewers. In [34], the authors propose method to automatically classify a given video as wide shot, medium shot, and close-up shot based on implicit cues of distance from RoI. The implicit cues used are color intensity, motion saliency, face size, etc. Carlier et al. [35] add diversity to the mashup by introducing virtual camera motion in the video in terms of zoom and pan. The camera operations are chosen based on a 3D interest map.

While some basic cinematography rules are universals, human professionals have often infused their own cinematography styles in producing mashups. Capturing and imitating such personal cinematography styles, which are sometimes fairly subtle, in a general manner with applications to a wide range of scenarios remain a challenging problem.

6.6 Overall Mashup Framework

As given in Eq. 6.1, our goal is to find a time-synchronized subset of audio and video chunks from the given set of video clips. In most cases, audio is taken from a separate source [19, 36] or it is obtained by stitching best quality audio segments from multiple tracks associated with video clips [22]. Every audio track has different types and amounts of noise along with the desired audio. Therefore, changing audio track too many times could be annoying to the users. A greedy approach is generally taken to maximize the quality with minimal number of segments [22]. A video mashup system, on the other hand, needs to answer two main questions: *Which video to choose at current time? When to switch to another video?* Factors considered while answering these questions include content quality, smoothness of transition, diversity, coverage, and cinematography rules.

In the case of surveillance, the only criterion used is content quality. Because cameras are generally static, the quality is mainly measured in terms of object appearance. Introducing too many transitions would cause distraction to the security operators. To avoid too many transitions, the methods keep the same camera unless there is a significant improvement in the view quality [28, 30]. In [27], the authors employ partially observable Markov decision process (POMDP) in the decision-making process so that switching does not occur too often, yet the selected view is best most of the time. Wang et al. [4] take a similar approach to create mashup of sports video clips recorded using static cameras. The authors first detect the quality of each view (video clip) in terms of object appearance, and then transit to a view that provides smooth transition while maximizing the view quality. To ensure smoothness, each view is assigned a transition cost in terms of view quality difference and visual angle difference. The cost decreases with view quality difference and increases with angle difference. In addition, a duration cost is also added to avoid frequent transitions.

In the above methods, a different video clip is chosen only when the current video quality goes below par; otherwise the same video clip is retained. When creating mashup of social videos (i.e., videos recorded for social sharing purposes), however, coverage and diversity are also equally important along with quality. Therefore, after pre-filtering the bad quality video clips [5, 23] and choosing a subset of good quality video clips according to cinematography rules [5], the next video clip is chosen to ensure coverage and diversity. Shrestha et al. [7] calculate the visual difference between current video clip and the next candidate video clips and choose the one with the highest visual difference. Saini et al. [5] classify video clips based on recording angle (left–right–center) and distance from the stage (near–far). To ensure diversity and coverage, video clips from difference classes are chosen over time. The exact sequence of class transitions (e.g., center–near to left–far) is learned from professionally edited videos in an HMM. Within the selected class, a video clip is chosen for the next segment that maximizes the diversity with respect to the recent history of five previously selected video clips.

Jiku Director 2.0 [37] adds diversity to the mashup by introducing virtual camera movement into the mashup in terms of zoom and pan. The region to zoom or pan to

is computed from the motion map of the videos. Carlier et al. [35] extends the concepts to 3D using a 3D interest map. The 3D interest map of the scene/event being captured is calculated as intersection of visual cones of the cameras. To zoom and pan, however, the resolution of the mashup needs to be smaller than the resolution of the input video clips. Arev et al. [8] divide the video clips into wide shot, long shot, medium shot, close-up, and extreme close-up; and choose diverse shot types in the mashup. The method encodes the cinematography rules and diversity constraints into an optimization function, which is then minimized using dynamic programming to obtain the final sequence of video clips. Wu et al. [23] also select video segments to maximize quality, diversity, camera motion, and semantic completeness. Particularly, the method prefers videos with less or no motion around cut points.

The cut point, or time instant we choose to switch from one video clip to another video clip, also impacts the perceived mashup quality. In traditional film-making, every cut point is motivated by some audio or video property such as camera motion, occlusion on area of interest, silence to voice transition, and music tempo [33]. In [31], cut points are determined based on camera panning instants. It is assumed that a camera pan signifies the start of an event. Arev et al. [8] choose the cut point when the ongoing action is in its peak state, which is detected by tracking the movement of the 3D joint attention map.

Audio has significant impact on perceived smoothness of camera/shot transitions [38]. Shrestha et al. [7] consider audio tempo along with change in brightness to choose cut points. Similarly, Bano and Cavallaro [22] choose cut points based on changes in audio (e.g., voice/silence to music and instrument change). Wu et al. [23] first detect suitable cut points and then choose appropriate shot from the pool of shots. The cut points are determined based on both audio tempo and audio energies. Audio tempo is used to make sure that a high pace audio has more switching points than a low pace audio. The audio energy is used to avoid switching when somebody is talking/singing. Table 6.5 lists various audio- and video-based criteria used to choose a cut point.

Another governing factor in choosing cut point is shot length. It has been found that while segments that are too long are boring, segments that are too short are incomprehensible by the viewers. Therefore, mashup frameworks choose the shot or segment length within a limited range [5, 7, 22, 31]. The exact shot length distribution depends on the genre of the videos. Shrestha et al. [7] choose a shot length

Table 6.5 Various audio- and video-based criteria used to choose cut points in mashups

Audio-based criteria	Video-based criteria
• Camera motion [31]	• Audio tempo [7, 23]
• Object motion [23]	• Audio energy [23]
• Occlusion of ROI [28]	• Voice to music change [22]
• Change in brightness [7]	• Silence to music change [22]
• Learning from professional videos [5]	• Change of instrument [22]

between 3 and 7 s for live concert videos. The exact shot length is determined based on an optimization function consisting of quality, diversity, and segment length. Saini et al. [5] learn shot lengths for live dance performances from existing professional music videos. The authors found the shot length to be between 1 and 7 s, with an average of 2.3 s per shot. Similarly, Bano and Cavallaro [22] choose to keep the video lengths between 3 and 10 s for normal user-generated videos.

Different levels of user involvement have been recommended in different mashup frameworks. Shrestha et al. [7] allow users to assign scores to each video clip as a user preference. These user preferences are included in the optimization function used to choose mashup segments. The Jiku Director [6] extends MoViMash [5] to provide an interface for the users to choose input video clips to the mashup algorithm. In this way, the Jiku Director provides an opportunity to personalize the mashup by selecting the desired video clips. *MyVideo* [19, 36] system annotates each video clip as people, song, instrument, etc. and provides an interface for users to manually create mashup. Users' queries for desired video clips are termed *personal* videos in the work. The system returns a subset of video clips with an associated rank according to the relevance with the query. Users now choose video clips to create a mashup. Arev et al. [8] provide an option to the users to define an *important character* of the event, which is considered while choosing video segments along with other factors. Wilk et al. [24] automatically select good quality video clips and present those to the users to create a mashup with a crowdsourcing approach. Users are shown a set of video clips from which they choose the best quality video and audio for the next shot. If the same video clip is chosen as the best again, the second best video clip is selected for the mashup to ensure diversity. Users also choose the cut points within a range that depends on the genre of the video.

6.7 Evaluation

Quantitative assessment of a given mashup is very challenging. There is no unique mashup that can be considered *ideal*. Different combinations of video and audio segment can be equally good. Therefore, the mashup frameworks are mostly evaluated subjectively with user studies. Shrestha et al. [7] compared their mashup with two other mashups, naive and manual. The dataset consisted of videos recorded during three concerts. Hence, each user watched nine mashup videos and rated nine statements related to diversity, visual quality, and pleasantness, on a seven-point Likert scale. The videos were presented in random order to each user. A similar approach is taken to evaluate the Jiku Director [6] on the Jiku Mobile Video Dataset [39]. Three versions of video mashups are shown simultaneously to the users on a web page. Users watched the videos in a random order and rated five statements related to visual quality, coverage, and interestingness, on a scale of 1 to 5.

Arev et al. [8] evaluated their work on scenes taken from their own as well as two published datasets, [40, 41]. The evaluation only describes how well the method follows the individual steps of the framework. The final output quality is not measured.

Table 6.6 Important mashup evaluation datasets

Work	Data source	Type	Events	Cameras	Availability
First-fit [7]	Smartphone cameras	Live concerts	3	4 to 5	On request
Arev et al. [8]	Handheld and head-mounted, smartphone cameras	Sports, performances, and social gatherings	8	3 to 18	On request
ViComp [22]	Handheld cameras	Live concerts	11	4 to 12	On request
Wilk et al. [24]	Smartphone cameras	Sports, music, live performances	5	4 to 12	On request
MoVieUp [23]	YouTube	live concerts	6	5 to 27	On request
Wang et al. [4]	Statically placed cameras	Sports	2	10 to 11	On request
Jiku mobile video dataset [6]	Smartphone cameras	Live performances	45	3 to 15	**Available Online**[a]

[a] http://www.jiku.org/datasets.html

Bano and Cavallaro [22] evaluate their mashup framework on 13 events taken from Nickelback concert, Evanescence concert, FirsFit dataset [7], and the Jiku Mobile Video Dataset [39]. Users are shown three videos on a web page simultaneously. The users watch the videos and rank them on a scale of 1 (best) to 3 (worst). Wilk et al. [24] also use their own dataset of five events to evaluate their mashup system. Users' rate perceived the quality of four mashups: crowdsourced, human-edited, only quality-based, and random, on a scale of 1 to 5.

In contrast to earlier research, Wu et al. [23] collect multiple recording of the same event from YouTube. The authors evaluate audio quality as well as video quality with a user study where users' rate mashups on a scale of 1 to 5 in response to various questions related to quality, diversity, and cut point suitability. Wang et al. [4] create a ground truth mashup based on user recommended viewpoints. The quality of a given mashup is measured based on how similar the video segments and cut points are with the user created mashup. The authors evaluate their video mashup (viewpoint recommendation) method on a dataset of two soccer games.

Table 6.6 lists important datasets that are used to evaluate video mashup frameworks.

Finally, building an objective measure of evaluation is very useful but challenging. The major challenge in building a generic objective quality measure is the variable requirements of the end application of the mashup. For surveillance application, information and event coverage are important, whereas in video broadcast applica-

tion, aesthetics are important. Therefore, the performance evaluation procedure as well as the quality measures depends on the application scenario.

6.8 Future Research Directions

This section poses several open research challenges on the topic of automated time-synchronized video mashup, with the goal of making this a commodity service useful to end users. Three challenges that we feel are most important and interesting are discussed in depth in Sects. 6.8.1–6.8.3, while other research challenges are discussed in Sect. 6.8.4.

6.8.1 Narrative-Preserving Mashup

The current mashup frameworks are designed to optimize quality, diversity, coverage, and aesthetics. While surveillance and sports mashups maximize the object appearance, the social mashups focus on diversity and aesthetics to make the mashup interesting. If we look at the videos uploaded on the Internet, particularly social videos, they contain a scene narrative or a story. The scene would contain a series of sub-events and important characters that make up a complete story.[2] This scene narrative is mostly ignored in the current mashup works. For example, the current works would create an interesting mashup of a birthday video that would contain all good quality segments but miss important events of cutting the cake and blowing the candles.

Ideally, the mashup frameworks should identify important characters and sub-events of a scene and create a mashup that revolves around these entities. It is extremely hard to obtain a generic set of sub-events that would apply to all scenarios. Therefore, each genre will have its own customized mashup framework that identifies the narrative requirements of that genre. The researchers would need to solve a number of multimedia content understanding problems to develop narrative-preserving mashup frameworks. For instance, to create a mashup of birthday party video, we need to identify the videos in which the person having the birthday is the main character, i.e., stands out from others. Similarly, for each genre, we need to learn from professionally edited videos and generate character–event compositions that are most pleasing to the viewers.

Some existing work toward solving this problem include [36, 43]. The multimedia research community has also been working toward deeper understanding of multimedia content. A holistic framework for producing narrative-preserving mashup, however, remains open.

[2] See [42] for a discussion on narration, story, and events.

6.8.2 Geographically Distributed Event

Most mashup frameworks assume that the event is confined to a small area with overlapping camera views. Many of the mashup building blocks have this assumption. For example, synchronization modules assume that all cameras are viewing the same scene and within listening range of the same main audio source. Similarly, many context parameters, such as viewing angle and distance, are defined with respect to a particular stage area. In real life, however, there are a number of events that spread over a larger area. For example, a parade and an art exhibition take place over a wide area. For such events, identifying the exact event area and finding relevant videos itself is a challenging task. Consequently, a new set of mashup composition rules is required. The video clips may not have overlapping views, and the common element of audio may be very weak. Because different cameras can be at different distances from the sound source, the audio-based synchronization is also difficult. The traditional mashup frameworks would need to extend the definitions of *context, diversity*, and *coverage* for geographically distributed events.

For geographically distributed events, it is very important to obtain a 3D model of the event area to build a useful scene context. Within that 3D model, the camera locations and viewing angles can be used to find interesting events and ensure coverage/diversity. The cameras are, however, mostly mobile. Building 3D context using mobile cameras is a very hard problem. Multimedia techniques can be very handy in these situations. For example, GPS and compass sensors can be used to localize cameras. For indoor events, GPS may not work and advanced methods are required for localization. For events that spread over a very large area, a single mashup may not meet requirements of all users. An interactive mashup, where users can choose the viewing perspective, is another interesting research direction.

6.8.3 Real-Time Mashup

Real-time video mashup is useful for broadcast of live events. It poses two challenges. First, the mashup decision to choose the next segment has to be done online—only based on the analysis of the videos that have been processed so far. These methods implicitly assume that the near future is equal to near past. With this assumption, current context and quality are taken as the representative of the next segment and correspondingly the most appropriate video clip is selected. This assumption, however, does not hold in many cases. Therefore, we need more sophisticated models that can make a better prediction of the context and quality of each individual video. Instead of assuming the past to be equivalent to the future, models can be trained to learn the pattern that is followed in the past to predict future.

Second, real-time mashups require the mashup processing and decision to be made within a reasonable amount of time (in order of seconds). In current online mashup systems, the video and audio are processed only at the time of choosing the

next video clip. Therefore, the processing overhead is bursty in nature. These bursts can hit the processing bottleneck and result in artifacts such as freeze frames. In order to avoid these bursts, the processing needs to be more distributed over time. If the processing is not done close to the cut point, however, the analysis results may not represent the future well. Therefore, there is a tradeoff between how distributed the processing is and how accurately the analysis results represent the future segment. Interestingly, this bursty nature would only delay the shot transition instant, affecting the overall quality of the mashup. But it would not introduce any delay in the output mashup as we are not modifying the content but only selecting from the pool.

Real-time mashup application requires data to reach the processing unit quickly. When the number of videos is large, such as the video clips recorded at a parade, the bandwidth would hit the bottleneck. In such scenarios, which are very common given a large number of smartphone users, video clips incur a variable delay to reach the processing server, which poses additional challenges in video synchronization. Furthermore, in order to avoid the processing bottleneck at the server, we need to distribute the processing tasks among the recording devices. This can be followed by selective streaming of the video clips. Only videos that have the right context and meet the quality requirements can be streamed. Mobile devices such as smartphones, however, have limited processing power and battery life. Therefore, there is a tradeoff between mobile resources and bandwidth.

6.8.4 Other Challenges

With the introduction of many new video recording devices, the video sources can be heterogeneous in nature. The video clips vary in resolution, frame rate, encoding method, and overall quality. Current methods take a conservative approach to combine these heterogeneous video clips. They choose the smallest available resolution, frame rate, etc. In this approach, we loose much important information available in the cropped areas or dropped frames. Intelligent methods are required to exploit all the information available and still merge the videos into a common format mashup. For example, instead of cropping, a window can pan over important areas of the video.

Being a relatively new research area, this field lacks concrete definitions of the terms frequently used in the mashup frameworks. For instance, there is a need to concretely define diversity, coverage, context, semantic completeness, and interestingness. The concrete definition of these terms will help synergize the efforts of multiple researchers. In addition, a user perception model that can be dynamically used to emulate user response for candidate video clips is needed. While assessing the quality of videos, most frameworks rely on frame-level attributes. For example, sharpness is calculated by processing the current frame. This value is taken as a representative of the whole video, but this may not be accurate. Instead, we should obtain video-level quality attributes. Also, there are many possible quality attributes

for video, audio, and image, to use—which attributes are the most relevant and how to combine those attributes in an optimal way remain as challenging open problems.

Finally, a better platform is required for evaluation of mashup frameworks. Current methods rely heavily on user studies. User studies have many hidden factors, which vary with each set of users and experimental setup. Therefore, current evaluation methods are not adequate to compare two given frameworks. Ideally, there should be an objective evaluation mechanism. It is, however, hard to define a benchmark mashup because different mashups with entirely different video compositions can be equally interesting to the users.

So far, mashup frameworks have completely ignored the privacy aspect. Video clips recorded at public events may contain individuals who do not want to be part of the mashup. A challenge is to minimize the privacy loss of such individuals while still preserving the aesthetics of the mashup.

6.9 Conclusions

Video mashup is an efficient and interesting way of visualizing simultaneous video recordings of an event. In this chapter, we have provided a formal definition of video mashups and identified important building blocks for mashing up time-synchronized videos. The mashup framework typically begins with a preprocessing phase where all video clips are brought into a common format and timeline. These video clips are analyzed to obtain contextual parameters and quality attributes. The final mashup is produced by taking segments from individual video clips according to the quality, context, and cinematography rules. The chapter provides a brief review of current research works in each of these subproblems. The main future research directions are identified, and they include preserving event narrative in mashups, dealing with geographically distributed events, and development of a real-time mashup system.

Definitions

Video Mashup A video that is produced by concatenating video segments cut from input video clips recoded at the same event from different views.

Time-Synchronized Video Mashup A video mashup that follows the same timeline as the event itself.

Asynchronous Video Mashup A video mashup that does not follow the same timeline as the event itself, and can be shorter or longer than the actual event. An example of an asynchronous video mashup is a summary video.

Cut point A time point in a video mashup when we choose to switch from one input video clip to another.

Shot Length The video playback time between two cut points. In other words, the length of a video segment from the same input clip included in the video mashup.

References

1. Nakano, T., Murofushi, S., Goto, M., Morishima, S.: Dancereproducer: an automatic mashup music video generation system by reusing dance video clips on the web. In: Sound and Music Computing Conference (SMC), pp. 183–189 (2011)
2. Fu, Y., Guo, Y., Zhu, Y., Liu, F., Song, C., Zhou, Z.H.: Multi-view video summarization. IEEE Trans. Multimedia (TOMM) **12**(7), 717–729 (2010)
3. Pritch, Y., Ratovitch, S., Hende, A., Peleg, S.: Clustered synopsis of surveillance video. In: IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 195–200. IEEE (2009)
4. Wang, X., Hirayama, T., Mase, K.: Viewpoint sequence recommendation based on contextual information for multiview video. IEEE Multimedia **22**(4), 40–50 (2015)
5. Saini, M.K., Gadde, R., Yan, S., Ooi, W.T.: Movimash: online mobile video mashup. In: ACM International Conference on Multimedia (MM), pp. 139–148. ACM (2012)
6. Nguyen, D.T.D., Saini, M., Nguyen, V.T., Ooi, W.T.: Jiku director: a mobile video mashup system. In: Proceedings of the 21st ACM International Conference on Multimedia, pp. 477–478. ACM, Barcelona, Spain (2013)
7. Shrestha, P., Weda, H., Barbieri, M., Aarts, E.H., et al.: Automatic mashup generation from multiple-camera concert recordings. In: Proceedings of the 18th ACM International Conference on Multimedia, pp. 541–550. ACM, Firenze, Italy (2010)
8. Arev, I., Park, H.S., Sheikh, Y., Hodgins, J., Shamir, A.: Automatic editing of footage from multiple social cameras. ACM Trans. Grap. (TOG) **33**(4), 81:1–81:11 (2014)
9. Su, K., Naaman, M., Gurjar, A., Patel, M., Ellis, D.P.: Making a scene: alignment of complete sets of clips based on pairwise audio match. In: Proceedings of the 2nd ACM International Conference on Multimedia Retrieval, p. 26. ACM, Hong Kong (2012)
10. Sinha, S.N., Pollefeys, M.: Synchronization and calibration of camera networks from silhouettes. In: International Conference on Pattern Recognition (ICPR), pp. 116–119. IEEE (2004)
11. Meyer, B., Stich, T., Magnor, M.A., Pollefeys, M.: Subframe temporal alignment of non-stationary cameras. In: British Machine Vision Conference (BMVC), pp. 1–10 (2008)
12. Caspi, Y., Simakov, D., Irani, M.: Feature-based sequence-to-sequence matching. Int. J. Comput. Vis. (IJCV) **68**(1), 53–64 (2006)
13. Elhayek, A., Stoll, C., Kim, K., Seidel, H., Theobalt, C.: Feature-based multi-video synchronization with subframe accuracy. Pattern Recogn. 266–275 (2012)
14. Hasler, N., Rosenhahn, B., Thormahlen, T., Wand, M., Gall, J., Seidel, H.P.: Markerless motion capture with unsynchronized moving cameras. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 224–231. IEEE (2009)
15. Kammerl, J., Birkbeck, N., Inguva, S., Kelly, D., Crawford, A.J., Denman, H., Kokaram, A., Pantofaru, C.: Temporal synchronization of multiple audio signals. In: IEEE International Conference on Acoustics. Speech and Signal Processing (ICASSP), pp. 4603–4607. IEEE, Firenze, Italy (2014)
16. Shrestha, P., Barbieri, M., Weda, H.: Synchronization of multi-camera video recordings based on audio. In: Proceedings of the 15th ACM International Conference on Multimedia, pp. 545–548. ACM, Augsburg, Germany (2007)
17. Haitsma, J., Kalker, T.: A highly robust audio fingerprinting system with an efficient search strategy. J. New Music Res. **32**(2), 211–221 (2003)
18. Cremer, M., Cook, R.: Machine-assisted editing of user-generated content. In: IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics, pp. 725,404–725,404–410 (2009)
19. Laiola Guimaraes, R., Cesar, P., Bulterman, D.C., Zsombori, V., Kegel, I.: Creating personalized memories from social events: community-based support for multi-camera recordings of school concerts. In: Proceedings of the 19th ACM International Conference on Multimedia, pp. 303–312. ACM, Scottdale, AZ, USA (2011)

20. Korchagin, D., Garner, P.N., Dines, J.: Automatic temporal alignment of av data with confidence estimation. In: IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 269–272. IEEE (2010)
21. Bano, S., Cavallaro, A.: Discovery and organization of multi-camera user-generated videos of the same event. Inf. Sci. **302**, 108–121 (2015)
22. Bano, S., Cavallaro, A.: Vicomp: composition of user-generated videos. Multimedia Tools Appl. (MTAP) **75**(12), 1–24 (2015)
23. Wu, Y., Mei, T., Xu, Y.Q., Yu, N., Li, S.: Movieup: Automatic mobile video mashup. IEEE Trans. Circ. Syst. Video Technol. **25**(12), 1941–1954 (2015)
24. Wilk, S., Kopf, S., Effelsberg, W.: Video composition by the crowd: a system to compose user-generated videos in near real-time. In: Proceedings of the 6th ACM Multimedia Systems Conference, pp. 13–24. ACM, Portland, USA (2015)
25. Mei, T., Hua, X.S., Zhu, C.Z., Zhou, H.Q., Li, S.: Home video visual quality assessment with spatiotemporal factors. IEEE Trans. Circ. Syst. Video Technol. (CSVT) **17**(6), 699–706 (2007)
26. Wilk, S., Effelsberg, W.: The influence of camera shakes, harmful occlusions and camera misalignment on the perceived quality in user generated video. In: IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE, Chengdu, China (2014)
27. Daniyal, F., Cavallaro, A.: Multi-camera scheduling for video production. In: Conference for Visual Media Production (CVMP), pp. 11–20. IEEE (2011)
28. Daniyal, F., Taj, M., Cavallaro, A.: Content and task-based view selection from multiple video streams. Multimedia Tools Appl. (MTAP) **46**(2–3), 235–258 (2010)
29. Goshorn, R., Goshorn, J., Goshorn, D., Aghajan, H.: Architecture for cluster-based automated surveillance network for detecting and tracking multiple persons. In: ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), pp. 219–226. IEEE (2007)
30. Jiang, H., Fels, S., Little, J.J.: Optimizing multiple object tracking and best view video synthesis. IEEE Trans. Multimedia (TOMM) **10**(6), 997–1012 (2008)
31. Vihavainen, S., Mate, S., Seppälä, L., Cricri, F., Curcio, I.D.: We want more: human-computer collaboration in mobile social video remixing of music concerts. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 287–296. ACM (2011)
32. Lerch, A.: An introduction to audio content analysis: applications in signal processing and music informatics. Wiley (2012)
33. Dmytyk, E.: On film editing: an introduction to the art of film construction (1984)
34. Canini, L., Benini, S., Leonardi, R.: Classifying cinematographic shot types. Multimedia Tools Appl. (MTAP) **62**(1), 51–73 (2013)
35. Carlier, A., Calvet, L., Nguyen, D.T.D., Ooi, W.T., Gurdjos, P., Charvillat, V.: 3d interest maps from simultaneous video recordings. In: ACM International Conference on Multimedia, pp. 577–586. ACM (2014)
36. Zsombori, V., Frantzis, M., Guimaraes, R.L., Ursu, M.F., Cesar, P., Kegel, I., Craigie, R., Bulterman, D.C.: Automatic generation of video narratives from shared ugc. In: Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, pp. 325–334. ACM, Eindhoven, Netherlands (2011)
37. Nguyen, D.T.D., Carlier, A., Ooi, W.T., Charvillat, V.: Jiku director 2.0: a mobile video mashup system with zoom and pan using motion maps. In: Proceedings of the ACM International Conference on Multimedia, pp. 765–766. ACM, Orlando, FL, USA (2014)
38. Beerends, J.G., De Caluwe, F.E.: The influence of video quality on perceived audio quality and vice versa. J. Audio Eng. Soc. (AES) **47**(5), 355–362 (1999)
39. Saini, M., Venkatagiri, S.P., Ooi, W.T., Chan, M.C.: The jiku mobile video dataset. In: ACM Multimedia Systems Conference (MMSys), pp. 108–113. ACM (2013)
40. Ballan, L., Brostow, G.J., Puwein, J., Pollefeys, M.: Unstructured video-based rendering: interactive exploration of casually captured videos. ACM Trans. Graphics (TOG) **29**(4), 87. ACM (2010)
41. Park, H.S., Jain, E., Sheikh, Y.: 3D social saliency from head-mounted cameras. In: Advances in Neural Information Processing Systems (NIPS), pp. 431–439 (2012)

42. Nack, F.: Event and story: an intricate relationship. In: Proceedings of the 2011 Joint ACM Workshop on Modeling and Representing Events, J-MRE '11, pp. 49–50. ACM, NY, USA (2011). https://doi.org/10.1145/2072508.2072520
43. Frantzis, M., Zsombori, V., Ursu, M., Guimaraes, R.L., Kegel, I., Craigie, R.: Interactive video stories from user generated content: a school concert use case. In: International Conference on Interactive Digital Storytelling, pp. 183–195. Springer, Berlin (2012)

Chapter 7
MediaSynch Issues for Computer-Supported Cooperative Work

Ketan Mayer-Patel

Abstract Computer-supported Cooperative Work (CSCW) systems are often complex distributed applications that incorporate a number of different tools working in concert to support goal- and task-driven collaborations that involve multiple sites and participants. These systems exhibit a wide variety of different system and communication architectures both within and between participating sites. In this chapter, we describe and characterize CSCW systems with respect to media synchronization requirements, identify a number of common challenges that arise as a result, and review a variety of protocol coordination techniques and mechanisms that can be brought to bear.

Keywords CSCW · Distributed systems · Protocol coordination

7.1 Introduction

Computer-supported Cooperative Work (CSCW) is a broad class of multimedia applications intended to support the communication needs of distributed teams working with a common set of tools, documents, and resources in a task- and goal-oriented context. Typically, these applications involve confederating together a number of different kinds of multimedia tools including live, interactive video and audio conferencing, shared editing of textual documents, multi-view rendering of graphical objects and/or 3D environments, annotation and note-taking tools, presentation broadcasting, screen sharing, and others. Often, these applications include specialized tools intended to support a particular type of endeavor (business meeting, scientific collaboration, team-based programming, etc.).

CSCW systems are often characterized by loosely coupled, independent tools with a wide spectrum of end-to-end communication needs. Communication priorities are often contextual and highly dynamic, accounting for task-specific and

K. Mayer-Patel (✉)
University of North Carolina, Chapel Hill, NC, USA
e-mail: kmp@cs.unc.edu

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_7

goal-specific needs. As such, there are may be a wide spectrum of synchronization requirements ranging from tight and interactive (e.g., video conferencing), loose and noninteractive (e.g., presentation broadcasting), loose but consistent (e.g., shared editing), and asynchronous (e.g., annotations). Furthermore, the mechanisms for handling synchronization in CSCW systems may need to account for a wide spectrum of interface modalities among participants. For example, some users may be in a shared, resource-rich environment, such as a multiuser immersive CAVE while other participants are participating via a resource-limited environment such as a mobile device.

The media synchronization challenge of CSCW systems involves managing the distribution, transport, and rendering of data while adapting to varying network conditions (i.e., jitter, delay, and bandwidth). This is particularly challenging in CSCW systems because the individual component tools and applications may or may not be aware of their roles within the larger system. Another challenge for CSCW systems is managing the use of a mix of provisioned and best-effort resources. Finally, CSCW systems must manage the issue of "fair-use" of best-effort resources both among streams of data within the system as well as between the system as a whole and other traffic competing for network resources.

To address these challenges, researchers have explored a number of mechanisms for providing the coupled tools within a CSCW system with a coordinated understanding of the overall state of the system. This coordinated understanding must occur both in terms of current system-wide priorities and needs as well as the availability and state of shared resources such as network bandwidth. There is a particular focus on coordinating the distribution and transmission of media data because of the need to minimize (or at least mitigate) cross-stream interference. In general, CSCW systems require largely decentralized solutions that can be flexibly adapted to incorporate new tools and configurations without extensive reengineering of the system at large. Used effectively, these mechanisms can be used to promote independent tool-level adaptation that achieves the best global system-level response to highly dynamic user-level priorities and available bandwidth.

In this chapter, in addition to describing and characterizing the media synchronization challenges faced by CSCW systems, we will review the evolution of multistream protocols and protocol coordination techniques in the literature including Structured Stream Transport (SST), Congestion Manager (CM), TCP trunking, and Coordination Protocol (CP).

7.2 Overview of CSCW Systems

CSCW first emerged during the 1980s as early communication and planning tools such as email and calendaring began to become more widespread in business and industry [1]. Research in the area generally concentrated on the design of tools to support collaborative work processes as informed by social theory and

organizational psychology. As technology advanced and the Internet developed, research into CSCW systems evolved to include real-time communication, such as video conferencing and interactive meeting spaces. An in-depth history of the field and its development can be found in [2].

From a systems perspective, in particular, with regard to media synchronization, CSCW systems have much in common with other large-scale distributed multimedia applications, such a telepresence, tele-immersion, shared 3D visualization, and distributed virtual environments. In fact, regarding such systems as CSCW systems is reasonable, especially when their design is informed by task-domain requirements.

A useful classification of CSCW systems and their characterization with regard to synchronicity and architecture is described by Rodden [3]. Figure 7.1 illustrates an updated version of this classification, modified to use broader labels to reflect modern terminology and examples. The classifications are characterized both with respect to both the synchronicity of interactions and latency between participants.

The four categories of CSCW applications identified are:

Messaging
Messaging systems are characterized by asynchronous communication among remote participants. Email, newsgroups, and bulletin boards are some of the oldest and most obvious examples. More modern examples include Slack and Microsoft Teams. Generally, CSCW systems will include some sort of asynchronous messaging component within a broader framework of tools. These tools may maintain state with regard to the order and relationship of messages to each other (i.e., threads) and which participants have seen which messages. Note that in Rodden's original classification, such systems were included as a type of conferencing system.

Conferencing
Conferencing systems provide structured, synchronous communication between participants. Video and audio conferencing are obvious examples of these kinds of systems. Screencasting applications in which either the contents of a participant's entire display or specific windows associated with particular applications are shared with remote participants are also in this category. Modern systems in this category

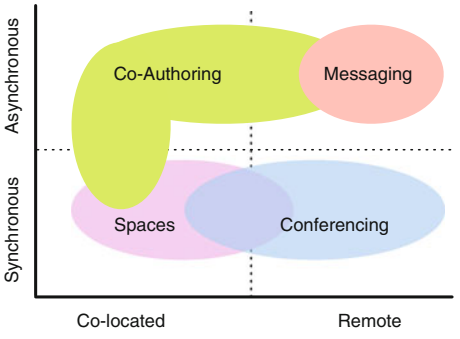


Fig. 7.1 CSCW system classification

include Google Hangouts, WebEx Meetings, Skype, BlueJeans, Twitch, and IBM Sametime.

Co-Authoring
Co-authoring systems support collaborative editing and annotation. In Rodden's original classification, these systems were categorized as only asynchronous. Modern co-authoring systems, however, often support both synchronous and asynchronous interactions among participants. Maintaining order consistency across multiple simultaneous updates is an important consideration for these applications. Examples of co-authoring tools include Google Docs and Microsoft Office365.

Spaces
CSCW systems in this category are generally a sophisticated mixture of messaging, conferencing, and co-authoring applications paired with domain-specific applications specialized to the needs of the participants. One example is a smart meeting room with a panoramic field of view cameras, large touch-sensitive displays, and specialized meeting capture and summarization software. Tele-immersive spaces with 3D capture and rendering capabilities tailored to support a specific activity such a remote medical consultation or surgery is another example. Also in this category are CSCW systems that create shared virtual environments without a specific associated physical manifestation. Commercial systems include Cisco Collaboration Meeting Rooms and Polycom RealPresence,

In this chapter, we focus primarily on mechanisms of media synchronization that can be brought to bear on CSCW systems with complex interstream semantics such as the systems in the spaces category described above. In large part, the media synchronization challenges faced in these systems is a superset of simpler CSCW systems in the other three categories.

7.3 Communication Architectures

The architecture of a CSCW system has two components: an intrapersonal architecture and an interpersonal architecture. Intrapersonal architecture refers to the relationship of programs and processes part of a CSCW system that supports a specific participant or shared physical space. These architectures are illustrated in Fig. 7.2 and include:

Monolithic
One or more processes executing on a single host or device with shared access to common hardware and networking resources. Systems that use a monolithic architecture must be provisioned with computational resources (i.e., processor speed and memory) that are capable of supporting simultaneous operation of all the different tools involved. Real-time operating system support for process scheduling may be required to provide adequate performance and isolation. On the other hand,

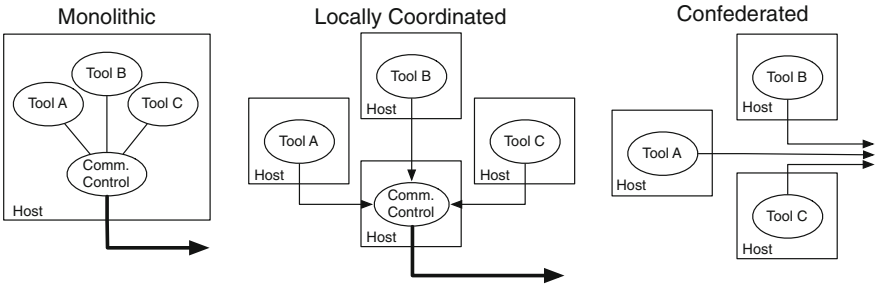


Fig. 7.2 Intrapersonal architectures

monolithic designs allow for tightly integrated tools with near-zero control latency and shared memory.

Locally Coordinated
Multiple processes on separate hosts with well-provisioned communication explicitly coordinated by a local coordinator or a distributed coordination algorithm. Within a locally coordinate architecture, each component tool or application can be matched with hardware resources most appropriate for that specific component within the CSCW system. Well-provisioned local communication resources can ensure reasonably low-latency communication among tools, although the principle of end-to-end protocol semantics must generally be abandoned.

Confederated
Multiple processes on separate hosts with independent end-to-end connections and streams without explicit coordination. The primary advantage of a confederated architecture is flexibility, modularity, and the ability to include independent and third-party tools and applications developed independently of the CSCW system as a whole. Furthermore, each tool is able to employ the most appropriate end-to-end protocol for its specific media type and task model. Interstream coordination and control, however, becomes more challenging.

The interpersonal architecture of a CSCW system pertains to the communication between participants and/or shared spaces. These architectures are illustrated in Fig. 7.3 and include:

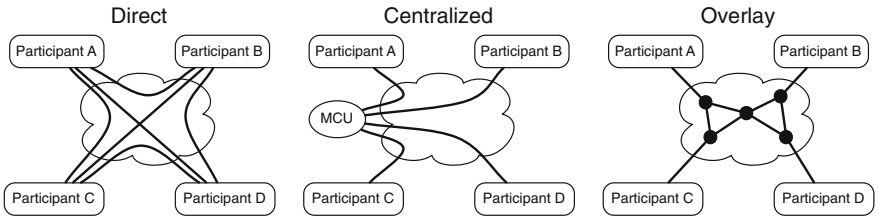


Fig. 7.3 Interpersonal architectures

Direct
Communication between participants is end-to-end and direct. Streams within a particular application or tool use end-to-end protocols. Minimized latency between any two participants is the main advantage of a direct interpersonal architecture. Distribution of media to two or more participants, however, requires redundant duplicate transmission or the use of end-to-end IP multicast. Last-mile access bandwidth is often a limiting resource, the use of which must be carefully coordinated amongst streams arriving for a participant.

Centralized
Communication between participants is mediated through a central in-network service, sometimes called a multipoint control unit (MCU). A variant of this architecture is realized when a particular participant also acts as an MCU. A centralized architecture provides a common point for control, distribution, and synchronization of streams within the system. Drawbacks include the need to provision the MCU with enough resources to process all of the streams in the system, requiring the MCU to support twice the bisection bandwidth of the system as a whole, difficulties in locating the MCU such that all participants experience similar interaction latency, and introducing a single point of failure.

In-Network Overlay
Communication between participants is routed through an overlay of in-network nodes that coordinate to route communication between participants and manage network resources. A variant of this architecture is realized when the participants themselves form a dynamic peer-to-peer overlay network. Overlay networks provide a balance between direct and centralized architectures and can very effectively and efficiently provide one-to-many and many-to-many multicast services. Furthermore, an overlay network can dynamically reconfigured in response to dynamic network conditions and participant "churn" (i.e., participants joining and leaving). Although deploying an overlay network requires significant infrastructure investment, these services can be purchased from a third-party commercial content distribution network provider (e.g., Akamai, Level 3, Limelight, etc.).

The architecture of any specific CSCW system will be some combination of these intrapersonal and interpersonal architectures. Hybrid architectures are common and within a system, different participants may employ different intrapersonal architectures with vastly different capabilities and constraints, especially if one or more participants are in a mobile context. The characteristic advantages of disadvantages of these architectures are summarized in Table 7.1.

Table 7.1 Summary comparison of intra and interpersonal CSCW architectures

		Strengths	Weaknesses
Intrapersonal architectures	Monolithic	+ Tight integration + Near-zero control latency + Shared memory	–Resource intensive –Potential for interference
	Coordinated	+ Tool-specific resource provision + Low-latency local communication	–Violates end-to-end principles of protocol design
	Confederated	+ Modularity + Tool independence + End-to-end protocols	–Challenging coordination and control
Interpersonal architectures	Direct	+ Minimized latency + No in-network infrastructure required	–Redundant transmission of media data –Limited access bandwidth
	Centralized	+ Common control and synchronization platform + System-wide understanding of performance	–Resource intensive –MCU placement –Single point of failure
	Overlay	+ Fault-tolerant + Dynamically reconfigurable + In-network media replication	–Requires extensive infrastructure –Expense

7.4 Synchronization Challenges

Within a specific system, there may be a number of different synchronization requirements necessary to support the various interpersonal streams of communication present (i.e., video, audio, graphics, etc.). Common synchronization regimes include:

Tight and Interactive
Video and audio conferencing is the canonical example of tight and interactive synchronization requirements. Others include floor control with respect to stream switching and viewpoint control in a shared virtual environment.

Loose and Noninteractive
When streams are largely independent with regard to the overall application-level semantics, loose and noninteractive synchronization is possible. Nonconflicting edits in a shared document are examples of information streams that enjoy loose and noninteractive synchronization.

Loose but Consistent
Loose but consistent synchronization is appropriate when the relative timing between information streams is elastic but all participants must have a consistent understanding of either the interstream timing and/or ordering. For example, event and collision detection within a shared virtual space often require a shared understanding of ordering to be maintained across all participants.

These synchronization requirements within CSCW systems give rise to a number of different challenges that must be addressed. The first of these is interpersonal bandwidth estimation. Bandwidth estimation is important in order to accurately map the application-level goals of a CSCW system to the available adaptation tradeoffs available for different streams of information. For example, in a system that included all-to-all distribution of video streams among participants, video quality may need to be managed in different ways given current estimates of upstream and downstream bandwidth. In systems with a confederated intrapersonal architecture, bandwidth estimation is complicated by interstream interference of end-to-end protocols such as TCP.

Another challenge faced is how to best express and implement complex system-wide adaptation policies and goals in a way that results in the most appropriate prioritization of tasks and streams within the system. Because data within separate streams of information within a CSCW system are often related or at least highly correlated, interstream dependencies must be managed and appropriately accounted for. This is particularly important if these dependencies and correlations are used as the basis for media encoding and compression. For example, in a multi-camera meeting capture environment, one video stream may act as a reference stream for the encoding of other nearby cameras.

Managing interstream jitter is another challenge faced by CSCW systems. Interstream jitter refers to variance in the relative timing of related media transmitted in separate streams. This is a distinct concept from network jitter which is variance in relative timing of successive media transmitted in a particular stream. Interstream jitter is a factor in media synchronization when excessive interstream jitter diminishes the utility of two or more related streams. This can be an important system issue because resources allocated to these related streams are effectively wasted.

Another challenge is recognizing when the use of network resources by a stream is having a dynamic effect on another stream within the system. In particular, adaptation strategies and protocol behavior of two or more streams may create complex interstream dynamics with unintended consequences that work against the efficient use of network resources.

7.5 Inter-protocol Coordination and Synchronization

To address these challenges, system designers can bring to bear techniques and mechanisms of inter-protocol coordination and synchronization. Here, we review a number of inter-protocol synchronization techniques in the literature that have been proposed and how they can be applied to CSCW systems. These include aggregate transport-level protocols, flow aggregation techniques, Congestion Manager, and the Coordination Protocol.

Table 7.2 Characteristics of inter-protocol coordination and synchronization techniques

	In-network infrastructure required?	Operating system support required?	Applicable CSCW architectures
Flow aggregation techniques	Yes	No	• Coordinated, confederated • Direct, overlay
Aggregate transport-level protocols	No	Yes	• Monolithic, coordinated • Direct, overlay
Congestion manager	No	Yes	• Monolithic • Direct, centralized
Coordination protocol	Yes	Yes	• Confederated • Direct

Aggregate transport-level protocols provide a unified service model for multiflow applications. Here, we review two: Structured Stream Transport and Paceline. Flow aggregation techniques, such as TCP-Trunking and Aggregate TCP multiplex related streams into a congestion controlled aggregate. Although these techniques may seem to address inter-application fairness and TCP performance issues, there are a number of significant drawbacks for their application to CSCW systems. Congestion Manager provides an operating system-level approach with flexible scheduling mechanisms for modeling rate constraints and a strongly application-driven approach for adaptation. Coordination Protocol specifically addresses the needs of systems with a confederated intrapersonal architecture with multiple independent end-to-end flows. Although this approach presupposes control of the gateway or access router in front of each participant, the mechanisms provide highly scalable aggregate congestion control while at the same time providing extremely flexible interflow control and coordination. Table 7.2 summarizes key characteristics of these various mechanisms.

7.5.1 Early Flow Coordination Protocols

Early work in interstream synchronization concentrates on managing temporal relationships between data unit in different media streams. The Multiflow Conversation Protocol (MCP) [4] is one of the earliest schemes to address the challenges of multiflow coordination. MCP relies on a token-based approach in which each flow within the system is associated with a transmission token. MCP provides the underlying token management primitives without imposing any specific token policy. In this way, specific concurrency semantics can be achieved by implementing different policies on how flow tokens are created, replicated, transferred,

and deleted. MCP also provides a mechanism for temporal synchronization across multiple independent streams called a "multiflow conversation." This mechanism allows a system to flexibly express a wide spectrum of interstream synchronization requirements. In particular, MCP multiflow conversations can be used to specify and enforce delay-constrained causality among application data units.

Escobar et al. [5] is a seminal work that explores interstream synchronization mechanisms and proposes the Flow Synchronization Protocol (FSP). FSP managed synchronization by adaptively changing playout delay at distributed clients in a coordinated manner. In this way, FSP is able to manage the tradeoff between end-to-end presentation latency, application requirements, and data loss due to late arrival. The Adaptive Synchronization Protocol (ASP) [6] generalizes this approach and relaxes the need for a synchronized clock required by FSP.

7.5.2 Aggregate Transport-Level Protocols

Structured Stream Transport (SST) proposed in [7] provides CSCW systems with a unified interface to an aggregate transport-level protocol that allows the specification and management of a hierarchically organized tree of child streams. Each of these child streams can independently transfer data and a perform flow control, allowing parallel transmission of application data units without a head of line blocking. SST supports a full spectrum of delivery semantics ranging from best-effort delivery to reliable byte streams in line with the TCP service model. CSCW systems are free to prioritize streams relative to each other and adjust priorities dynamically in order to achieve system-level goals with respect to adaptation and synchronization among flows. The SST protocol architecture is illustrated in Fig. 7.4.

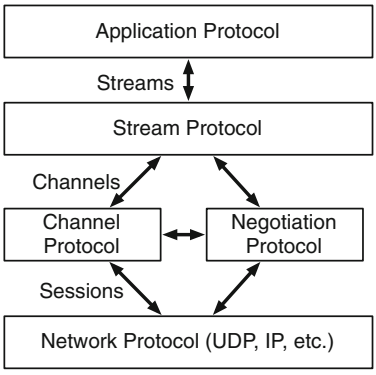


Fig. 7.4 Structure stream transport architecture

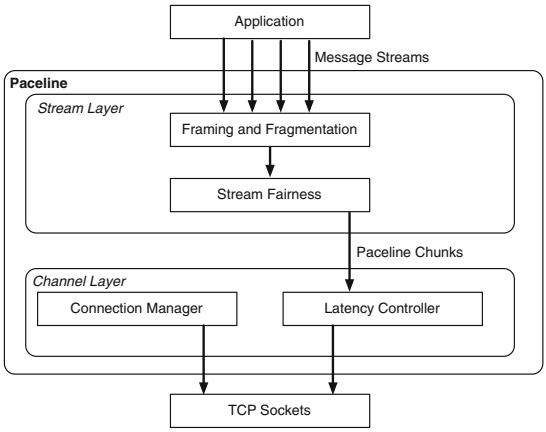


Fig. 7.5 Paceline architecture

Paceline proposed in [8] is another transport-level protocol that provides a unified interface to an aggregate set of flows appropriate for complex CSCW systems. The Paceline service model is message-based rather than stream-based. Mechanisms for managing adaptation, latency mitigation, and synchronization operate at the granularity of individual application data units. A CSCW system is able to specify message importance on a per-message basis and Paceline delivers transmitted messages in order of importance. Sending applications are able to cancel pending messages, allowing applications to efficiently schedule future transmissions given current conditions while allowing rapid and agile adaptation to changes in network conditions that render previously scheduled data useless or suboptimal. To most effectively use Paceline, CSCW systems must develop domain-specific adaptation policies that can be mapped to the priority and transmission control features provided by Paceline. These mechanisms are implemented in two layers, a stream layer and a channel layer. The Paceline architecture is illustrated in Fig. 7.5. The stream layer manages a system-wide message queue across all flows. This queue is managed to ensure that data with greater utility with respect to overall system goals are transmitted first, ensuring the lowest latency possible for messages with the highest utility. The channel layer realizes the transmission decisions made by the stream layer across a set of TCP connections configured with very small sender-side buffers. By employing standard TCP connections as an underlying transmission service, TCP-fair congestion control is achieved by definition and Paceline can be deployed at the user level without modifications to the operating system.

7.5.3 Flow Aggregation Techniques

Other approaches for protocol coordination attempt to construct flow aggregates, multiplexing the flows within a CSCW system together into a single congestion controlled flow between participants. This could be done explicitly at the application-level or transparently by a mechanism implemented within the networking infrastructure. This approach is taken by Kung and Wang [9] in their work on TCP trunking for connections that traverse a common backbone path. Kung and Wang define a TCP trunk as "an aggregate traffic stream whose data packets are transported at a rate dynamically determined by TCP's congestion control." Individual flows sending data along the same intermediary path do so using whatever transport-level protocol is appropriate for their purposes. When the packets reach the endpoint of the trunk, they are buffered until they can be sent by a single management connection to the remote end point of the trunk and then forwarded to their destination. What makes the trunk congestion responsive is the fact that the management connection employed is a TCP connection.

Another variation of this approach, known as aggregated TCP (ATCP), is presented in [10]. In this approach, end-to-end connections are divided into a local subconnection with a gateway router and a shared remote subconnection between this router and a commonly accessed remote host. In the context of a CSCW system, the remote subconnection may represent interparticipant paths or common routes along an in-network overlay. While the original intent of this work is to improve the performance of TCP connections by growing congestion windows more quickly and using persistent connections, the approach can be adapted to CSCW systems as a way of introducing aggregate flow and congestion control across streams.

These flow aggregation techniques when applied to a CSCW context have a number of limitations. First, the approach reduces aggregate traffic to a single flowshare as multiple flows utilize a single management or remote subconnection. Limiting aggregate CSCW flows to a single shared congestion responsive flow is unfairly restrictive in circumstances, where the system employs numerous flows or is competing with numerous flows at the bottleneck link. Second, the approach fails to inform CSCW application endpoints of current network performance, generally providing an opaque interface to communication resources that reveal little about underlying network conditions (available bandwidth, loss rates, etc.) which may be crucial for adaptation. Third, aggregating approaches may result in a substantial end-to-end delay as application data are buffered at the trunk node waiting to be forwarded. This is clearly disadvantageous for real-time streams with tight and interactive synchronization requirements. Finally, the end-to-end semantics of individual flows are not preserved when communication is segmented into subconnections. For example, reliability semantics dictate that an acknowledgment received by a sender indicates that a receiver has successfully received the transmitted data. In the segmentation approach, however, an acknowledgment may inform an endpoint only that the data was successfully transmitted to the next

multiplexing agent. There is no way to know whether the data was actually received by the destination endpoint.

7.5.4 Congestion Manager

The Congestion Manager (CM) architecture proposed in [11] provides a compelling solution to the problem of coordinating aggregate flows that share the same end-to-end path as often occurs in CSCW systems. Unlike the above schemes, CM emphasizes application control by informing flows of bandwidth available to them and avoiding the buffering of flow data during the forwarding process. The CM architecture consists of a sender and a receiver. At the sender, a congestion controller adjusts the aggregate transmission rate based on its estimate of network congestion, a prober sends periodic probes to the receiver, and a flow scheduler divides available bandwidth among flows and notifies applications when they are permitted to send data. At the receiver, a loss detector maintains loss statistics, a responder maintains statistics on bytes received and responds to CM probes, and a hints dispatcher sends information to the sender informing them of congestion conditions and available bandwidth. An API is presented that allows an application to request information on round-trip time and current sending rate, and to set up a callback mechanism to regulate send events according to its apportioned bandwidth. These mechanisms are implemented within the kernel of the operating system making them particularly applicable to CSCW systems with a monolithic or locally coordinated intrapersonal architecture.

It should be noted that CM addresses the problem of providing aggregate congestion control for flows that share the entire end-to-end path. That is, all flows share the same source and destination hosts. In contrast, flows in CSCW context often share a significant portion of the forwarding path, but not the entire path end-to-end. For CSCW systems with a confederated intrapersonal architecture, CM would need to be reengineered to provide a distributed implementation across multiple devices being used by the system. The main strength of the CM approach is putting the application in control allowing endpoints (or a local coordination agent) to understand the aggregate bandwidth available to the system.

CM employs a hierarchical fair-service curve (HFSC) scheduler to apportion bandwidth among flows within the network-layer mechanisms of the operating system. Consequently, a CSCW system must be able to translate its system-level requirements for flow coordination and synchronization into an appropriate HFSC configuration. Because CSCW systems can have complex schemes for adding and deleting flows, and for responding to changes in available bandwidth and changes in system state, adaptation strategies may result in very dynamic rate adjustments for individual flows. As a result, characterizing each flow's rate requirements is difficult to do a priori. The HFSC scheduler at the core of CM also serves to police the aggregate sending rate and ensures that the resulting traffic conforms to the calculated congestion controlled rate. Thus, while CM is able to take a set of

well-characterized flows and static priorities and synchronization requirements and build a hierarchical schedule for bandwidth allocation, this approach is less suitable in a more dynamic context.

7.5.5 Coordination Protocol

The Coordination Protocol (CP) architecture proposed in [12] was designed to address the needs of systems with a confederated intrapersonal architecture with multiple independent end-to-end flows. In particular, CP aims to:

- Inform endpoints of network conditions over the interpersonal data path, including aggregate bandwidth available to a CSCW system as a whole while allowing for a distributed, loosely coordinated intrapersonal architecture.
- Provide an infrastructure for exchanging state among flows and allowing the system to implement its own flow coordination scheme, and
- Avoid problems encountered by schemes that require a centralized approach to adaptation by empowering independent and loosely coupled end-to-end adaptation rather than scheduling or aggregating mechanisms.

To realize these goals, CP makes use of a shim header inserted by application endpoints into each data packet. Ideally, this header is positioned between the network-layer header and the transport-layer header. The network stacks of each host and their associated gateway router are modified to process CP packet headers, while all other nodes along the wide-area data path require no special modifications (i.e., CP is transparent to forwarding agents within the network).

CP mechanisms are largely implemented at the gateway router associated with each CSCW participant where there is a natural convergence of flow data. Without loss of generality, however, this functionality may be placed within the operating system in the case of a monolithic intrapersonal architecture or integrated within the nodes of an overlay network architecture. In general, CP is applicable for any CSCW architecture that can be modeled as a set of end-to-end flows that traverse through a common forwarding gateway on its path toward the local environment of a remote participant. This gateway is called an aggregation point (AP).

Figure 7.6 summarizes CP operation by tracing a packet traversing the path between the source and destination endpoints. The CP header is processed by the AP during packet forwarding. Essentially, the AP uses information in the CP header to maintain a per-application state table. Flows deposit information (e.g., their current priority) into the state table of their local AP as packets traverse the outbound path from an endpoint to the local AP, and then onward toward the remote participant. Packets traversing the inbound path in the reverse direction pick up entries from the AP state table (e.g., the priority of peer flows, the estimated bandwidth available) and report them to each endpoint.

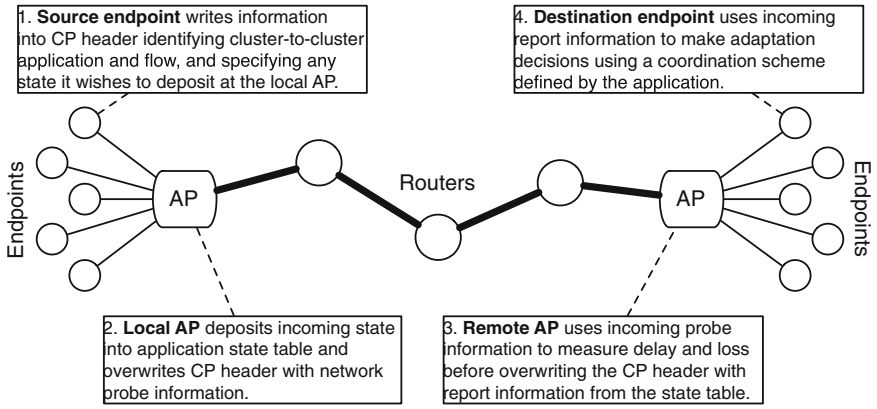


Fig. 7.6 CP operation

In addition, the two APs conspire to measure characteristics of the interpersonal data path, such as round trip time, packet loss rate, available bandwidth, etc. These measurements are made by exchanging probe information via the CP headers available from application packets traversing the data path in each direction. Measurements use all packets from all flows belonging to the same CSCW system and thus monitor network conditions in a fine-grained manner. Resulting values are inserted into the AP state table.

Report information is received by an application endpoint on a per packet basis. This information can take several forms, including information on current network conditions on the interpersonal data path (round trip time, loss, available bandwidth), information on peer flows (number of flows, aggregate bandwidth usage), and/or application-specific information exchanged among flows using a format and semantics defined by the application. An endpoint uses a subset of available information to make send rate and other adjustments (e.g., adaptation and encoding strategy) to meet application-defined goals for network resource allocation and other coordination tasks.

It is important to emphasize that CP is an open architecture. Its role is to provide information "hints" useful to application endpoints in implementing their own self-designed coordination schemes. In a sense, it is merely an information service piggybacked on packets that already traverse the interpersonal data path. As such, aggregation points do no buffering, scheduling, shaping, or policing of application flows. Instead, coordination is implemented by the application which must configure endpoints to respond to CP information with appropriate send rate and other adjustments that reflect the higher objectives of the application.

7.6 Conclusion

In conclusion, CSCW systems are complex, multiflow distributed systems that can exhibit a wide array of different intrapersonal and interpersonal architectures. Synchronization of data among flows within a CSCW system must respect both encoding and presentation dependencies inherent within a flow as well as how those flows are related to higher level task semantics. CSCW system designers may find protocol coordination techniques and mechanisms useful for realizing complex interstream synchronization, prioritization, and adaptation requirements. These mechanisms include sophisticated aggregate transport-level protocols (i.e., Structured Stream Transport and Paceline), in-network flow aggregation (i.e., TCP trunking), and kernel- or network-level interflow state sharing mechanisms (i.e., Congestion Manager and Coordination Protocol). The most appropriate mechanism to bring to bear will largely depend on the architecture of the overall CSCW system and the feasibility of translating system-level requirements onto the features provided by these mechanisms.

In practice, practical and widely used mainstream CSCW systems and tools such as Google Hangouts, Cisco WebEx Meetings, and Microsoft Office365 are designed to support browser-based clients and deployed in generic computing environments using a standard TCP/IP implementation. As such, the use of advanced protocol coordination mechanisms such as those described in this chapter is likely limited to special purpose CSCW systems, such as high-end telepresence meeting rooms, remote surgery and telemedicine, and virtual training environments.

Several standards for real-time communication and media streaming have emerged to support distributed multimedia applications such as CSCW systems. WebRTC [13] defines client-side APIs for peer-to-peer real-time communication within browsers. The MPEG media transport (MPEG-MMT) standard is a multimedia container format with advanced quality of service and quality of experience features that addresses flow multiplexing and synchronization needs of real-time multistream applications [14]. Dynamic adaptive streaming over HTTP standardized as MPEG-DASH [15] is rapidly becoming the dominant approach to video distribution.

In recognition of the practical limitations of real-world systems, more recent systems-level research in support of distributed multimedia applications such as CSCW has focused on approaches that are built on top of current standards. SmoothCache [16], for example, provides a distributed peer-to-peer adaptive streaming overlay on top of HTTP. In [17], MPEG-MMT is used in support of a collaborative virtual environment. Another recent research direction applicable to CSCW systems is the latency-sensitive management of cloud resources. For example, VMShadow [18] explores how to transparently migrate and optimize the placement of virtual desktops within a cloud computing infrastructure to minimize latency.

Definitions

Computer Supported Collaborative Work A class of distributed multimedia systems that support goal- and task-driven collaboration among multiple participants.

Overlay Network A logically connect group of nodes within the Internet that provide packet-based media and message transport and processing services.

Jitter Variation in packet delay experienced in a network typically as a result of dynamic queuing delays at forwarding nodes.

References

1. Grief, I. (ed.): Computer Supported Cooperative Work: A Book of Readings. Morgan Kaufman Publishers (1988)
2. Grudin, J., Poltrock, S.: Taxonomy and theory in computer supported cooperative work. In: Kozlowski, S.W.J. (ed.) The Oxford Handbook of Organizational Psychology, pp. 1323–1348. Oxford University Press, New York (2012)
3. Rodden, T.: A survey of CSCW systems. Interact. Comput. **3**(3), 319–353 (1991)
4. Yavatkar, R.: MCP: a protocol for coordination and temporal synchronization in multimedia collaborative applications. In: Proceedings of the 12th International Conference on Distributed Computing Systems, pp. 606–613 (1992)
5. Escobar, J., Partridge, C., Deutsch, D.: Flow synchronization protocol. IEEE/ACM Trans. Netw. **2**(2), 111–121 (1994)
6. Rothermel, K., Helbig, T.: An adaptive stream synchronization protocol. In: Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV). Lecture Notes in Computer Science, vol. 1018, pp. 178–189. Springer, Berlin (1995)
7. Ford, B.: Structured streams: a new transport abstraction. In: Proceedings of SIGCOMM'07, Kyoto, Japan, 27–31 Aug 2007
8. Erbad, A., Krasic, C.: Sender-side buffers and the case for multimedia adaptation. Commun. ACM **55**(22), 50–58 (2012)
9. Kung, H.T., Wang, S.Y.: TCP trunking: design, implementation and performance. In: Proceedings of the Seventh International Conference on Network Protocols (ICNP), Oct 31–Nov 3 1999
10. Pradhan, P., Chiueh, T., Neogi, A.: Aggregate TCP congestion control using multiple network probing. In: Proceedings of the 33rd International Conference on Distributed Computing Systems, Los Alamitos, CA (2000)
11. Balakrishnan, H., Rahul, H., Seshan, S.: An integrated congestion management architecture for internet hosts. In: Proceedings of SIGCOMM'99, Cambridge, MA (1999)
12. Ott, D., Mayer-Patel, K.: An open architecture for transport-level protocol coordination in distributed multimedia applications. ACM Trans. Multimed. Comput. Commun. Appl. **3**(3) (2007)
13. World Wide Web Consortium (W3C), WebRTC 1.0: Realt-time communication between browsers. https://www.w3.org/TR/webrtc
14. Lee, J.Y., Park, K., Lim, Y., Aoki, S., Fernando, G.: MMT: an emerging MPEG standard for multimedia delivery over the internet. IEEE MultiMed. **20**, 80–85 (2013)

15. Sodagar, I.: The MPEG-DASH standard for multimedia streaming over the internet. IEEE MultiMed. **18**(4), 62–67 (2011). http://dx.doi.org/10.1109/MMUL.2011.71
16. Roverso, R., Reale, R., El-Ansary, S., Haridi, S.: SmoothCache 2.0: CDN-quality adaptive HTTP live streaming on peer-to-peer overlays. In: Proceedings of the 6th ACM Multimedia Systems Conference (MMSys'15), pp. 61–72. ACM, New York, NY, USA (2015). http://dx.doi.org/10.1145/2713168.2713182
17. Venkatraman, K., Tian, Y., Raghuraman, S., Prabhakaran, B., Nguyen, N.: MMT+AVR: enabling collaboration in augmented virtuality/reality using ISO's MPEG media transport. In: Proceedings of the 6th ACM Multimedia Systems Conference (MMSys'15), pp. 112–119. ACM, New York, NY, USA (2015). http://dx.doi.org/10.1145/2713168.2713170
18. Guo, T., Gopalakrishnan, V., Ramakrishnan, K.K., Shenoy, P., Venkataramani, A., Lee, S.: VMShadow: optimizing the performance of latency-sensitive virtual desktops in distributed clouds. In: Proceedings of the 5th ACM Multimedia Systems Conference (MMSys'14), pp. 103–114. ACM, New York, NY, USA (2014). http://dx.doi.org/10.1145/2557642.2557646

Part III
User Experience and Evaluation Methodologies

Chapter 8
Perceiving, Interacting and Playing with Multimedia Delays

Ragnhild Eg and Kjetil Raaen

Abstract Just like interactions with the physical world, humans prefer to interact with computers without noticeable delay. However, in the digital world, the time between cause and effect can far exceed what we are accustomed to in real life. System processes, network transmission and rendering all add to the delay between an input action and an output response. While we expect and accept a computer system to process a request within a period of time, the duration of this period is dependent on the nature of the task. Highly interactive tasks, such as control systems, computer games, design software and even word processing have stringent temporal requirements, where any sustained delay can be detrimental to performance. The research interest for the human capability to operate with delay has grown over the past decades, and it has grown in at least three separate fields. In this chapter, we review relevant work from cognitive psychology, human-computer interaction and multimedia research.

Keywords Human-computer interaction · Motor-visual interaction · Delay

8.1 Introduction

Media synchronisation involves more than aligning outputs in the form of audio and video streams, or even haptic or olfactory stimulation. Media systems must also optimise for the temporal interaction between a user's input and the computer's generated output. In some scenarios, consumers are passive recipients of multimedia content; in other scenarios, consumers become users that are active in their engagements with a media system. To highlight the distinction between these scenarios, we need only look at an everyday example.

R. Eg (✉) · K. Raaen
Westerdals – Oslo School of Art, Communication and Technology, Oslo, Norway
e-mail: egrag@westerdals.no

K. Raaen
e-mail: raakje@westerdals.no

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_8

Our daily routines involve a wealth of sensory stimulation and motoric actions. Interactions with our physical surroundings are so rehearsed that we rarely reflect on them. We may see a plump, purple plum on a fruit stand and this could trigger us to reach out the arm and hand to feel the bouncy resistance of the plum's surface, before finally picking it up and leading it in a fluent motion to the mouth. If this plum had fallen off the stand and started rolling, we might also be inclined to intervene by evaluating its path and pace to predict a future location where we could place a hand to stop it. This type of interaction does not only take place in the physical world, in past decades they have become increasingly common in the digital world. With hours and hours of practice using keyboards, computer mice, joysticks, touchpads and other input devices, the computer savvy human being has become skilled at interacting with visual objects on a monitor.

Yet, the interplay between what we do and what we see differs between the physical and the digital world. First and foremost, the digital world does not adhere to physical laws. Visual objects on a standard computer screen are two-dimensional, their size are typically up- or downscaled, their movement are restricted by lines of script rather than gravity, and the temporal reaction that follows an input action is affected by system and network restrictions. Nevertheless, humans are able to overcome the unnatural setting of the digital world, partly due to the flexibility of the perceptual and motor systems. The visual system can attribute object size and lack of depth to distance, motion can be scripted to mimic physical laws, and even in nature, reactions do not instantly follow actions. Still, there is a limit to the perceptual system's flexibility.

Because the human perceptual and motor systems have evolved to interact with the physical world, it follows that interactions in a digital environment should approximate the ones we meet elsewhere. If the plump, purple plum was portrayed on a monitor, rolling along a table, the intervention would require a mouse or another input device. Still, the scene plays out the same, visual information establishes the plum's location, trajectory and speed of movement, cognitive processes predict a future location, the hand directs the intervening action there, and the continuous visual information guides the action. Despite the simplicity of the example, it serves to establish an important point. Humans grow up to navigate the physical world and the digital world is created to mimic the one we live in, any element that fails in this regard is likely to be judged as unnatural. Among the many unnatural elements in the digital experience, the temporal relationship between visual and motor events could be the most detrimental to the successful save of the tumbling plum.

Zero delay is not only unnatural, it is computationally impossible, thus humans accept delay and expect a computer system to respond within a given time. How much delay is acceptable depends on the nature of the task. If the task does not involve a direct outcome, which would be the case when downloading a software suite from the Internet, most accept minutes, even hours of delay. Granted, feedback on the progress of the operation may alleviate some impatience. At the other extreme, a player firing a gun in a fast-paced action game is interacting in an environment similar to the physical world and will expect a reaction within a fraction of a second. Clearly, temporal requirements are stringent for highly interactive tasks, but they are

also likely to depend on the nature of the task. Fortunately, the research interest in this field has grown with the technical advancement of computer and network systems. In this chapter, we draw on knowledge from three disciplines, human-computer interaction (HCI), psychology and multimedia. We use this body of work to discuss the challenge of establishing the limitations of a user's tolerance to interaction delays, with models and approaches that vary across the different schools of thought.

8.2 Models, Approaches and Terminology

Questions regarding delay in interactions have been illuminated from different domains. In many cases, the same or similar questions have been asked and answered, but information gained from each of these often do not flow smoothly to the others.

In HCI, it is popular to assume that we interact with computers in much the same way as we interact with each other. Hence, HCI assumes interactions can be modelled as two-way conversations between the user and the machine. A user asks the computer a question, and the computer answers. This means that turn-taking is involved, and users do not necessarily see themselves as the direct cause of what the computer does.

However, HCI researchers have not overlooked the importance of experienced causation, neither have cognitive psychologists. Investigations into the conscious experience of causation encompass the temporal relationship between an action and its consequence. In these areas of research, the main challenge is to establish the longest time interval that can separate a cause and an effect without disturbing the sense of agency.

Another approach comes from perceptual psychology, a branch within cognitive psychology. Here, the focus is shifted away from the longest time interval a system can operate with, focusing instead on the lowest sensitivity of the human perceptual system to delayed sensory signals. These perceptual studies build on psychophysical methodology and consider sensory thresholds that separate perceptible and imperceptible delays. Our sensory modalities are naturally separated in time due to both external and internal transmission times; light moves faster than sounds, while tactile signals must follow longer neuronal pathways than signals triggered by chemical compounds that incite taste and smell. Fortunately, our perceptual system has evolved to overcome these natural delays. Moreover, recent studies in this field have found the perceptual system capable of adapting to extended delays, for instance, when it comes to continuous interactions that engage the motor and visual systems. Because of the many nuances in perceptual studies, they do not approach the topic of a single universal threshold, nor would that be fruitful to the understanding of human perception.

In the pursuit of optimal designs for systems and applications, multimedia researchers often test human tolerance to interaction delay for specific situations. Applied studies of delayed interactions for multimedia systems have explored diverse scenarios from such a practical perspective. Most of this work has focused on

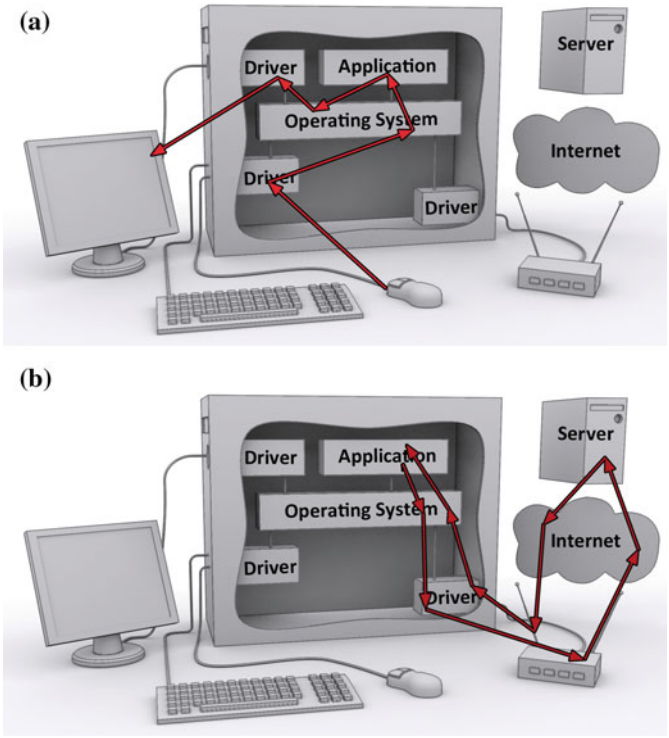


Fig. 8.1 **a** *Interface delay* denotes delay between an input device, such as a mouse, and an output device, such as a screen. **b** *Network latency* describes the round-trip-time taken to traverse a network

computer games, although virtual reality is becoming more and more relevant. Both involve systems with fast interactions between user and software. In many games, players have to do split-second decisions as well as complicated tasks of hand-eye coordination at high speed. These studies rarely adopt a model to investigate delayed interactions, focusing instead on how particular types of games and particular actions in these games are affected by the delay.

When discussing delay in games, it is common to distinguish between *network latency* and *interface delay*. Both are delay, but they appear different to the user. *Interface delay* (Fig. 8.1a[1]) is the one we predominantly refer to in this chapter, that is, delay between user input and results on screen. *Network latency* (Fig. 8.1b[2]) refers to the time it takes for an interaction to be communicated across a network. Because developers attempt to hide or mitigate network latency in their products, it affects the interaction in more subtle and varied ways. Generally, results that can be calculated locally are displayed after the interface delay, while effects that need input from the

[1] Illustration by Ivar Kjellmo.

[2] see Footnote 1

server or other participants are delayed by the network latency. Note that unless it clearly states otherwise, research on the effects of network latency generally ignores local delay.

8.3 Conversational Models in HCI

HCI draws on knowledge about human to human interactions to make inferences about human interactions with computers. In line with the underlying assumptions, the terminology in this field tends to build on the workings of human conversation. This extends to matters of timing in human-computer interactions and Seow [33, Chap. 4] puts forward a valuable summary of this terminology.

The fastest interactions are described as *instantaneous* response. This denotes a system with a process so fast that the user is unaware of any delay. For graphical controls and interfaces that mimic the physical world, Seow recommends instantaneous response times. In games too, most interactions either mimic the physical world or are graphical in nature, indicating that responses should be instantaneous. Where Seow proposes that instantaneous responses go unnoticed and generally stay below 0.1–0.2 s, these values have not been supported by consistent empirical findings. On the contrary, Sect. 8.4 outlines experimental studies whose findings suggest that delay is imperceptible only at very short time intervals.

Immediate is Seow's next responsiveness class. This describes responses that feel natural in human interaction, such as conversations. When one participant in a conversation ends a statement, the other will respond immediately, or between 0.5–1 s. Any additional delay will give the impression that something unexpected is happening in the conversation. Maybe the partner needs to think about an answer. This level of responsiveness is appropriate in human-computer interactions when the workflow mimics a conversation; if a user requests information assumed to be loaded and ready, which would be the case when scrolling down a web page, any delay will heighten the anticipation of the waiting user. After *immediate* follows the *continuous* responsiveness class [33]. In human to human interaction, we sometimes need to stop to think, or emphasise something. These pauses can be between 2–5 s without breaking the flow of the conversation. Pauses of this magnitude are accepted if we expect our conversation partner to have to think about an answer. Thus, when interacting with computers, we accept such delays for queries that we think of as complex. To maintain continuity and not break the flow of the interaction, we expect a reply within these limits. If results are not available after this time, it is critical to provide feedback that progress is happening, and that results will be ready soon.

In HCI, interactions that involve a single input and an expected outcome typically fall in the *immediate* or *continuous* class, this could be an interaction as simple as deleting an email and wait for the confirmation. With this starting point, Weber et al. [41] set up an experiment in which participants were tasked with monitoring an assumed manager's email inbox. For every incoming email, participants had to assess its relevance and either forward or delete it; each option involved moving the

mouse cursor to select the email and the appropriate action, then wait for the pop-up window to select the recipient or confirm deletion. While waiting, participants were presented with an activity animation that was displayed for the duration of the delay introduced between the input action and the appearance of the window. Ranging from 300–3000 ms, the delay was either constant or variable. Weber et al. [41] found that participants responded faster with low variability, even when this entailed longer delays. Yet, failure rates did not differ depending on the variability of the waiting time. The authors conclude that user experience for this type of system depends on the constancy of the response, not the duration of the delay. Presumably, the isolation of the task carries a heavy load when it comes to task motivation and execution time, but demands less when it comes to performance accuracy.

Finally, Seow uses the term *captive* for delays that range between 7–10 s. This class describes the longest time interval for a system to yield an output, without losing user engagement. Beyond this delay, a user will give up or assume something is wrong with the system, much like they would assume a human conversation partner had lost interest. Following such an extensive delay, users require useful results. If results cannot be yielded, the system should instead provide users with the option to think about something else and return to the slow task at a later stage. At this point, many users will take a break and go away for coffee. Avoiding this type of waiting game is perhaps more common sense than empirically founded research, further discussion of captive delays are for the time being deferred to the next coffee break.

8.4 Perceptual Models

In contrast to Seow's broad range of categories and time intervals, perceptual and psychophysical research centers around the processing of sensory inputs, narrowing the scope to sensory thresholds that stay well below a second. In this area of research, the term visuomotor is used in reference to motor activities that synchronise with, and depend on, visual information. These activities typically involve complex actions, such as driving a car, we therefore, refrain from applying this term to the limited degrees of freedom found in many human-computer interactions. Instead, focus is directed at the direct causal relationship between motoric inputs and visual outputs, that is, input actions followed by visual events on screen. We deem their temporal relationship to be most accurately labelled as motor-visual delays. Accordingly, the word order determines the initiating and the consequential event in the interaction and delay refers to the time that passes between, disregarding the cause of the delay.

Research on the processing of different sensory signals shows that humans are incapable of perceiving very short delays between stimuli from different modalities [14], a perceptual mechanism referred to as temporal integration [36]. In simplified terms, humans have a perceptual buffer that uses redundant information to establish which signals belong together in order to align and integrate them. The information used for integration can be spatial, temporal or semantic, it can also relate to the magnitude or shape [42]; combined they strengthen the bond between the

modalities and establish perceptual integration. In turn, the perceptual bond becomes more resilient, even capable of compensating for some discrepancy between modalities, such as temporal delay. Thus, our sensory systems process many external stimuli in parallel, but somewhere along the line they converge and the perceptual system aligns them to create a unified experience. The perceptual system is active in preserving this unity, but it can only compensate for a certain amount of discrepancy. This limit is remarkably difficult to establish from the literature that in itself is discrepant.

With respect to temporal discrepancy between something felt and something seen, the subjective sensitivity to delays could be well below 50 ms [26]. Indeed, in a study on the temporal integration of visual and tactile signals, Spence et al. [37] explored the ability to distinguish the temporal order between a light flashing and a vibration felt by the finger [37]. At the most sensitive, they established that participants could discern signals presented to the different modalities when they were separated by more than 28 ms. Another study on the temporal sensitivity to visuotactile asynchrony found that temporal integration ceases with separations longer than ≈80 ms [14]. However, this lower bound applied only to isolated presentations, repeated pulse presentations yielded far higher values. Higher values are also presented in a study by Noel et al. [25], for visual stimuli paired with vibration stimuli. The results established a midpoint for simultaneity at 73 ms, and a 136 ms for temporal integration. Although inconsistent, these results highlight two things: absolute synchrony is not required to achieve temporal integration between the two sensory modalities, and when it comes to the visual and tactile inputs, imperceptible delays range in the tens-of-milliseconds, not in the hundreds.

Human-computer interactions rely not only on sight and touch, they also depend on clicks, keystrokes and other movements. Consequently, these interactions go beyond the convergence of visual and tactile sensations, instead they emphasise the intended movement rather than the associated touch. In other words, the tactile sense is supplemented by kinaesthetic information, adding the complexity of intent and action. This complexity is highlighted by findings that motor–visual interactions are more adaptive to timing judgements than are visuotactile interactions [38]. This type of temporal adaptation is attributed to a recalibration of the perceptual system, meaning that continued exposure to two asynchronous signals can lead to greater tolerance to subsequently delayed presentations. Research on adaptation to artificially introduced delay is studied in order to understand just how flexible the perceptual system is in its ability to both compensate and realign to sensory discrepancy. For instance, Fujisaki and Nishida [14] and Spence et al. [37] demonstrate that humans can detect very short temporal offsets between sensory signals, yet their results also indicate that we are capable of adapting to fairly long temporal delays.

Rohde and Ernst [31] have demonstrated that recalibration can take place when the movement precedes the visual stimulus, as well as when the movement follows. In both situations, the point of subjective simultaneity, the point in time where two events are deemed to co-occur, is shifted away from the point of actual simultaneity. Combined with similar results presented by Stetson et al. [38] and Sugano et al. [40], it is evident that the challenge with motor-visual delays in multimedia systems encompasses more than temporal thresholds. Considering that the human perceptual

system can realign motoric and sensory signals with continued exposure to consistent delay, it is plausible that this mechanism is activated frequently during our daily interactions with multimedia systems.

Clearly, simultaneous is not synonymous with zero delay, a computational impossibility. On the one hand, the human sensitivity to sensory delays places strong demands for speedy system responses, on the other hand, the human perceptual system is adaptable and quite capable of compensating for short temporal offsets between corresponding signals. However, even though short delays are frequently imperceptible, they can still affect user performance, for instance through increased stress levels [2]. Thus, perceptual integration and temporal thresholds may not provide the sole explanation model when we seek to understand how tolerant users are to interaction delay.

8.5 Sense of Agency in HCI and Cognitive Psychology

In fast-paced human-computer interactions, a button push and a visual event have to coincide in order to ensure fluent operations. Fortunately, as noted, they need not be in perfect synchrony. Humans learn from experience what to expect following a familiar action, but timing is of the essence. Causality is an important factor in motor–visual interaction, if too much time passes after an action, the delayed consequence may be attributed to another event [16]. Humans are very adept at handling and acting on objects, facilitated by both the motoric and the visual systems, along with other inputs. Sense of agency is adopted and studied by researchers in both HCI [20] and cognitive psychology [22]. It refers to the experience of being the direct cause of an event, an experience that can be altered by timing, among other factors [15]. Indeed, one study found that participants maintained the sense of agency from a joystick controlling the movements of an image for intervals as long as 700 ms [11]. At the same time, these participants were clearly aware of delays much lower than this, though the authors do not draw any conclusions about how short delays need to be before they cease to be noticeable.

Noting the difference between sense of agency and sense of simultaneity, Rohde et al. [32] compared the respective temporal boundaries for delays between a button-push and a visual flash. Participants either made judgements on the simultaneity of the events, or on the event serving as the agent. Ingeniously, the virtual button was implemented using a haptic feedback device that allowed tracking of the participants' movements well before they pushed the button. Because movements leading up to a button-push are predictable, the researchers could trigger the flash before users believed they had pressed the button. On average, the button-push was perceived as the agent as long as the visual flash was not delayed by more than ≈400 ms; conversely, the two events were judged as simultaneous, at greater than chance rates, when the flash delay stayed below ≈250 ms. Hence, the experience of control persists longer than the experience of events being in synchrony.

Using shorter delays, Metcalfe et al. [21] investigated the sense of agency for three age groups in an experimental task that required participants to control a cursor using a mouse in order to avoid some objects on screen and to touch others. Task performance, participants' self-judgements of performance and self-judgements of control were compared across a control condition and two delay conditions, with the cursor delayed by either 250 or 500 ms. Unsurprisingly, all age groups showed best performance for the control condition, and also better performance for the 250 ms compared to the 500 ms delay condition. To compare the sense of agency across conditions, Metcalfe et al. [21] subtracted subjective judgements of control from subjective judgements of performance, then they looked at the difference between the baseline (the control condition) and the experimental conditions. The resulting negative scores showed that participants experienced significantly lower sense of agency both for the 250 ms and the 500 ms delay conditions. Only the childrens' agency scores showed a somewhat lower sense of agency for the 500 ms compared to the 250 ms delay condition, the older adults showed similar scores for both conditions, as well as the greatest sense of agency of the three age groups. On the other hand, the college students experienced the least sense of agency, showing high awareness that they were not fully in control of the task operations. The differences between age groups are likely related to several factors, but first and foremost experience and perceptual learning. Of the three groups, college students are likely to spend the most time operating a computer with a mouse. Hence, they are also the group most affected by the interaction delay. Common across the age groups is the lowered sense of agency that follows interaction delays. Findings pertaining to sense of agency bring forward detrimental consequences that have an impact on more than mere perceptibility and motor performance; interaction delays affect the experience of being the initiator of events, dampening the feeling of being in control. The study by Metcalfe et al. [21] suggests that sense of agency is affected even by delays smaller than 250 ms.

It seems then that the sense of agency endures across a greater temporal window than does perceptual integration. This assumption is supported by indications that active touches are registered sooner than passive touches, which relates back to a so-called efference copy, a plan that precedes a movement and that is transmitted to the brain as a copy before the movement is initiated [43]. The idea of the brain receiving signals about intended movement is further supported by an experiment that recorded cerebral activity around 350–400 ms before participants could ascertain their intention to engage in a spontaneous, voluntary action [19]. Accordingly, the temporal resilience associated with sense of agency is connected to our actions being perceived as the cause of an event, leading to the experience that the event is taking place earlier than it, in fact, does [10]. In this regard, the predictability of an event plays an important role in reducing the delay between cause and effect [23].

Keeping in mind that sense of agency involves the anticipation and prediction of an initiated event, it follows that we will attribute a delayed effect to our action simply because we know it is coming. In fact, the perceptual system aids us in this process. Nevertheless, sense of agency may be more applicable to isolated interactions where each action and its consequence is distinguishable from the next. In continuous interactions with many and quick motor inputs, the sense of agency becomes

difficult to preserve, not due delay, but due to the challenge of attributing events to single inputs. Hence, in computer games and other fast-paced computer interactions, other approaches may yield better representations of users' tolerance to delay.

8.6 Applied Studies on Delay

Despite the collective knowledge from the fields of perceptual and cognitive psychology, as well as HCI, the literature does not provide clear-cut thresholds for tolerable interaction delays.

Most applications rely on more than one type of input and output, adding to the complexity of an already complex interaction. In their work, researchers and developers alike require thresholds for acceptable interaction delays in multimedia, giving rise to a body of applied studies on interactive applications.

In line with some of the experimental tasks implemented in the discussed psychological research [11, 21], interactive games serve as relevant examples of time-dependent applications due to their fast interactions and indirect action inputs. Unlike psychological experiments, games are far more complicated and involve numerous disturbing elements. This makes games an interesting case for exploring delay and interactions. Two main approaches have been employed to study delay in games: Controlled laboratory experiments and studies observing organic gameplay.

8.6.1 Controlled Studies on Games

Allowing participants to play games in laboratory settings with controlled delay is a natural approach to investigating the effects of the different types of delay. It gives the researcher control over all parameters when the game is played. On the other hand, gathering datasets from a significant number of players is expensive in terms of time and effort. Most work on games has focused on network latency. There seems to be an implicit assumption in play that interface delay is insignificant, because this is rarely recorded or even mentioned in these studies.

Because games are extremely varied in content, context, pace, inputs and outputs, we need to understand the different game categories in order to understand the impact of latency on game interaction. Claypool and Claypool [5] present a much used categorisation of games based on player interaction. Each category comes with its own recommended range of tolerable network latency, grounded in experimental studies carried out on games representative of the respective categories.

The most time-dependent of these categories is the *first person avatar* games. In these games, the player controls an avatar directly, and the output shows the world from the point of view of the avatar. A subcategory of first person avatar games is *first person shooter*, in these games players control a character that moves around freely in the world searching for, and preferably, shooting opponents. Studying one

such game, *Unreal Tournament 2003*, Beigbeder et al. [1] gave players a set of game tasks, one was to move in a specified pattern and to shoot at moving targets. They ran the experiment three times at different latency levels, but included only two participants. Despite the limited data, the study concludes that the positive experience of playing the game is reduced with network latencies above 100 ms. Running the same game, Quax et al. [28] set up a 12-player match in a controlled environment. Each player was assigned a pre-determined latency value that was in operation for the duration of the match. After the match, players answered a questionnaire about their experience playing the game. This study also included relatively few players, but they still concluded that 60 ms of latency can noticeably reduce both performance and experience of playing this particular game.

Another subcategory of the first person avatar game involves *racing games*, where players control a vehicle obeying more or less realistic physics. Pantel and Wolf [27] stipulate that racing games are the most sensitive class of game. In their study, they found that the *average* player's performance deteriorates already at their lowest tested network latency of 50 ms. While the beginners drove too slowly to notice this, the skilled players were able to compensate for more latency. Performances of the excellent drivers degraded sharply at 150 ms.

Third person avatar are games where the player controls a specific avatar seen on screen. Because the control of the avatar is less direct than for *first person avatar* games, players tend to be less sensitive to any type of delay. Fritsch et al. [13] evaluated the performance of two players engaged in the game *Everquest 2*, under different conditions. They conclude that approximately 500 ms is a reasonable limit for network latency in this game.

Lastly, *omnipresent* games are games where the player has no clear avatar, but controls the game from an outside perspective. This gives indirect control of the action. These games have been explored by game researchers [4, 34], but a latency threshold is difficult to establish. For the time being, the consensus has landed on 1000 ms as an acceptable upper bound for network latency.

Team sport games are played in a different manner than most games. Usually players have control of one character at a time, much like a third person avatar game. Unlike an avatar game, players frequently switch characters, depending on who is most involved in the action. One study has looked into the consequence of interaction delay in one such game, *Madden NFL Football*, and concludes that network latencies as high as 500 ms are not noticeable [24].

Although less controlled than a psychophysical experiment, these studies implement control over the game situation. With this approach, it is evident from the presented results that game performance decreases as network latency increases above a threshold. However, it is not clear what this threshold is, nor is it clear how to group games with similar characteristics. In general, few studies have found effects of latencies shorter than 100 ms, and for some games, thresholds are considerably higher. It is important to note that all these studies have worked with network latency rather than interaction delay. How the two forms of delay interact is still an open question.

8.6.2 Observational Studies on Games in Action

To avoid the limitations of controlled studies, others have taken different approaches to determining acceptable network latency for games. To get around the primary limitation of controlled experiments, the resources and time required to gather data, some researchers choose to gather data from games as they are being played across the net. This gives access to large amounts of data, although researchers must compensate for random variation across conditions.

An interesting approach is that of Chen et al. [3]. They examine an online role playing game, *ShenZhou Online*. Following Claypool's classification, this game falls within the third person avatar game. Instead of using a controlled lab environment, the authors chose to analyse network traces from an existing and running game. In their investigation of how quality of service (QoS) could influence duration of play session, they examined several network transmission factors, including packet loss, latency and jitter. Their presented results show a linear correlation between increased latency and decreased game session length at 45–75 ms delay. For standard deviation of latency, which is used a measure of jitter, the uncovered correlation is even stronger. This finding contrasts directly with Quax et al. [28], who found no effects of jitter. Most likely, these different results represent the different characteristics of the respective games. The study by Chen et al. [3] suggests that the negative impact of network latency occurs sooner than the 100 ms threshold mentioned in the earlier literature. Using session duration as a measure of game experience is a fruitful approach, but the chain of events from network latency to session duration involves many twists and turns. There is spacious room for unforeseen and undetected variables.

Dick et al. [9] use two separate methods to investigate how network latency affects players. Following the International Telecommunication Union's recommended *impairment scale* [17] with a *Mean Opinion Score* (MOS) rating, they ran an online survey that assessed the level of network latency that corresponded to different degrees of impairment. Players reported that they could play *unimpaired* with up to about 80 ms of network latency, and 120 ms latency was *tolerable* for most games. Further, the authors ran a controlled experiment testing the negative impact on game play for different latencies. They included different games in their study, revealing large differences between games. The most time-dependent game turned out to be *Need for Speed Underground 2*, showing impairment to game play even at the lowest tested latency of 50 ms.

In general, these observational studies present lower limits than the controlled studies, suggesting that latencies as low as 45 ms can have an impact on game play.

8.6.3 Interactions Using Other Interfaces

The outlined studies on both controlled and observational game experiments cover interactions where users rely on a pointing device, typically a mouse, while the out-

put is directed to an observed screen. This type of interaction is in wide use, but other means of interactions are also common. Conversely, research on the temporal interaction between human and computer using less linear input devices are scarce. Yet, some exceptions exist and we include a selection to highlight the need for further explorations in these areas.

Research on input–output delay, very similar to interface delay, was carried out already in 1963, although this early work with an electromechanical tool did not consider delays shorter than 1 s [35]. Conversely, Jota et al. [18] examined users' sensitivity to delay in touch interfaces, making use of fast-response touch-sensitive screens (down to 1 ms). Participants were asked to drag an object to a target using a touch gesture, the task completion time was used as a measure of user performance. The authors found a significant decrease in task performance for delays down to ≈25 ms and an overall sensitivity to delays for actions that require pointing and dragging. This type of interaction introduces some caveats though. If the user drags an object at constant velocity across the screen, temporal delay appears to the user as spatial offset from the finger to the object; it is possible that the measured effect is due to this spatial offset rather than the temporal delay.

Taking a step away from both desktop and laptop computers, one advancement that has been worked on intermittently for many years is to equip users with a screen attached to the head. Combined with a system to track head movements, this could allow for a more immersive virtual world. These virtual worlds can either overlap with or replace the physical world. If a user sees only computer-generated content, the system is called *Virtual Reality* (VR); on the other hand, if a user sees virtual content superimposed on the real world, the system is called *Augmented Reality* (AR). With a stereoscopic view that follows head movements users are provided with an unprecedented visual immersion in the computer-generated world. AR and VR share the same potential problems when it comes to motion sickness and discomfort while using the systems. Davis et al. [8] have investigated a condition they term *cybersickness*, as an analogy to motion sickness. They consider and propose several causes to these problems, delay among them. However, they do not specify the amount of delay that might lead to symptoms.

These examples show the diversity of the numerous applied studies on interaction delay. Some interactions have received far less attention than others, and the time is ripe to harvest these fruits and share their secret flavours with the community.

8.7 Uniting HCI, Psychology and Multimedia

Over time, researchers have opened their eyes to adjoining fields and the work on temporal interactions has slowly begun converging into multidisciplinary studies. These works are still few and far between, but their results highlight the advantage of bringing together methods and knowledge from different disciplines.

In one such study, research methods from cognitive psychology are put into practice in a computer game with an obstacle-avoidance task. Participants played the

game with a 235 ms delay between the mouse movements and the corresponding movements on screen [7]. Participants performed significantly better in the absence of added delay than they did with the inserted delay. However, with continued exposure to the motor-visual delay the perceptual system had time to adapt and recalibrate; with this temporal adaptation, participants improved their performance. Notably, even with adaptation, the task became increasingly difficult as the visual elements sped up. Following the main experiment in the obstacle-avoidance study [7], participants attempted the game without the imposed delay. This lead to an interesting illusory effect, with reports of events occurring before the mouse input that triggered them. The illusion serves as a lively example of the perceptual system's flexibility.

To bring their investigation of game delay closer to psychological research practice, Stuckel and Gutwin [39] used a simple cooperative game. In their study, they define the concept *tightly coupled interaction* as 'shared work in which each person's actions immediately and continuously influences the actions of others'. Such interactions are a cornerstone of games, whether cooperative or adversarial. This study compares simple interface delay with delay compensated by the system, which is comparable to how network latency appears in many games. Evaluating user performance under different delay situations, they conclude that interface delay is actually preferable to the simulated network latency, seeing how it allows users to compensate.

In our own work, we started with the assumption that the study of delay in human-computer interactions should commence with the most rudimentary form of interaction and progress iteratively with more complex interactions. Similar to Cunningham et al. [7], we introduced psychological methodology to more applied contexts. We started with two simple button-push experiments. In the first, participants pushed a button to switch a black disc on or off, in other words, the act of pushing a button resulted in the appearance or disappearance of a black disc on screen [30]. The second experiment added one level of complexity, replacing the static display with a moving one. Here, the button push caused the rotating disc to change direction [29]. In both experiments, we set out to explore how long the delay between button-push and disc-action could be before users could perceive them. While the majority of participants could perceive delays around 170 ms, some came close to the experiment's lower bound of 50 ms.

Further increasing the level of complexity, we shifted focus from perceptibility to user performance. We designed two games [12], differing in task and input device, to evaluate the effects of delay on user performance. One is a target pursuit game, where the goal is to use the mouse to move the cursor and target a bouncing ball, all the while compensating for the delay between the moving mouse and the moving cursor. Following up on this game in a controlled experiment, we found that as the game-pace increased, the delayed interaction became more challenging. This was reflected in poorer performance scores from around 100 ms delay [6].

A common observation across all our experiments is that temporal sensitivity is highly individual and some are able to perceive even the shortest delay we could achieve with the system at hand. Consequently, when specifying limits to delays in

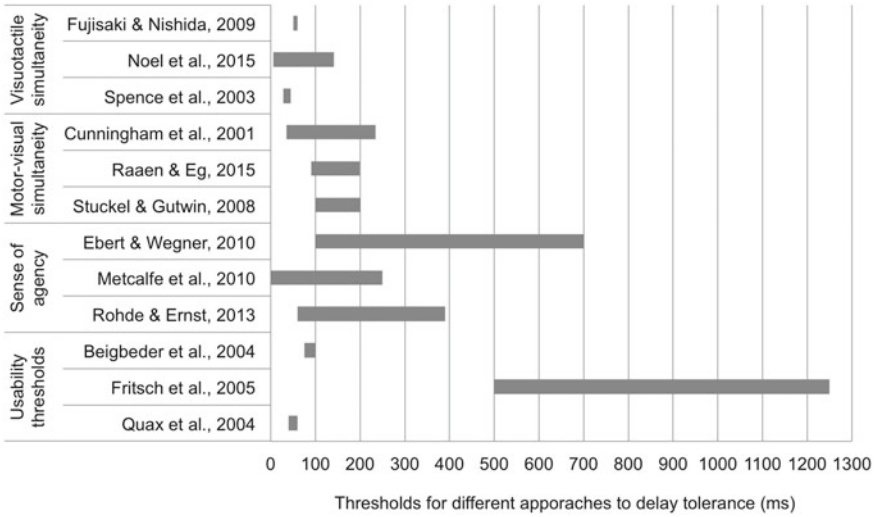


Fig. 8.2 Temporal thresholds for different experimental and applied approaches to establish tolerance to delay. Note that Cunningham et al. [7] and Metcalfe et al. [21] did not test values between their minimum delay and their reported threshold

interactions, it is important to remember that accommodating the average user might not satisfy all users. This point is further emphasised with the overview of the large variation in delay thresholds in Fig. 8.2.

8.8 Conclusion

Measuring a user's tolerance to motor-visual delays is not a straightforward endeavour. HCI and multimedia research, along with perceptual and cognitive studies, employ a diverse set of methodologies in their investigations of human sensitivity to temporal offsets. Each comes not only with a unique angle and individual strengths, but also limitations in the generalisability of results. Moreover, the various thresholds, upper bounds and sensitivity measures that spawn from the different fields serve to highlight the many different mechanisms at work in what is a far stretch from being a simple interaction between human and computer.

Applied research on interaction delay, particularly game studies, shows that performance and quality of experience deteriorates anywhere between 50–500 ms depending on the properties of the interaction. Faster, more precise interactions generally require faster response. How different factors and game characteristics contribute to increased or decreased tolerance to interaction delay is still unclear.

Consistent with the diverging results yielded by applied studies on interaction delay, HCI and psychological research introduce delay values that start at the

tens-of-milliseconds and end at the hundreds. However, distinctions should be made regarding the nature of the various interaction tasks. When working with isolated inputs and outputs, a user can tolerate a couple of seconds of delay, as long as it remains constant. In contrast, performance on fast-paced and continuous interactions is far more vulnerable to delays. Sense of agency deteriorates and is lost somewhere along the range of 250–700 ms, again depending on the task. The tactile and visual modalities are far more tightly coupled, humans are actually able to detect 50 ms of separation between the two. Yet, the human perceptual system is adaptable and can recalibrate the synchrony between tactile and visual, as well as motor and visual, signals. The latter indicates that continuous exposure to temporally offset visual presentations can dampen the conscious experience of delay, which again could explain the preference for constant over variable delay.

As remarked, human sensitivity to delays varies greatly and a universal boundary to how much delay a user will tolerate is unlikely. This limit depends on the system, the nature of the interaction, the pace, and the experimental approach. Bearing in mind that the different fields, with their different traditions and methods, are approaching the same question from different angles, we conclude that we all benefit from seeking out the knowledge brought forward by others. Currently, little interaction can be found across the boundaries of the respective fields, despite the endeavours to understand the same phenomenon: just how tolerant humans are to a delay that follows an action. Considering the amount of time the working human spends interacting with a computer system, not to mention the never-ceasing popularity of games and the advent of virtual reality, this topic is nothing but timely. Deepening the understanding of human temporal perception extends beyond scientific curiosity, it is crucial to defining system and application requirements so that future human-computer interactions are optimised for the human experience.

References

1. Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., Claypool, M.: The effects of loss and latency on user performance in unreal tournament 2003. In: Proceedings of 3rd Workshop on Network and system support for games (NetGames), pp. 144–151 (2004)
2. Chen, K.T., Lei, C.L.: Are all games equally cloud-gaming-friendly? an electromyographic approach. In: 11th Annual Workshop on Network and Systems Support for Games (NetGames), pp. 1–6 (2012)
3. Chen, K.T, Huang, P., Wang, G.S., Huang, C.Y., Lei, C.L.: On the sensitivity of online game playing time to network Qos. Proceedings of IEEE INFOCOM (2006)
4. Claypool, M.: The effect of latency on user performance in real-time strategy games. Comput. Netw. **49**(1), 52–70 (2005)
5. Claypool, M., Claypool, K.: Latency and player interaction in online games. Commun. ACM **49**(11), 40–45 (2006)
6. Claypool, M., Eg, R., Raaen, K.: Modeling user performance for moving target selection with a delayed mouse. In: Amsaleg, L.,Gudmundsson, G., Gurrin, C., Jonsson, B., Satoh, S. (eds.). Lecture Notes in Computer Science, pp. 226–237. Springer (2017)
7. Cunningham, D.W., Billock, V.A., Tsou, B.H.: Sensorimotor adaptation to violations of temporal contiguity. Psychol. Sci. **12**(6), 532–535 (2001)

8. Davis, S., Nesbitt, K., Nalivaiko, E.: A systematic review of cybersickness. In: Proceedings of the Conference on Interactive Entertainment (2014)
9. Dick, M., Wellnitz, O., Wolf, L.: Analysis of factors affecting players' performance and perception in multiplayer games. In: Proceedings of 4th Workshop on Network and System Support for Games (NetGames), pp. 1–7 (2005)
10. Eagleman, D.M., Holcombe, A.O.: Causation and the perception of time. Trends Cogn. Sci. **6**(8), 323–325 (2002)
11. Ebert, J.P., Wegner, D.M.: Time warp: authorship shapes the perceived timing of actions and events. Conscious. Cogn. **19**(1), 481–489 (2010)
12. Eg, R., Raaen, K.: How to demonstrate delay? let's play! In: Proceedings of the 7th International Conference on Multimedia Systems, MMSys '16, pp. 1–28 (2016)
13. Fritsch, T., Ritter, H., Schiller, J.: The effect of latency and network limitations on MMORPGs: a field study of everquest2. In: Proceedings of 4th workshop on Network and system support for games (NetGames) (2005)
14. Fujisaki, W., Nishida, S.: Audio-tactile superiority over visuo-tactile and audio-visual combinations in the temporal resolution of synchrony perception. Exp. Brain Res. **198**(2–3), 245–259 (2009)
15. Haggard, P., Chambon, V.: Sense of agency. Curr. Biol. **22**(10), R390–R392 (2012)
16. Heron, J., Hanson, J.V.M., Whitaker, D.: Effect before cause: supramodal recalibration of sensorimotor timing. PLoS One **4**(11), e7681 (2009)
17. International Telecommunication Union (ITU-T) (2014) Methods for the subjective assessment of small impairments in audio systems bs series. Recommendation BS1116-2 2
18. Jota, R., Ng, A., Dietz, P., Wigdor, D.: How fast is fast enough? a study of the effects of latency in direct-touch pointing tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2291–2300 (2013)
19. Libet, B.: Unconscious cerebral initiative and the role of conscious will in voluntary action. Behav. Brain Sci. **8**(04), 529–539 (1985)
20. Limerick, H., Coyle, D., Moore, J.W.: The experience of agency in human-computer interactions: a review. Front. Hum. Neurosci. **8**(3), 643 (2014)
21. Metcalfe, J., Eich, T.S., Castel, A.D.: Metacognition of agency across the lifespan. Cognition **116**(2), 267–282 (2010)
22. Moore, J.W., Obhi, S.S.: Intentional binding and the sense agency: a review. Conscious. Cogn. **21**, 546–561 (2012)
23. Moore, J.W., Lagnado, D., Deal, D.C., Haggard, P.: Feelings of control: contingency determines experience of action. Cognition **110**(2), 279–283 (2009)
24. Nichols, J., Claypool, M.: The effects of latency on online madden NFL football. In: Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), pp 146–151 (2004)
25. Noel, J., Wallace, M., Orchard-Mills, E., Alais, D., van der Burg, E.: True and perceived synchrony are preferentially associated with particular sensory pairings. Sci. Rep. **5** (2015)
26. Occelli, V., Spence, C., Zampini, M.: Audiotactile interactions in temporal perception. Psychon. Bull. Rev. **18**(3), 429–454 (2011)
27. Pantel, L., Wolf, L.C.: On the impact of delay on real-time multiplayer games. In: Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '02, pp. 23–29 (2002)
28. Quax, P., Monsieurs, P., Lamotte, W., De Vleeschauwer, D., Degrande, N.: Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In: Proceedings of 3rd Workshop on Network and System Support For Games (NetGames), pp. 152–156 (2004)
29. Raaen, K., Eg, R.: Instantaneous human-computer interactions : button causes and screen effects. Int. Conf. Hum. Comput. Interact. **2015**, 1–12 (2015)
30. Raaen, K., Eg, R., Griwodz, C.: Can gamers detect cloud delay? In: Proceedings of the 13th Annual Workshop on Network and Systems Support for Games (NetGames), vol. 200 (2014)

31. Rohde, M., Ernst, M.O.: To lead and to lag-forward and backward recalibration of perceived visuo-motor simultaneity. Front. Psychol. **3**, 1–8 (2013)
32. Rohde, M., Scheller, M., Ernst, M.O.: Effects can precede their cause in the sense of agency. Neuropsychologia **65**, 191–196 (2014)
33. Seow, S.C.: Designing and Engineering Time. Addison-Wesley (2008)
34. Sheldon, N., Girard, E., Borg, S., Claypool, M., Agu, E.: The effect of latency on user performance in warcraft iii. In: Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames), pp. 3–14 (2003)
35. Sheridan T, Ferrell W (1963) Remote manipulative control with transmission delay. IEEE Trans. Hum. Factors Electro. HFE **4**(1), 25–29
36. Spence, C.: Crossmodal correspondences: a tutorial review. Atten. Percept. Psychophys. **73**(4), 971–995 (2011)
37. Spence, C., Baddeley, R., Zampini, M., James, R., Shore, D.I.: Multisensory temporal order judgments: when two locations are better than one. Percept. Psychophys. **65**(2), 318–328 (2003)
38. Stetson, C., Cui, X., Montague, P.R., Eagleman, D.M.: Motor-sensory recalibration leads to an illusory reversal of action and sensation. Neuron **51**(5), 651–659 (2006)
39. Stuckel, D., Gutwin, C.: The effects of local lag on tightly-coupled interaction in distributed groupware. In: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work (CSCW), pp. 447–456 (2008)
40. Sugano, Y., Keetels, M., Vroomen, J.: Adaptation to motor-visual and motor-auditory temporal lags transfer across modalities. Exp. Brain Res. **201**(3), 393–399 (2010)
41. Weber, F., Haering, C., Thomaschke, R.: Improving the human computer dialogue with increased temporal predictability. Hum. Factors J. Hum. Factors Ergonomics **55**, 881–892 (2013)
42. Welch, R.B., Warren, D.H.: Immediate perceptual response to intersensory discrepancy. Psychol. Bull. **88**(3), 638–667 (1980)
43. Winter, R., Harrar, V., Gozdzik, M., Harris, L.R.: The relative timing of active and passive touch. Brain Res. **1242**, 54–58 (2008)

Chapter 9
Methods for Human-Centered Evaluation of MediaSync in Real-Time Communication

Gunilla Berndtsson, Marwin Schmitt, Peter Hughes, Janto Skowronek, Katrin Schoenenberg and Alexander Raake

Abstract In an ideal world people interacting using real-time multimedia links experience perfectly synchronized media, and there is no latency of transmission: the interlocutors would hear and see each other with no delay. Methods to achieve the former are discussed in other chapters in this book, but for a variety of practical and physical reasons, delay-free communication will never be possible. In some cases, the delay will be very obvious since it will be possible to observe the reaction time of the listeners modified by the delay, or there may be some acoustic echo from the listeners' audio equipment. However, in the absence of echo, the users themselves do not always explicitly notice the presence of delay, even for quite large values. Typically, they notice something is wrong (for example "we kept interrupting each other!"), but are unable to define what it is. Some useful insights into the impact of delay on a conversation can be obtained from the linguistic discipline of Conversation Analysis, and especially the analysis of "turn-taking" in a conversation. This chapter gives an overview of the challenges in evaluating media synchronicity in real-time communications, outlining appropriate tasks and methods for subjective testing and how in-depth analysis of such tests can be performed to gain a deep understanding of the effects of delay. The insights are based on recent studies of audio and audiovisual communication, but also show examples from other media synchronization applications like networked music interaction.

G. Berndtsson (✉)
Digital Representation and Interaction, Ericsson Research, Stockholm, Sweden
e-mail: gunilla.berndtsson@ericsson.com

M. Schmitt
Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands

P. Hughes
British Telecom Research and Innovation, Martlesham Heath, Ipswich, UK
e-mail: peter.j.hughes@btinternet.com

J. Skowronek · A. Raake
Audiovisual Technology Group – Institute of Media Technology – Ilmenau University of Technology, Ilmenau, Germany

K. Schoenenberg
Clinical Psychology and Psychotherapy, University of Wuppertal, Wuppertal, Germany

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_9

Keywords Transmission delay • Latency • Subjective evaluation
Test method • Conversation analysis

9.1 Introduction

In order to provide users of a real-time communications system with a satisfactory experience there are two major requirements: first we need to ensure the propagation time between the endpoints is as short as possible—ideally no more than a few tens of milliseconds, and second the relative delay between the audio and video should be very small. In practice, this may not always be possible, so this chapter discusses the impact of failing to meet these requirements, with an emphasis on the human factors involved.

In order to conduct a proper system evaluation, it is important to understand the factors that influence the impact of delay: for example, the system context and user expectations. Typically, we might expect the difficulty of performing an appropriate evaluation to increase in line with the complexity of the application. At one extreme we could consider the trivial case of Internet radio: only one stream exists and in one direction only. Here the expectation of the user is simply that they can listen smoothly to the content without stretches, speedups, and breaks. The end-to-end delay is largely irrelevant and there are no other synchronization demands. If more sensory channels are added, such as for video or haptic data, this represents an increase in complexity because we must ensure there is correct synchronization between them. In audiovisual systems, this is commonly referred to as "lip-sync".

There has been a lot of research on the limits within which audio and video are perceived to be synchronized, at least for TV screens. In [35] it is stated that "subjective evaluations show that detectability thresholds are about +45 ms to –125 ms and acceptability thresholds are about +90 ms to –185 ms on the average, a positive value indicates that sound is advanced with respect to vision."

Real-time interactive communications are, of course, much more complicated and are the main focus of this chapter. Talking with people over a distance has become a ubiquitous commodity, taken for granted by most of its users. Yet these telecommunications systems can never be a perfect replacement for a real face-to-face conversation, and as participants get familiar with the features of conferencing systems they consciously or unconsciously adapt their behavior to compensate for difficulties they may encounter. This behavior is also influenced by a multitude of factors such as cultural aspects, personality, environment, etc. In other application areas, such as interactive music performances, precise timing is even more important.

After this introduction, Sect. 9.2 discusses factors that influence the impact of delay. The impact of delay in a system, application, or service will always be influenced by other factors such as the context of the situation and the characteristics of the users. Even with a careful evaluation design, these factors cannot be fully excluded from a test and thus have to be taken into consideration.

In Sect. 9.3, we then delve further into the context by gaining an understanding of how people interact in normal face-to-face conversations. This has been extensively studied in the linguistic discipline of Conversation Analysis. Crucial to this is the concept of "turn taking" which describes the organization of a conversation in terms of who talks when. This is implicitly negotiated by a multitude of verbal cues within the conversation and also by nonverbal cues such as physical motion and eye contact. An important feature is that the next talker is usually determined by who starts to speak first after the previous turn comes to an end—and, when there is video, by nonverbal cues such as gestures or looks. Unfortunately, delay makes it harder for the participants to be clear on who was to be taking the next turn, leading to a period of confusion over who, "has the floor".

Equipped with an understanding of how delay influences remote conversations, we turn to the actual evaluation of delay from a human perspective. We first present a number of studies conducted in different kinds of real-time communication scenarios. Section 9.4 covers audio and audiovisual conversation tests in both two-way and multiparty scenarios. With an understanding of how concrete delay evaluations can look like we turn to the general recommendations on test methods in Sect. 9.5. What kinds of scenarios and procedures are appropriate for tests, what questions should we pose to test participants, and how can we understand and interpret the results properly? A range of test methods can be required depending on the degree of delay. Several existing test methods use tasks that are insufficiently sensitive to delay, for example, since they are too demanding or complex. Some more interactive tasks, such as quick exchanges of numbers are sensitive to delay, but do not reflect normal conversations.

A related analytical process referred to as Conversation Surface Structure analysis, is based on the analysis of patterns of the temporal occurrence of utterances, without actually labeling their purpose. This approach does not rely on the content of the utterances and can, therefore, be simply built on voice activity detection algorithms. It also lends itself to deriving performance metrics from the interactivity as described in Sect. 9.6. The complexity increases in multiparty situations, since now the next speaker selection is less clear.

Section 9.7 describes how, with the help of the measurements obtained from subjective tests, models can be constructed that can, for example, be used in the planning of telecommunication networks. A number of international standards organizations such as the International Telecommunication Union (ITU-T) have produced recommendations for acceptable delays in telemeetings, as well as methods to evaluate their impact on speech quality. One of the most significant is recommendation G.114 [26] from ITU-T Study Group 12,[1] which covers one-way transmission time for speech communication in the absence of echo. G.114

[1]ITU-T Study Group 12 is responsible for recommendations on performance, Quality of Service (QoS), and Quality of Experience (QoE) for the full spectrum of terminals, networks, and services ranging from speech over fixed circuit-based networks to multimedia applications over networks that are mobile and packet based. The different areas are discussed under specific "questions"; telemeetings are discussed under Question 10. More details on this can be found at [36].

provides guidance on acceptable delays, and states, “Although a few applications may be slightly affected by end-to-end (i.e., “mouth-to-ear” in the case of speech) delays of less than 150 ms, if delays can be kept below this figure, most applications, both speech and non-speech, will experience essentially transparent interactivity” and also, “Regardless of the type of application, it is recommended to not exceed a one-way delay of 400 ms for general network planning…”.

9.2 Factors Influencing the Impact of Delay

We will start by taking a look at the many factors that influence the impact of delay in real-time communications. The experience of participants in a telemeeting is shaped by a multitude of factors. In addition to the system delay, the experience is very much dependent on the task at hand and the expectations of the individual users. Current analysis models for Quality of Experience (QoE) [43, 55], have divided these factors into three main categories: system factors, human factors, and context factors. To understand the results of a subjective evaluation, contextual and user factors need to be taken into account.

These main factors can be further categorized into a layered approach in which more specific factors are embedded within the more general ones. User and contextual factors can sometimes be hard to distinguish. For example, a certain role (user factor) can be linked to a specific work situation (contextual factor). Furthermore, there is a reciprocal link between the interactional context and the system factors. For example, in a group discussion with high-delay participants might switch to a slower or faster conversation style, depending on the task at hand.

The user and context factors shape in many ways the direct or indirect expectation users have of a system or service, which in turn greatly shapes the participant’s QoE.

9.2.1 System Factors

System factors include the technical aspects of the system or service in question. In audiovisual telecommunication applications, typical factors are the audio representation (encoding, spectrum), video representation (resolution, frame rate, encoding) and effects due to network transmission (bandwidth, loss, delay, jitter, synchronization). While we are focusing on delay and synchronization in this chapter, there are several system factors including audio and video encoding, and packet loss which can influence the impact of the delay. It is common that echo can occur in delay situations. Echo has usually a much more dominant effect compared to pure delay effects [20], so echo control should be addressed, as well as the problems of delay. If not mentioned otherwise in this chapter echo-free delay is assumed.

9.2.2 User Factors

User factors or human influencing factors describe the aspects that deal with the idiosyncrasies of individuals such as perception due to different roles, mood, preferences, expectations, and experiences. In telecommunication with delay, the sociocultural context may shape the expectations of the timing of the conversation. For example, in a direct comparison between North American native English and Spanish speakers, it was found that the Spanish native speakers leave shorter pauses in-between speaker changes, and overlaps are more common [4]. This lead to an impression of being rude from the English native speakers' viewpoint and in turn the Spanish native speakers perceived their counterparts as more distant. Similar findings could be observed in the interpretation of delay [59].

9.2.3 Contextual Factors

Contextual factors refer to the current session, the physical environment and other factors of the current situation which may shape the users' expectations. For example, physical distance might influence the user's expectation as there is usually an awareness that longer distances need a longer traveling time. If we assume physical media where signals propagate at roughly two-thirds of the speed of light in vacuum (as for example via copper wire), it takes about 50 ms to travel 10,000 km. Therefore, expectations might be lower when talking to someone on the other side of the world than when talking to someone in the office next door.

Similarly, users might be aware that they can expect a worse network quality when in motion, for example, riding in the train, compared to when at home.

Economic aspects can also influence users, for example, the expectations for a free service may be lower than for a paid one.

Multiparty telemeetings follow a different conversation dynamic than two-party conversations, which results in different thresholds for delay. As detailed in Sect. 9.4.5, there seems to be a systematic difference between multiparty and two-party conversation in terms of the impact due to delay.

A final factor is the sociocultural context, for example, whether we are chatting with friends or are attending a work meeting will shape our expectation and interactions. With higher familiarity, the participants are more likely to perceive subtle changes in the conversation style. Casual conversations are often faster paced than formal ones, which might lead to a higher sensitivity for delay [3]. An example how the situational context changes the sensitivity to delay was shown in [57], where adding a competition element to a delay evaluation task greatly influenced the results.

9.3 Conversation Analysis and Turn-Taking

Before we begin to examine the impact of delay in a conversation, we should examine how we interact with each other in a normal face-to-face situation. This behavior has been extensively analyzed in the linguistic discipline of conversation analysis.

9.3.1 Basic Concepts

In a free conversation, the organization of the conversation, in terms of who speaks when, is referred to as "turn-taking". This is implicitly negotiated by a multitude of verbal cues within the conversation, and also by nonverbal cues such as physical motion and eye contact. Turn-taking behavior is more-or-less universal, even across languages of very different structures [12], though there will be variations between members from varying cultures. This behavior has been extensively studied in the discipline of Conversation Analysis (CA), for example, [53].

Some useful concepts from CA are: the turn constructional unit (TCU), which is the fundamental segment of speech in a conversation—essentially a piece of speech that constitutes an entire "turn", and the transition relevance place (TRP), which indicates where a turn can take place between speakers. TCUs are separated by TRPs.

These processes enable the basic turn-taking process to take place, as described in [53] and shown in a slightly modified form in Fig. 9.1. As a TCU comes to an end, the first decision to take place is: has the current talker selected the next talker? If so, the designated new talker usually feels obliged to talk. Otherwise, any participant including the current talker will self-select; a process which generally works by the first person to speak gaining the right to the next turn.

Most of the turn-taking process is carried out subconsciously in line with what the participants consider to be appropriate behavior. Knowing when it is acceptable to take one's turn in a conversation is usually learned from an early age and becomes an important social skill.

Despite its advantage of providing a systematic approach to characterize how speaker changes occur, the model of Sacks has a few limitations:

First, the model focuses on turn-taking behavior based on verbal information provided by the speaker. However, speakers also use nonverbal signals to manage turn-taking, such as eye movements [40] or gestures [14].

Second, there are a number of other utterances that do not follow this turn-taking process such as backchannels (listener responses to indicate interest, or comprehension, for example, "hmm", or "uh-huh"), brief side-conversations, corrections, confirmations, and choral behavior such as laughter. We should note that in some highly interactive conversations, the underlying turn-taking process can completely be obscured.

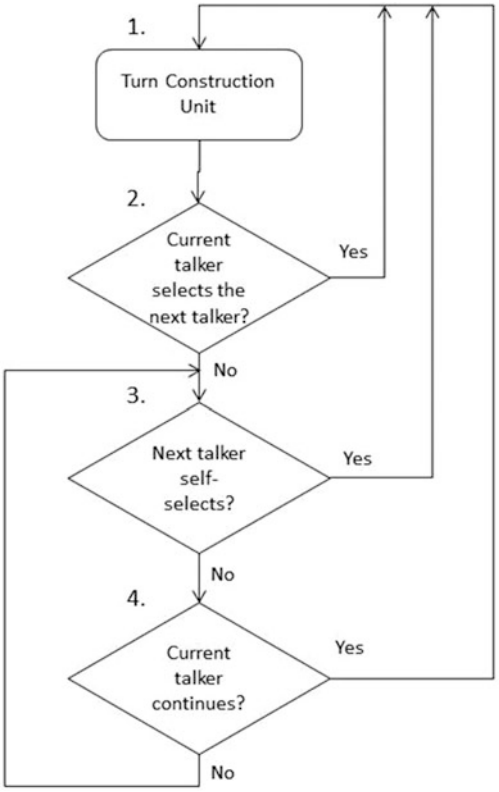


Fig. 9.1 Block diagram of the conversation turn-taking process

Third, interruptive behavior, that is, speaker changes at any time, is not allowed in this process. As a consequence, any attempt to start a turn outside a TRP is considered as "violative interruption" and is, therefore, often interpreted as ill-mannered or rude behavior.

9.3.2 Turn-Taking with Delay

In the presence of delay, the turn-taking process is disrupted because each participant has a different understanding of when the TRP occurs, and this effectively breaks the self-selection process. A new talker may believe they are starting to talk at the TRP, but the other participants will hear them sometime later when they themselves may have already started talking based on their own perception of when the TRP occurred. This leads to confusion amongst the group as to who has the floor, and can lead to the interpretation of the other's action as violative interruptions.

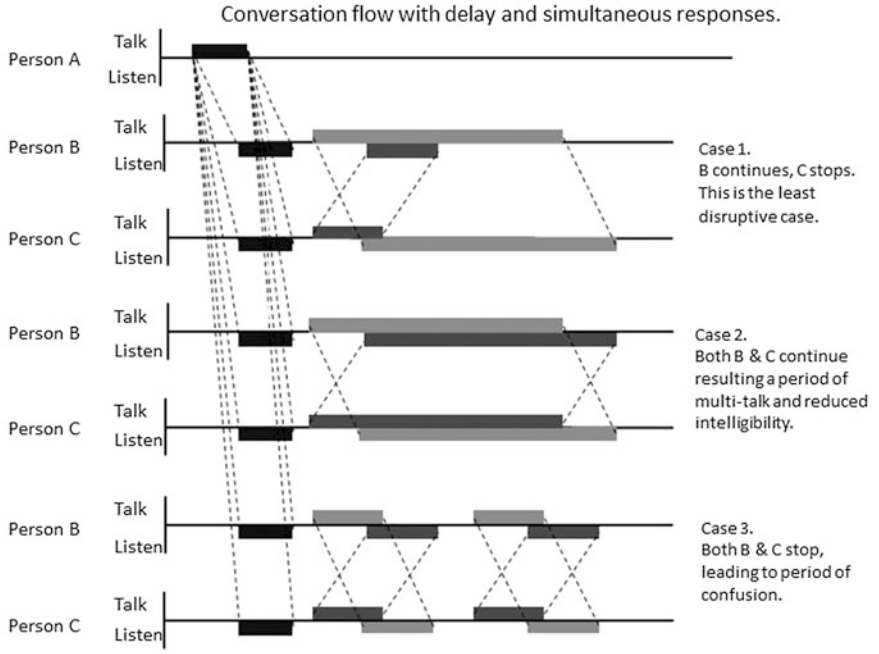


Fig. 9.2 False starts in 3-way conversation

A good example of how this type of confusion occurs can be seen in the following example, an occurrence sometimes referred to as a "false start". Consider a three-party conversation according to Fig. 9.2 where a talker has posed a question to the two other participants, who then respond more-or-less simultaneously. When they realize the other is talking they will either stop, or continue. This leads to the following events, which will probably be familiar to anybody who has taken part in a telephone conference.

1. One participant stops talking and the other continues. From a conversation analysis perspective, this might be the least disruptive case compared to the other cases. However, it still can be a significant disruption of the information exchange. For example, the participant who stopped talking did it for reasons of social appropriateness, even if his or her contribution would have been more beneficial from a content perspective. Or the participant who continues talking is confused for the short period of double-talk that he or she perceives, and might react or comment on that first instead of directly talking about the content.
2. Both participants keep talking, resulting in a longer period of double-talk. From a conversation analysis perspective, this likely has a moderate impact, since the conversation is still continuing. However, this also can lead to strong negative effects on understanding each other, and even on social aspects if the double-talk is (mis-)interpreted as rude or inappropriate behavior.

3. Both participants stop talking. From a conversation analysis perspective, this has the strongest impact: the conversation stops unexpectedly and in the period of mutual silence again a false start is possible. This may lead to multiple attempts to correct the false start; it may even involve other participants; and it may lead to a period of a meta-conversation trying to sort out who actually has the conference floor instead of continuing with the original conversation.

9.4 Subjective Tests Evaluating Effects of Delay

As described in the previous section, many factors are influencing a conversation and the impact of the delay. For this reason, it is important to assess this impact and the best way to do it is by running experiments with real people communicating using the system of interest. These tests are commonly referred to as "subjective tests". In this section, we will describe different subjective tests and knowledge obtained from them. We will also discuss the suitability of different evaluation methods to assess the impact of transmission delays during real-time interactions. Subjective tests in telecommunications fall into two categories: observational tests such as viewing and listening tests and conversation tests which focus on the interactive aspects such as delay, echo, and temporal effects. In this chapter, we are obviously mainly dealing with the latter.

Initially, we will focus on two-way conversations and then extend this to the various multiparty options. At the end, we will also touch upon music interaction over networks.

9.4.1 Two-Party Conversation Test on Delay Sensitivity for Different Test Tasks

A well-known subjective conversational experiment that examined the effect of transmission delay on speech quality in telecommunication quality was performed by NTT [39].

In this test, six types of tasks with varying temporal characteristics were used:

1. Take turns reading random numbers aloud as quickly as possible
2. Take turns verifying random numbers aloud as quickly as possible
3. Words with missing letters are completed with letters supplied by the other talker
4. Take turns verifying city names as quickly as possible
5. Determine the shape of a figure described verbally
6. Free conversation.

Both trained and untrained test subjects participated in the experiment.

The results show that the conversation task, the subject group and the degree of the subject's understanding of the technologies involved in propagation delay influenced the perceived speech quality to a large extent.

Recordings of the test conversations were used to analyze temporal characteristics. These analyses revealed that the speed of conversation switching decreased from Task 1 to Task 6.

To take turns reading random numbers aloud as quickly as possible, as in the first test task, had a detectability threshold of 45 ms for a one-way transmission. For the free conversation task, the delays were detected at 380 ms one-way.

In this test, the test participants experienced delay effects on communication quality for about 30 min before the test, which made them more aware and also more critical toward delay than in many other tests.

9.4.2 Audio-Only and Audiovisual Two-Party Conversation Tests on Influence of Delay on Conversations

Several two-party conversation tests were performed at Ericsson in 2006 to investigate the influence of delays on conversations and to develop a suitable test methodology for delay tests. Two audio conversation tests and one audiovisual conversation test will be described in this section [3].

9.4.2.1 Audio Tests Investigating Effects of Delay and Packet Loss

The goal of the first test was to evaluate how the perceived conversation quality is influenced by packet loss and delay. Different values of the end-to-end audio delay (from 160 to 600 ms) were combined with different amounts of packet loss (from 0 to 10%).

Five questions were asked in the tests. Continuous scales were used for the first three questions asked (the endpoint labels are given in brackets), and *yes* or *no* alternatives were used for Questions 4 and 5.

1. How do you judge the total quality of the communication? (Bad—Excellent)
2. How would you assess your ability to converse back and forth during the conversation? (Did you feel a vague irritation..?) (Bad—Excellent)
3. Did you perceive any quality impairments during the conversation? (All the time —Never)
4. Did you have any difficulty in hearing or talking over the connection? (Yes or No)
5. Would you accept the quality of this type of conversation? (Yes or No)

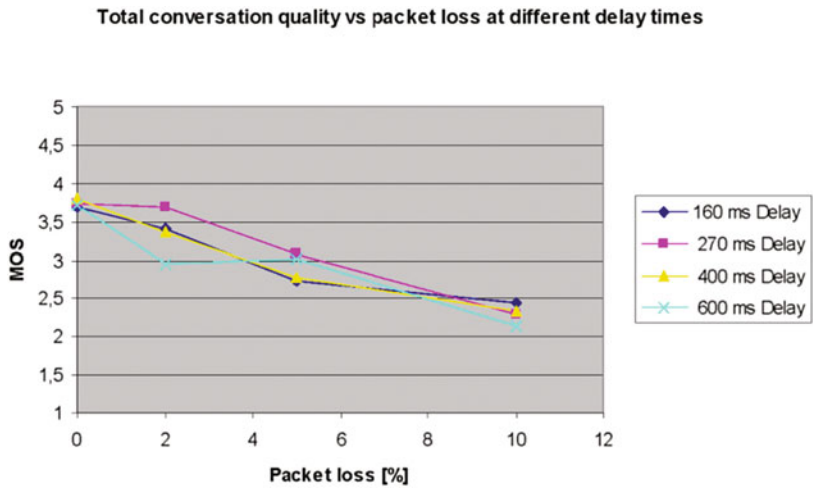


Fig. 9.3 Speech conversation test 1 with expert subjects. Total conversation quality in mean opinion score (MOS) versus packet loss for different delays. From [3]

The phrase "(Did you feel a vague irritation..?)" was added to Question 2 to make the test subjects more observant toward small conversation discrepancies without mentioning the word delay.

The Short Conversation Test Scenarios (SCT), described in [28] were used in the test. These scenarios have been developed to create a well-balanced conversation between the two participants. Each conversation lasts approximately 2.5 or 3 min. The test subjects are given a task similar to a daily life situation, for instance ordering a pizza.

The results show that the impact of packet loss was very clear, but the longest delay used in the test, 600 ms, was not perceived to degrade the quality of the conversation. The reason for this was probably that there was some time needed for information retrieval in the conversation task, making the delays less noticeable. The Mean Opinion Scores (MOS) for the perceived overall quality are shown in Fig. 9.3.

The interactive Conversation Test Scenarios (iSCT), also described in [28], were tested informally. These scenarios contain short utterances, for instance, quick exchanges of numbers or names, but they were only slightly more sensitive to delay. These conversations were also very structured, so when the conversation partner was supposed to reply, the other test subject waited politely for the answer, and there was not much double-talk.

Both the SCT and iSCT scenarios have been applied in several other delay tests with similar test results as in the tests described in more detail above [22, 47].

To examine if the delay is more disturbing when using other conversation tasks, informal tests of several tasks recommended in [24, 29, 30] were performed.

For many of these tasks, it was not easy to understand if a delayed answer by the conversation partner was due to the need to think before answering, or if there was transmission delay. Most of the tested conversation tasks required some searching for information or thinking about what to answer, so pauses were created that made the delays less obvious.

For the second two-party audio-delay test, the test participants got a paper with the same story but with different missing parts. A requirement for a conversation tasks is that the participants should be approximately equally active in the conversation. In this case, these requirements were met, as they had the same number of missing sentences in their texts.

One participant started to read the story and the other participant read the text silently, and interrupted when the reader missed a part of the text. Then he read that part and continued to read the text until he was interrupted. As the reader continued to read until he heard the other person speak, this conversational task led to several double-talk situations. In addition to the text reading, the participants were allowed to have a short discussion to make the conversational task more natural. Each conversation lasted between 3 and 5 min.

In this test, two longer delay values were added compared to the previous test (800 and 1000 ms), but only two packet loss values, (0 and 5%)were used.

The test was run with both experts and "naïve" participants. The "naïve" participants were Ericsson employees, who did not work in the media area.

The end-to-end delays were more easily noticed in this test than in the previously described one. It was more obvious that there was a delay when it took a long time before the person reading reacted to the interruption from his conversation partner.

The expert group gave slightly lower scores for longer delays when there was no packet loss than the naïve group. It can be seen in Fig. 9.4 that the conditions with 0% packet loss are considered to be more difficult and are less accepted when the delay time is over 400 ms.

9.4.2.2 Two-Party Delay and Audio–Video Synchronization Test

The goal of the test was to investigate how synchronization of audio and video affects the judgment of audiovisual conversation quality and to examine whether audio–video synchronization requirements depend on the end-to-end delay.

The test method can be characterized as follows:

- Free conversations during about 3 min for each condition. The conversation partners in every pair knew each other.
- Audio and video end-to-end delays (14 combinations).
- Both expert and "naïve" (nonexpert) subject pairs.

The same questions as in the audio conversation tests were posed about overall quality, interaction quality, difficulty in hearing and talking and acceptability, but

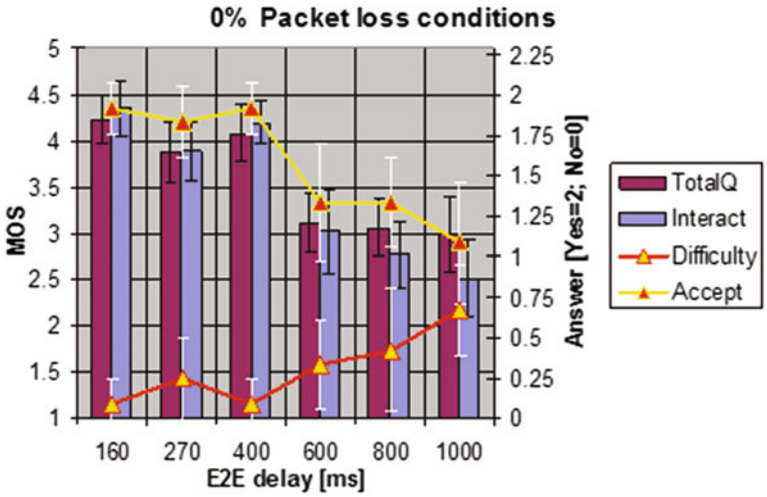


Fig. 9.4 Expert judgments for different delay times. Conditions with no packet loss are shown in the diagram. Ratings for questions (1 and 2) about total quality and interaction refer to the left axis and the answers to questions about difficulty and acceptance refer to the axis to the right. The 95% confidence intervals are indicated with black lines for the total quality and interaction questions and with white lines for the questions about difficulty and acceptance

the third question in the audiovisual test was: "How do you judge the synchronization between audio and video to be?" (Bad—Excellent).

The results showed that the conversation quality judgments were more affected by the audio–video synchronization than the end-to-end delay, see Fig. 9.5. Also in this test, the "naïve" subjects were less critical toward long delays than the experts.

Low synchronization scores were given by both experts and nonexperts when the audio was 200 ms or more before the video. As the audio was gradually delayed to coincide with the video delay of 500 ms, the synchronization was rated as increasingly better. The condition where the audio was delayed 100 ms more than the video was not judged to be worse than the condition with both audio and video delayed by 500 ms, except for the conversation difficulty. This asymmetry was expected as the sound normally reaches a person after the associated visual information, for example, the thunder comes after the lightning.

The results indicate that, in terms of quality judgments, it is more important to synchronize the audio and video than to send the audio as fast as possible in audiovisual conversations. Usually video is delayed more than audio, so generally, for e.g., video telephony, audio should be delayed to obtain audio–video synchronization, at least for delays up to 600 ms.

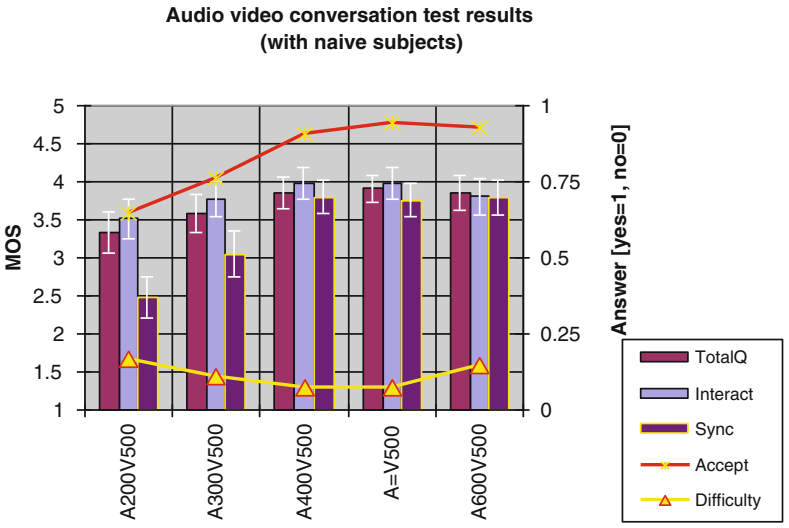


Fig. 9.5 Ratings for all conditions with 500 ms video delay combined with different audio delays. "A200V500" indicates an audio E2E delay of 200 ms, and a video E2E delay of 500 ms. The ratings for the first three questions (TotalQ, Interact, and Sync) relate to the left axis. Questions 4–5 (Difficulty and Accept) relate to the right axis. 95% confidence intervals are indicated with white lines. From [3]

9.4.3 Conversation Tests on the Impact of Delay, Audio Bandwidth, Task Type, and Group Size

A series of two-party and three-party conversation tests was conducted in [60] in order to investigate the impact of delay, task type, and group size on conversational quality and conversation structure.

In those tests, groups of either two or three participants had a number of telephone calls with different transmission delays on the line, ranging from 50 ms to 1600 ms one-way. In addition, different tests were conducted with either a narrowband—NB (300–3400 Hz), wideband—WB (50–7000 Hz), or fullband—FB (20–20000 Hz) telephony system. Furthermore, different conversation tasks were used in different tests: 2-party Random Number Verification—RNV [28], 2-party Random Number Verification Timed—2RNT [57], 3-party Random Number Verification Timed—3RNT [58], 2-party Short Conversation Test Scenarios—2SCT [28], 3-party Conversation Test Scenarios—3CT [48], shortened versions of them—3SCT [61], Celebrity Name Guessing—CNG [60]. In total, there were eight tests; an overview is given in Table 9.1.

Focusing on MOS ratings as indicators for overall conversational quality, the following results were found by means of Analysis of Variance (ANOVA).

Delay had a significant impact on MOS in all experiments except in one experiment (task: 3CT, audio bandwidth: FB). Post hoc tests showed significant

Table 9.1 Overview of tests from [60]

Test ID	Communication modality	Audio bandwidth	Tasks	Group size
A.NB.2a	Audio-only	Narrowband	2SCT, 2RNV, 2RNT	2
A.NB.2b	Audio-only	Narrowband	2SCT, 2RNT	2
A.NB.3	Audio-only	Narrowband	3SCT, 3RNT	3
A.WB.2	Audio-only	Wideband	2SCT, 2RNT	2
A.WB.3	Audio-only	Wideband	3SCT, 3RNT	3
A.FB.2	Audio-only	Fullband	2SCT, 2RNT	2
A.FB.3	Audio-only	Fullband	3CT	3
V.WB.2	Audiovisual	Wideband	CNG	2

differences between most conditions, whereas often low-delay conditions were not significantly different.

For the NB or WB tests, no significant main effect was found, but a significant interaction of group size and delay. For delays below or equal to 400 ms in the WB case, the MOS ratings were lower for three-party than two-party calls; for higher delays, the ratings were on a similar level.

9.4.4 Audio and Audiovisual Multiparty Tests with Asymmetric Links

The sensitivity for delays in audio versus audiovisual conversations and the different experience from sitting alone in a room compared to in a group room were examined in [3]. Two test tasks were used, free conversations and a quiz game where the participants cooperated to guess a word. The end-to-end delays values in the test were 200, 400, and 800 ms.

These delay tests were performed using an Ericsson-developed large screen conference system with loudspeakers. The test setup was similar in the audio and audiovisual tests, but the screen was not used for the audio-only telemeetings:

- Audio (fullband stereo)
- Video resolution 720p, bit rate: 1500 kbps

Nonexpert test subjects participated in the test. They formed groups of 5–7 persons. Three silent rooms in the multimedia lab at Ericsson Research were used for single-test participants and a larger silent room was used for a group of test participants. During the break between the sessions, the participants switched places so that all subjects experienced both sitting alone and together with other persons in a room.

Both the quiz and free conversation test tasks were performed during 6 min per condition, so that every participant could have enough time to speak. For the free conversations, the discussion topic was either of the test subjects own choosing or

picked from a list of suggested topics. The test persons were instructed to try to divide the speech activity equally between them. They were not allowed to talk about the quality and properties of the system as this could influence the other participants' assessment of the system, but they had the possibility to write comments on a paper if they noticed something special during the test.

The participants were to assess how well the videoconferencing system functioned during communication with other persons judging the:

- Overall quality
- Interaction quality
- Audio or audiovisual quality, depending on the test (without/with video)
  - Continuous scales, endpoints "Bad" and "Excellent"
- Acceptability
  - Yes or No answers

Generally, the quality was considered to be worse for longer delays. The test subjects were slightly more critical to delays in the audiovisual telemeeting test. The interaction quality was considered lower for longer delays, especially for the small rooms. The acceptability was also lower for longer delays, as expected. It could be seen in the video if the reactions to what was said were slow, which might explain the lower scores for longer delays in the audiovisual case.

The quiz game was not as delay sensitive as the free conversation. The reason for this could be that it took some time for the subjects to agree on which questions to pose. Further, paying attention to the game rules and concentrating on the game as such may have decreased the attention paid to delay. Also in a game, delay effects may have been attributed to the game partners rather than to the line.

The test results revealed that delays were most noticed in single-user rooms, during the audiovisual test and during free conversations. The interaction and acceptability scores were most affected by delay, see Fig. 9.6.

The situation was different for the persons sitting in the large room compared to the persons sitting in the small rooms. There was, of course, no delay between the persons in the large room, but there was always at least 200 ms delay between the rooms. As a consequence, each person in the large room had a significantly lower number of delayed interactions than the persons in the small rooms. Hence, the persons in the small rooms were more exposed to delay conditions.

The participants were interviewed after the test. They commented that it was easier to enter the conversations when they were in a room together with other persons, but easy to forget the persons that were sitting alone in a room.

The persons in the single rooms sometimes felt somewhat disconnected from the conversation, but they looked more at the screen and noticed the delays more.

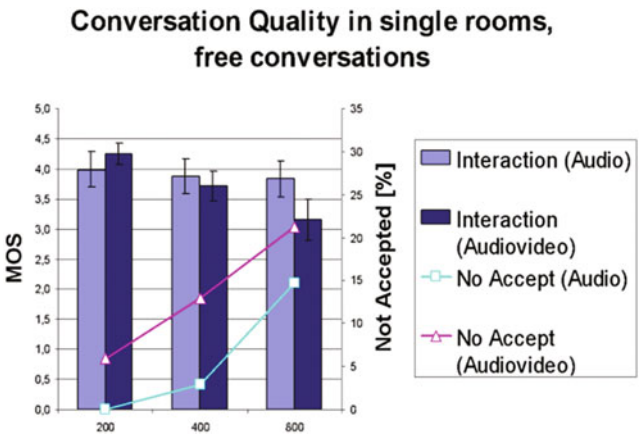


Fig. 9.6 Audio and audiovisual test in single rooms. Results shown for Interaction and Acceptance votes during free conversations. The 95% confidence intervals are marked with black line for the question about interaction. From [3]

9.4.5 Multiparty Desktop Telemeeting Tests

Centrum Wiskunde & Informatica (CWI) conducted a series of studies to investigate the effects of delay in multiparty telemeetings using desktop hardware. Multiparty videoconferencing sessions are gaining traction since current hardware and broadband connections provide the capabilities. The goals of these experiments were to gather thresholds for upper and lower delay limits, investigate differences in the perception based on roles and interaction.

In exploratory pre-study sessions, the desert survival scenario was tested [32]. The scenario was eliciting a lively conversation however, the variance of how much time groups needed was varying from 25 min to 2 h. To keep the topic but have a structure that is easier dividable into rounds for different conditions, a variation of this group building task was chosen [5]. In this scenario, questions about surviving in the wilderness were chosen and participants had to cooperate to together choose one answer from three suggestions.

Further, in the not time-constrained sessions in the pre-study it could be observed that nearly always one of the participants got a central role that kept driving the discussion. This is in line with research about small group discussions [13]. To better control this effect, one of the participants was randomly chosen to be the moderator. The moderator was asked to submit the final group answers and move the discussion along to keep the 10 min time constraint per round. During the study, all participants were alone in separate rooms. Before and after the study, there was a briefing (short introduction of research, institute, and participants) and debriefing (semi-structured interview about the study) session, jointly with all participants of a given group.

The study was conducted on Desktop PCs with webcam and headset and conducted on the premises of the institute. The videos were transmitted in SD Quality (640 × 480 px, 30 fps, H.264). A detailed description of the test system and how the manipulation and control of delays was achieved can be found in [55].

The study concerning symmetric delays (i.e., all participants have the same delay) tested 4 delay conditions (75, 500, 1000, and 2000 ms). In the asymmetric study, only one participant (either the moderator or a random participant) had an additional delay (500 or 1000 ms). After each condition, the participants rated the delay by three different categories (quality of the connection, how annoyed they were by the delay, and how much they noticed the delay) on a nine-point Likert-type scale.

9.4.5.1 Symmetric Experiment

Figure 9.7 shows the effect that the experience gets worse with higher delay, confirmed by statistical analysis [56]. The scores are normalized in a way that a lower score means a worse experience (e.g., for annoyance that the participants were more annoyed). The further pairwise comparison revealed that the quality is perceived noticeably worse between 500 and 1000 ms delay.

Even though the ratings in the high-delay conditions are still relatively good, a behavior change was observed. Groups often abandoned the free conversation and switched to an explicit organization mechanism. As participants would talk over each other's utterances all the time, the moderator ended up explicitly managing the turns, by individually addressing each participant. Most of the time, this delay was noticed at a 1000 ms added delay and some groups switched to a more explicit organization of the conversation. At 2000 ms, nearly all groups switched to the explicit organization-type conversation structure.

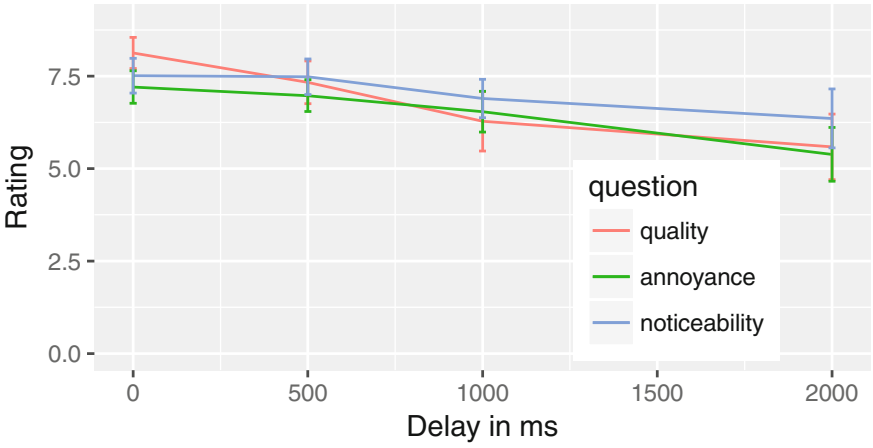


Fig. 9.7 Average questionnaire quality ratings with 95% confidence intervals

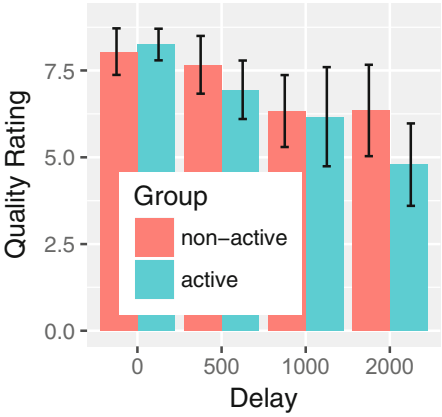


Fig. 9.8 Average quality ratings (on a 9-point scale) clustered by blocks percentage duration of non-active participants (red solid), and active participants (blue striped) with 95% confidence intervals

9.4.5.2 Qualification by Activity

This experiment investigated the hypothesis that participants who have a more central role in the conversation are also more affected by delay (as a User factor, see Sect. 9.2). The scenario was designed such that the moderator would probably have a central role in the conversation. However, since the moderator was randomly chosen, in some groups he or she may still be quite reserved and another participant took the de facto moderator role. To determine which participant had a central role, the participants were clustered by percentage of the talking time per round into *active* or *non-active* participants with the k-means algorithm.

Figure 9.8 shows the results for the quality ratings. Pairwise comparison revealed that active participants experienced a significant quality drop between 0 and 500 ms, while the non-active participants noticed a similar drop between 500 and 1000 ms. Similarly, the comparison between active and non-active participants within the same condition showed that it was significantly different in the 500 ms condition but similar in all other conditions.

9.4.5.3 Asymmetric Experiment

The asymmetric delay study had a similar design (5 participants having an ad hoc group discussion over a videoconferencing system), but in this trial only one of the participants had an added delay of either 500 or 1000 ms. The statistical analysis showed that the delay had an impact on values starting from 1000 ms added delay, independent of whether the moderator or a randomly chosen other participant was considered. This disturbance was noticed by all participants in the session, not just by the ones having a delay. However, the impact of only one participant having a delay (asymmetric setting) on the QoE of the whole group is still smaller than for all participants having a delay (symmetric setting). The average ratings for quality with 95% confidence intervals of both experiments are plotted in Fig. 9.9. The

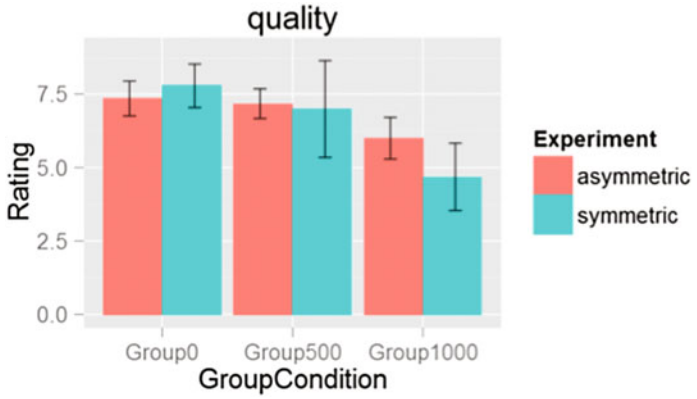


Fig. 9.9 Responses to quality by group condition for asymmetric and symmetric conditions

conditions Group0, Group500, Group1000 denote 0 ms, 500 ms, or 1000 ms of added delay respectively. In the symmetric case this applies to all participants, but in the asymmetric case to just one participant. Similar to the symmetric case, the whole group experiences a drop in quality with 1000 ms added delay. As can be seen in Fig. 9.9 the groups within which only one participant had added delay still experienced an overall better quality than the groups in which all participants had an added delay.

In the debriefing, it was reflected that participants did not notice one participant as particularly delayed. When asked to guess whether some participants were more delayed than others, most people chose themselves if at all. The comments reflected that participants were more noticing general communication problems in the session. This may be an indicator that the QoE of participants is not only reduced when they themselves are involved in communication problems, but also when they are just noticing that others have connection problems.

9.4.5.4 Multiparty and Two-Party Conversations

In the conducted studies regarding multiparty desktop videoconferencing a delay of up to 650 ms was seldom noticed by less active participants. Even with a delay up to 1150 ms, though the delay was noticed, many groups sustained a normal conversation. These thresholds are higher than results from previous research in two-party conversations [64, 67] but similar to findings for up to 800 ms from the multiparty study described in Sect. 9.4.4, Berndtsson [3]. The results from the symmetric delay experiment, Wang [67] and Berndtsson [3] are shown in Fig. 9.10. The label "*general*" refers to the ratings obtained without qualification by activity (see Sect. 9.4.5.2) while "*active*" and "*non-active*" refer to the corresponding activity groups. The studies used different setups so the plot is only meant to illustrate the trend. All three studies used the same question, about overall quality

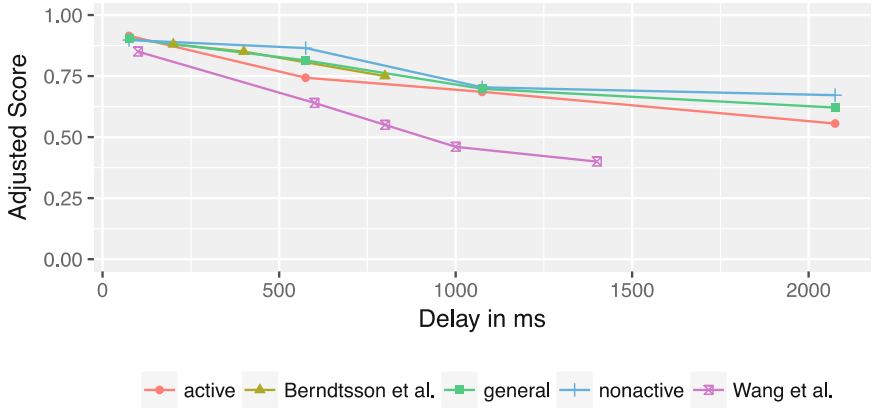


Fig. 9.10 Comparison with [3, 67]. Adapted from [55]

(ITU-T P.805 [28] in [67] translated to Chinese) with different scales (continuous, 5-point, and 9-point). In the plot, the scores are adjusted to the range 0–1. In [67] the ratings were presented in dependence to the average length of talkspurts. For comparison, the scores according to the average length of turns in the experiment (7.9 s) were used. The plot shows that the perceived quality in the multiparty studies declines much slower than in the dyadic study.

9.4.6 Music Interaction Over Networks

The main focus of this chapter is on voice conversations since these represent the majority of interactive multimedia communications, but also communication with music provides a number of opportunities: rehearsals, distributed music performance, and distance tuition.

Attempts at remote musical performance were made during the mid-1990s, initially using 128 kbit ISDN connections, and a number of ideas for networked media performance were presented in a 1998 AES white paper [1]. Later in 2000, a team at Stanford University's CCRMA presented a networked concert between two venues on the university campus, both with live audiences using the campus intranet and TCP protocol [10].

In 2005, a custom-built network framework called GIGAPOPR was described by Kapur [38]. Here, multichannel uncompressed audio, video, and MIDI data was transmitted between an arbitrary number of nodes, and was used for performing a special composition based on North Indian classical music with performers based in McGill University in Montreal, Canada, and Princeton University in New Jersey, USA. The system round-trip latency was measured at 120 ms and the media performance was composed with this delay in mind.

For most music performances, the acceptable delays are much smaller than for speech. According to Kapur, a one-way latency of around 20–30 ms is required, which is considerably less than the 150 ms accepted for voice communication in [26]. This figure ties in well with practical experience: most musicians are used to coping with latencies of 10–20 ms, simply due to the propagation time of sound—approximately 3 ms/m.

In practice, most VoIP products do not have delays that low: typically, VoIP system buffering and codec delay results in a minimum latency of 60–100 ms.

More recently, ultralow latency devices targeted at online rehearsal and music performances over distances make it possible to achieve a low enough delay.

The distances from the instruments to the microphone and from the loudspeakers to the listeners' ears also add to the perceived delay. Therefore, it is an advantage to wear headphones if you want to keep the overall delay short.

In an investigation by Farner [18], the experimental setup described by Chafe and Gurevich in [11] was used in real and virtual acoustic environments under the influence of a delay. They found that the tempo decreased the longer the delay was, and that it actually increased with low delays.

Music tuition over a distance is of major interest since it allows musicians in remote locations access to a tutor, and especially makes it practical for players of unusual instruments to obtain tuition. Videoconferencing services can be used for this. For example, some private tutors advertise tuition on Internet resale sites, and a number of Internet-based agencies offering a range of instrument tuition have recently emerged. Tuition using high-quality proprietary multimedia communications was studied as part of the European Seventh Framework Program (FP7) collaborative project "TA2—Together anywhere, together anytime" [63].

In [15] a number of trials were run to investigate tuition of music students using the high-quality videoconferencing equipment. Starting by examining the interaction between teacher and student in a co-located lesson, the study went on to see how the interaction changed when the videoconferencing link was used. The general conclusions from this were:

- The success of one-to-one tuition depends on the success of the interaction, and therefore trust, between student and tutor.
- The musical score has a pivotal role in the copresent lesson, constantly being referred to by both teacher and student. Remote teaching systems must reflect this.
- Effective lessons were possible over the videoconferencing system, but need supplementing with face-to-face lessons.
- Good audio quality is paramount.
- Multiple cameras are very beneficial to allow greater shared content, e.g., the score.
- Any form of joint playing, including the student playing to the tutor's set rhythm was impossible with the latency (approximately 200 ms) in the test system.

Here, too, the presence of interactions or turns obviously moderates the perception of delays, and a more listening-/viewing-type scenario with less frequent turns (e.g., comments by the tutor vs. playing to a remote rhythm) is affected by the delay to a minor extent.

9.5 Recommended Delay Test Methods

The described tests in Sect. 9.4 have led to an understanding of what is important to consider when designing and performing a subjective test. The first thing to consider when designing a test is the research question addressed, that is, what is to be investigated? What are the goals of the test? Which types of situations are most important to investigate? If a telemeeting system is to be tested, the normal/desired usage should be reflected in the test. Is it voice-only communication or audiovisual communication? Will there be communication link asymmetry?

A proper test design is needed when it comes to aspects such as test task, test protocol, test subjects, training session, instructions, and quality assessment questionnaires and scales.

Observational tests could be used for pure audio–video sync acceptance tests, but to evaluate the impact of delay on group interaction, some kind of conversation is needed.

The tests on group interaction can be divided into two main types:

1. Holistic methods that involve assessing the meeting as a whole. These are typically two-party or multiparty conversation tests.
2. Micro-feature methods focus on a specific aspect of talker interaction such as task efficiency, double-talk, or turn-taking.

There are several standardized test methods described in recommendations such as [27, 28, 31, 32]. Guidance is provided in [32], Clause 8 on how to select a suitable test method for a multiparty telemeeting test. Special considerations need to be taken into account when performing tests to assess the impact of transmission delays in telemeetings, as they are described in this chapter and also in [33]. Further information on tests for audio and speech quality evaluation can be found, for example, in [2, 37, 45, 47, 62].

9.5.1 Task Design

As mentioned in the previous section, the choice of test task is very important especially for a delay test. Several standardized test tasks are not delay sensitive, but could be appropriate for testing of other impairments such as packet loss.

In a conversational test, the test task should reflect a normal conversation situation and allow for interruptions from the subjects. Free conversation is suitable for both audio and audiovisual tests, since there is no need to read a written instruction during the test [3]. A rapid turn-taking makes it easier to notice delays in conversations. To stimulate a more interactive conversation, the participants could be encouraged to pose questions and try not to be too polite. Some kind of competition, such as winning an extra price for the fastest group, might motivate subjects and lead to a more interactive conversation flow [57]. Furthermore, when it comes to audiovisual tests, the test task should trigger subjects to use both the auditory and visual channel, in order to enable them to detect delay also in the visual channel.

A test methodology for free conversations can be found in [32], Appendix 3. Other recommended test tasks for audiovisual quality evaluations regarding delays, such as the Survival task can be found in [32], Appendix 5 and in [33].

9.5.2 Test Conditions

If the goal is to assess the influence of delay, the test will typically include conditions with different amounts of delay. Sometimes it is interesting to vary the conditions tested in some other way, for instance, to investigate the relative influence of audio–video synchronization versus synchronized audio and video with longer delays.

The conditions have to be chosen carefully because usually a conversation with a duration of several minutes is needed for each condition to be evaluated. The test time should not be too long, not more than around 15–20 min per session so that the test subjects can stay focused, and using around three such sessions per test is recommended. Additional time is needed for instructions, preliminary tests, and pauses between the sessions, so usually test subjects should be invited for a 2 h time slot.

The test design should preferably be balanced so that all participants experience all different delay conditions. The conditions should be presented in a different random order for each group, so that ordering effects are averaged out.

9.5.3 Types of Assessments

The most relevant questions for the particular test should be chosen and it is recommended to ask the least number of questions needed. This way, the test participants will be able to focus on the most important aspects of the test.

For delay tests, it has been shown that questions about the amount of effort and impairment in the communication reveal the impact of delays more than questions about quality. The problem with asking for the quality alone (see Q5 in Fig. 9.11) is that participants might not be aware of the delay and will rate the connection as

How would you judge the effort needed to interrupt the other party (or parties)?

No effort ——————————— *Extreme effort*

Did you perceive any reduction in your ability to interact during the conversation?

Imperceptible ——————————— *Very annoying*

How would you assess your ability to converse back and forth during the conversation (did you feel a vague irritation?)

Bad ——————————— *Excellent*

How do you judge the synchronization between audio and video?

Bad ——————————— *Excellent*

Opinion of the connection you have just been using

Bad	*Poor*	*Fair*	*Good*	*Excellent*
☐	☐	☐	☐	☐

Did you have any difficulty in talking or hearing over the connection? ☐ *yes* ☐ *no*

Would you accept this conversation quality? ☐ *yes* ☐ *no*

Fig. 9.11 Example questions suitable for assessing the effect of delay

excellent while in reality the communication was severely hindered but the problems attributed to the communication partners rather than the system [59]. Asking for judgments on the delay problems directly may influence subjects and lead to conversation behavior that differs considerably from natural conversations. Hence, questions in respective ITU-T Recommendations are of more indirect character.

The ITU recommends scales of different granularity (from 5-point discrete to continuous scales). The 5-point absolute category rating (ACR) scale is the most widely employed one. However, no advantage of a particular scale could be shown [23]. Continuous scales can, in theory, provide the finest results, however this depends on how precisely a given effect can be perceived. For continuous scales, it is important to not provide tick marks along the scale, as these will influence the results [44]. Some examples of questions suitable for evaluation of the effects of delay are shown in Fig. 9.11.

More scales that can be used for conversation tests can be found in [27, 28, 31], and in [32], Appendix 3.

The participants can be encouraged to write comments during the test and take notes if anything unexpected happened during the test. It is also recommended to ask the subjects some questions after the test about their test experience. This is an important complement to the voting scales. Aspects that are not explicitly asked for can then be commented.

9.5.4 Test Subjects

The amount of previous experience is an important factor when deciding which type of test subjects are needed for a given test. If the goal is to evaluate how an average person perceives a given effect, persons should be invited that do not work on the evaluation of telecommunication qualities or delay-related tasks (typically referred to as "naïve" subjects). The test subjects should preferably be similar to the group of persons that will use the system to be tested.

When evaluating the effects of delays in a system it may be an advantage if the test subjects know each other. Then their conversations are often more spontaneous and they might more easily detect delay rather than thinking that the other's personality may be hindering the conversation [3, 59]. Expert subjects may be invited when the goal of the study is about identifying delay detection thresholds, or when especially critical users are to be represented [39].

9.5.5 Instructions and Training Session

A paper with the written test instructions should be given to the test subjects when they arrive for the test. They should read the paper and then the test instructions should also be given verbally, with the respective wording maintained as constant as possible across the participant groups of a given test. It is important that the test subjects have the possibility to ask questions about the test methodology. In case of a test for an unguided assessment [37] of the effect of certain system conditions on quality, nothing should be mentioned about the technical background of specific test conditions in the test, so in this case the term *delay* should not be mentioned, as this may make the test subjects more observant to delays than they normally are. When targeting more diagnostic tests, e.g., aiming at a delay threshold detection, the instructions may explicitly include the mention of delay.

It is advised to present the instructions with all participants in the same room, so that they will experience a face-to-face meeting at the start and be able to compare that experience with the experience using the telemeeting system. This is especially important if the test persons did not know each other before the test. It is noted that it may also be desirable that subjects do not know each other, for example, when the effect due to delay beyond the impact on quality evaluation is to be investigated. For example, in [59] the attribution of delay effects to personality traits was investigated.

After providing the instructions, the test participants take their test seats. The pretest preferably starts with a short presentation by each participant, so that everybody gets the opportunity to see and hear all participants through the conference system.

Examples of instructions can be found in [32].

9.5.6 Comparison of Subjective Opinion and Performance Measures

As mentioned in the introduction, users are not always aware of the presence of delay even if they do notice its effects. A result of this is that subjective tests targeting quality ratings, which essentially measure the subject's impressions, may not provide a good measure of the delay impact. Of course, the subjects could be told about the presence of delay in the test system, but this raises the risk that more critical quality judgments might be used. Nevertheless, if subjective opinion scores are used, care should be used in the choice of questions presented to the subjects. For example, questions addressing the "Ease of conversation", or "Difficulty in communication" are likely to be more revealing than questions on "Quality".

An alternative test strategy is to use a test that focuses on the performance or behavior of the subjects while carrying out a specific task or tasks under differing amounts of delay. These types of tests have the major benefit that they provide a measurement which will not depend on the awareness of the subjects. Some of these measures are described in [32].

One difficulty is that we want the subjects to interact normally as they carry out the task, but it can be quite difficult to design a delay sensitive task which also allows the subjects to develop a natural conversation. The task should also be quite simple in order to reduce variations in performance due to the subjects' innate abilities.

A good example of a test that measures how much time it takes to complete a specific task is the "Task Effectiveness Test" described in [34]. This test involves measuring the rate at which multiple participants can simultaneously exchange data between themselves using the communications system, and comparing it to their performance at the same task in a face-to-face meeting. This results in an effectiveness score for that system, so it can be compared to other systems. Many features of a conference system, including audio quality, spatial audio and/or video, and delays will influence the effectiveness score.

9.6 Conversational Surface Structure Analysis

The previous section described recommended methods to perform tests in which participants are rating the communication link. However, as discussed in the introduction, the impact of delay is not always reflected in such subjective ratings. Therefore, an additional way to characterize the impact of delay is to conduct a *Conversational Surface Structure Analysis,* an approach that is presented in this section. The idea of *Conversational Surface Structure Analysis* is to extract parametric information about the on–off patterns of speech in a conversation, an idea introduced by [6–8]. Once extracted, this information can be used to characterize the impact of transmission delay on the conversation.

9.6.1 Method and Parameters

The extraction of such parametric information consists of two main steps. In Step 1, a statistical description of the different speakers' utterances and turns in terms of so-called *Conversation States* as well as accompanying temporal data are derived, using automatic *Voice Activity Detection (VAD)* applied to multitrack audio recordings of the conversations (e.g., captured at the sending devices or at the conference bridge). *Conversation States* refer to the number of speakers that are active at the same time: silence (no active speaker, S in Fig. 9.12), single-talk (one active speaker, $I_{N,x}$), double-talk ($M_{N,xy}$), triple-talk ($M_{N,xyz}$), etc., where N denotes the number of interlocutors in the telemeeting, and x, y, and z are indices for the, respectively, active speakers. Accompanying temporal data refers to the information when the conversation was in which state. A visualization of such a state model is given in Fig. 9.12.

In Step 2, one computes a number of parameters from the state model that comprise simple statistics or describe aspects of the conversation progress. Several

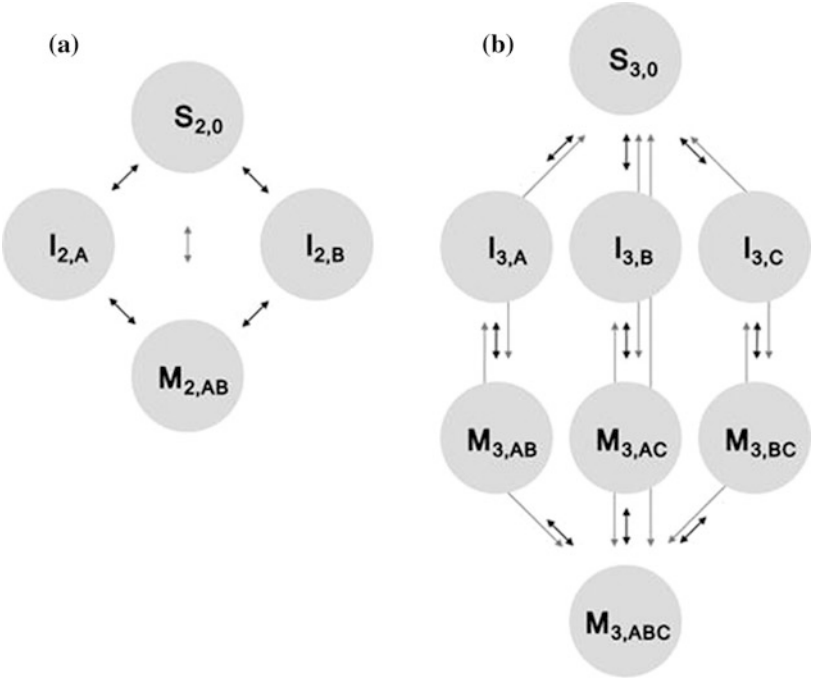


Fig. 9.12 Visualization of a conversational state model for **a** two-party and **b** three-party call, from [60]. The letters refer to silence (S), one individual talker (I) and multiple talker (M); the letters in the indices to the persons (A, B, C) and the numbers in the indices to the group size (2 for two-party, 3 for three-party)

Table 9.2 Overview of conversation parameters used in literature. For a detailed discussion see [60]

Conversation parameter	Literature
Duration of call	[51, 65, 21]
Number of words	[41, 42]
Number of backchannels	[46]
State probabilities	[6–9, 19, 46]
State sojourn time	[21]
Speaker alternation rate	[21, 16]
Conversational efficiency	[39, 54, 66]
Involuntary/unintended interruption rate	[16, 50, 52, 64]
Intended interruptive interruption rate	[17]
Transition tendencies	[8]
Conversational synchrony	[54, 66]
Conversational interactivity	[54]

such parameters can be found in the literature. A concise list of parameters based on the state model as well as other annotations has been compiled and discussed in [60], see Table 9.2.

Based on the insights from the existing measures in the literature, Schoenenberg [57, 60] developed four additional measures:

1. Probability of multi-talk *Pmulti*

Pmulti is the sum of all probabilities representing multiple speaker states in the state model. In the data of Schoenenberg, for instance, this is computed from the test-conversation recordings as the number of speech samples with double-talk and triple-talk divided by the total number of samples.

2. *Divergence*

This measure is computed from sequences of states, which Schoenenberg refers to as *State Walks*, during a conversation. Each person can perceive four different types of state walks: Break (single-talk of one speaker → silence → single-talk of same speaker), Alternating Silence (single-talk of one speaker → silence → single-talk of another speaker), Non-successful Interruption (single-talk of one speaker → multi-talk → single-talk of same speaker), Successful Interruption (single-talk of one speaker → multi-talk → single-talk of another speaker).

As explained in more detail in [57, 60], the perceived state walks can be different for each person if delay is present, that is, each person can perceive a different reality of the same call. This is illustrated by the example shown in Fig. 9.13.

Then the *Divergence* measure is computed by first counting the number of matching state walks (i.e., state walks that are the same in the realities of every interlocutor) and mismatching state walks (state walks that differ between the

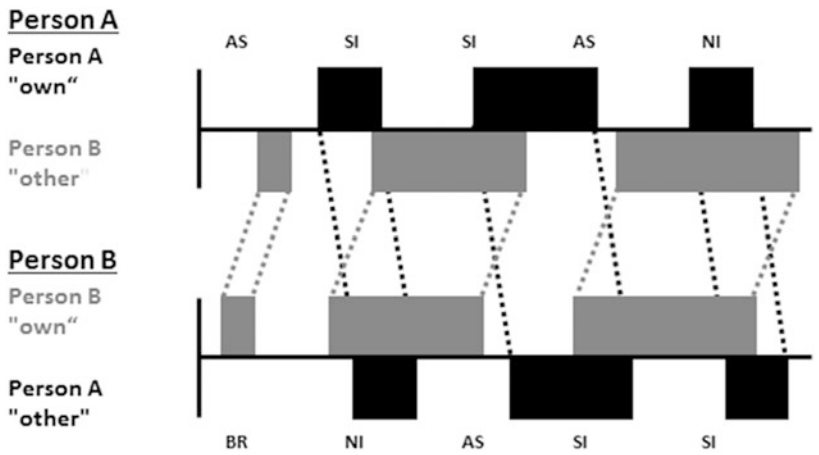


Fig. 9.13 Hypothetic example of a conversation part with high degree of divergence between the two persons' realities resulting from transmission delay; BR: Break, AS: Alternating Silence, NI: Non-Successful Interruption, SI: Successful Interruption, from [60], modified from [57]

realities of every interlocutor) and by second dividing the number of matches by the number of mismatches.

3. Speaker Alternation Rate Corrected *SARc*

The original measure, Speaker Alternation Rate *SAR* [21], is computed as the number of speaker changes per minute. This measure is directly decreasing with an increasing delay due to longer mutual silence periods when speakers wait for the delayed response of the others. This effect, however, does not necessarily mean a change of the participants' behavior in terms of the "true" density of speaker changes. For that reason, a modified measure, the Speaker Alternation Rate Corrected *SARc*, is computed as the number of speaker alternations divided by the overall time minus the waiting time caused by delay in case of alternating silences:

$$SARc = \frac{Number\ of\ Speaker\ Changes}{Duration - Correction\ Term}$$

$$SARc = \frac{\#\_AS\_A\_to\_B + \#\_AS\_B\_to\_A + \#\_SI\_A\_to\_B + \#\_SI\_B\_to\_A}{\text{Duration } - 2 \times \text{Ta } \times \#\_AS\_A\_to\_B}$$

with #: number of, AS: Alternating Silences, SI: Successful Interruptions, Ta: one-way delay time, A: person A, B: person B

4. Utterance Rhythm Change ΔURY

The Utterance Rhythm *URY* is the mean time from the beginning of one own speaker turn to the beginning of the next own speaker turn. To account for individual preferences of speakers and different utterance rhythms for different conversation scenarios, this measure is extended to the Utterance Rhythm Change ΔURY, by subtracting from the measured URY for a person in a certain conversation scenario with a transmission delay on the line the measured URY for the same person in the same conversation scenario without delay on the line.

9.6.2 Example Results

To show the impact of delay on a conversation, and thus on the conversation surface structure measures, Table 9.3 summarizes the behavior of the four above-mentioned measures for the eight delay tests from [60], see also Sect. 9.4.3.

Apparently, the three measures P_{Multi}, *SARc*, and *Divergence* showed significant differences for the tested delays in all data sets. That means, those measures are suitable to characterize the impact of delay. The measure ΔURY seems to be suitable as well, however, it failed to differentiate between delay conditions in two data sets.

Furthermore, the expected task dependency of the conversations is clearly reflected in all four measures, which means that these measures can be used to assess also the impact of the conversation tasks.

The measures show a more mixed picture concerning the effectiveness to reveal the impact of group size on the conversation. P_{Multi} and *SARc* enable to differentiate the conversation surface structure of two-party and three-party calls for all three available data sets, and *Divergence* allows this for two data sets. Only ΔURY does not appear to reflect the impact of group size.

As this analysis suggests, the conversation surface structure—and with it the four parameters—is not only influenced by the transmission delay, but also by the conversation task and group size. For that reason, Schoenenberg et al. also looked into the behavior of those four measures for individual cases in more detail; a short summary is given here, for the original results and detailed discussion see [60].

Pmulti:

In two-party conversation scenarios, Pmulti increases up to a certain delay (order of 400–800 ms) and then it saturates with further increasing delay. In three-party conversation scenarios, Pmulti shows a slight increase. In the scenarios including video, Pmulti is rather constant. In the RNT tasks, Pmulti is rather constant or it slightly decreases with increasing delay, depending on the audio bandwidth.

SARc:

In the two-party and three-party conversation scenarios, SARc is rather constant with the exception of two experiments (A.WB.2, A.NB.3), for which SARc increases with increasing delay. In the video scenarios, SARc shows a very slight

Table 9.3 Summary of MANOVA results of [60], showing the main effects of delay, task type, and group size on four measures extracted from conversation surface structure analysis. The numbers represent the p-values, and significant differences ($p < 0.05$) are emphasized

Data Set		A. NB.2a	A.NB	A.WB	A.FB	V. WB.2
Delay Test		A. NB.2a	A.NB.2b & A.NB.3	A.WB.2 & A.WB.3	A.FB.2 & A.FB.3	V. WB.2
P_{Multi}	Delay	***0.000***	***0.000***	***0.000***	***0.000***	***0.041***
	Task type	***0.000***	***0.000***	***0.000***		
	Group size		***0.000***	***0.000***	***0.000***	
SARc	Delay	***0.000***	***0.000***	***0.000***	***0.003***	***0.001***
	Task type	***0.000***	***0.000***	***0.000***		
	Group size		***0.000***	***0.000***	***0.000***	
ΔURY	Delay	***0.000***	***0.000***	***0.000***	0.635	0.319
	Task type	***0.000***	***0.000***	***0.000***		
	Group size		0.240	0.056	0.177	
Divergence	Delay	***0.006***	***0.000***	***0.000***	***0.000***	***0.000***
	Task type	***0.000***	***0.046***	***0.000***		
	Group size		***0.002***	***0.000***	0.659	

Note To test the impact of task type and group size, the eight individual tests were combined into five data sets, according to the rows Data Set and Delay Test

increase. In the RNT tasks, SARc is clearly decreasing with increasing delay in all cases, except for experiment A.NB.3.

ΔURY:
In the two-party conversation and the video scenarios, *ΔURY* increases only slightly. In the three-party conversation scenarios, *ΔURY* does not show a clear behavior. In the RNT scenarios, it clearly increases with increasing delay.

Divergence:
Divergence shows the most complex behavior. In four cases (2SCT NB, 3SCT NB, 2RNT WB, 3RNT WB) it increases up to a certain delay (order of 400–800 ms) and then it saturates with further increasing delay; in four cases (2SCT WB, 2SCT FB, 3CT FB, CNG) it shows a clear and in one more case (3SCT WB) a slight increase; in two cases (2RNT NB, 3RNT NB) it is rather constant.

From these analyses, it becomes clear that delays of around 400 ms or higher may not have a significant impact on speech quality ratings, however, have a significant impact on the conversational structure and behavior of interlocutors both

in two-party and multiparty conversations, and with and without the use of video. As a consequence, future research should address the indirect implications of such disruptions of conversational flow and behavior. While speech quality may be considered as high, participants may attribute the delay to the conversation partner (s) (see for example [3, 59]), with similar consequences for the usage of a given system: If conversations turn out to be troublesome with certain persons via certain telecommunication links (possibly always the same), the latter may not be used by a given user any longer, or less frequently. As a consequence, besides the analysis of opinions by subjective rating or using opinion models such as the E-Model ([25], see Sect. 9.7), also the analysis of conversation structure should be undertaken by operators or other service providers when effects of delay are to be analyzed.

9.7 Modeling the Quality Impact of Delay

While testing the impact of delay by means of conversation tests is the optimum way to assess delay on a communication link, it is not always possible to conduct such tests, either for technical reasons or resource limitations. For that reason, quality prediction models, which can be used for planning or monitoring purposes, are continuously being developed in the field. One well-established model, the E-Model, is presented in this section and currently investigated extensions are discussed as well.

9.7.1 General Modeling Approach

The E-Model [25, 45, 47] is a computational model useful in transmission planning to ensure that users will be satisfied with end-to-end transmission performance. With transmission delay as one parameter, this model allows designers to characterize speech quality under the impact of the delay.

The basic approach of the E-Model is to describe, on the so-called transmission rating scale R, the impact of different quality degradations as impairment factors Ix, with the assumption that the total quality degradation can be calculated as the summation of individual impairment factors.

In terms of the R-scale, this is expressed as

$$R = R_0 - Is - Id - Ie, eff + A.$$

Here, R_0 represents the *basic quality* resulting from the given speech and noise levels of the connection (line and room noises). The Simultaneous Impairment Factor Is comprises all degradations that occur simultaneously with the speech signal, such as suboptimal speech levels (too high, too low) and signal-correlated noise. The Effective Equipment Impairment Factor Ie, eff accounts for the effects

due to speech coding, IP packet loss and the decoder-side packet loss concealment (PLC). The Advantage Factor A addresses the quality-bonus that users give to certain connections, for example when they are using a mobile phone and are aware that speech quality may be lower than with fixed line connections.

For the topic of delay, the so-called *delayed impairment factor*, *Id*, is the most relevant one. It covers all quality degradations due to delayed signals, which are transmission delay (*Idd*), talker echo (*Idte*), and listener echo (*Idle*):

$$Id = Idd + Idte + Idle$$

The pure transmission delay part *Idd* is a function of the absolute end-to-end one-way delay *Ta* and is computed as follows:

$$\begin{aligned} &\text{If } Ta \leq mT \quad Idd = 0 \\ &\text{If } Ta > mT \quad Idd = 25 \cdot \left(\left(1 + x^{6 \cdot sT}\right)^{\frac{1}{6 \cdot sT}} - 3\left(1 + \left(\tfrac{x}{3}\right)^{6 \cdot sT}\right)^{\frac{1}{6 \cdot sT}} + 2 \right) \\ &\qquad\qquad \text{with } x = \frac{log_{10}(Ta/mT)}{log_{10}(2)} \end{aligned}$$

In addition to the one-way delay *Ta*, this equation builds on two more parameters that allow to take contextual information—more precisely, the interactivity of the conversation and the sensitivity of the user to the delay effect—into account:

mT, minimum perceivable delay,
sT, delay sensitivity

In earlier versions of the E-Model, those two parameters were not explicitly considered: *mT* was a fixed value of 100 ms, and *sT* was a fixed value of 1. The introduction of *mT* and *sT* was proposed in [49] and the ITU-T has incorporated these into the latest version of the E-Model in June 2015. Furthermore, predefined values for *sT* and *mT* are provided in [25] for different recommended application scenarios of the E-Model (see Table 9.4). If it is uncertain what user group or what application scenario is being addressed with the planned service, it is recommended in [25] to use the default class.

To compute the quality resulting for a certain transmission delay *Ta* without any other degradation, *Idd* has to be computed and combined with the baseline quality (R_0) and the other impairment factors (*Idte*, *Idle*, *Is*, *Ie, eff*, *A*) that are applicable for the current connection. If required, the *R*-values can be transformed into the commonly used Mean Opinion Scores by a transformation given in [25]. The *opinion model* E-Model allows to calculate predictions of opinion ratings also for combinations of degradations, for which different parameters describing the speech transmission link are converted to the respective impairment factors. It is noted here that only the impact on user opinion is addressed by the model, not the impact on the conversation behavior or effects such as divergence or misattribution of delay to the interlocutor(s), as it was discussed in Sect. 9.6.

Table 9.4 Definition of E-Model parameters sT and mT, which take contextual information for different application scenarios into account (slightly adapted from ITU-T G.107)

Definition of E-Model application scenarios		E-Model parameters	
Class of delay sensitivity	Use case	Delay sensitivity *sT*	Minimum perceivable delay *mT*
Default	Applicable to all types of telephone conversations Must be used for: – Carrier-grade fixed or mobile telephony – Enterprise-grade fixed or mobile telephony – When targeted user group and delay requirements are unknown	1	100
Low	Applicable only in cases where it is known that users have low sensitivity to delay, e.g., non-time-sensitive conversation scenarios	0.55	120
Very low	Applicable only in cases where it is known that users have very low sensitivity to delay, e.g., in primarily noninteractive cases, such as mainly listening to a conversation or to a lecture	0.4	150

9.7.2 Link to Conversational Surface Structure Analysis

Although the two new parameters *mT* and *sT* allow to include the effect of the contextual factor *interactivity/conversation type* on speech quality ratings, determining actual values is not trivial. While the recommendation text [25] provides values for different planning scenarios, see Table 9.4, [49, 60] have shown that the actual parameter values vary for different test scenarios.

This means, *mT* and *sT* can, in principle, be computed by fitting E-Model predictions with quality judgments observed from user tests. However, as the effect of delay is not always represented in perceptual quality ratings, since users may not notice the delay or may not attribute it to the system (see Sect. 9.4 and comments above), an alternative is to consider the impact of delay on the communication as such. In that line of thought, [49, 60] showed the possibility to compute *mT* and *sT* from the measures of conversation surface structure analysis and to achieve accurate results. Raake [49] used the parameter *SARc*; [60] used the parameter $\Delta URY_{gradient}$, which is the gradient of ΔURY along different delay values *Ta*.

As an example from [60], Fig. 9.14 shows a linear and a quadratic fitting between *sT* and $\Delta URY_{gradient}$; the fitting accuracy is computed as the norm of the residual with values of 1.55 for the quadratic and 1.62 for the linear fitting.

Figure 9.15 depicts the predictions obtained for the Delayed Impairment Factor *Idd* using the models of *sT* and *mT* from [49] using *SARc*. It can be seen that the impairment of speech quality ratings can quite well be predicted when the delay T and SARc are monitored during a connection. As discussed earlier, the effect of delay on the conversations as such may need to be considered in addition.

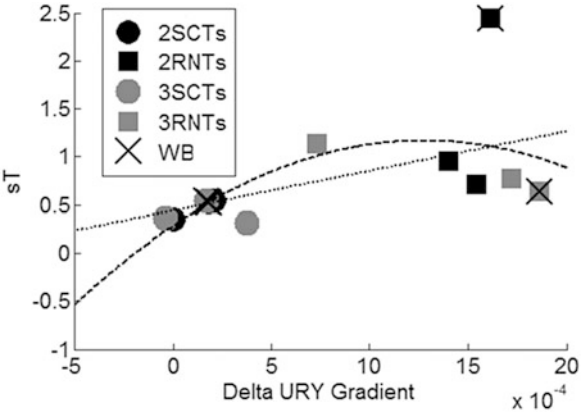


Fig. 9.14 Estimation of E-Model parameter sT based on conversation surface structure parameter ΔURYgradient, showing two possible fitting curves, linear and quadratic, from [60]

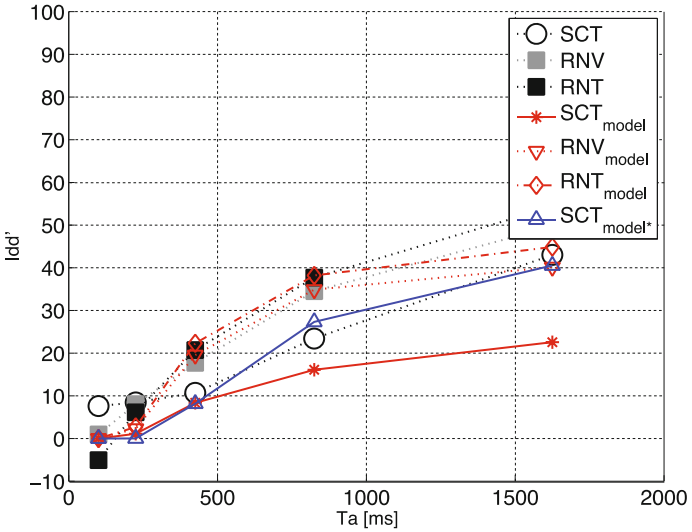


Fig. 9.15 Predictions of delayed impairment factor obtained when mapping SARc-values to sT and mT values, as a function of delay Ta. The prediction of speech quality including conversation surface structure parameters is a topic of ongoing research

9.7.3 Discussion

The E-Model has been specifically developed as a network planning tool, which means the model's quality predictions are "on the safe side". Regarding delay this means that with the default settings, the E-Model assumes a high delay sensitivity of the users.

With the two additional parameters, *mT* and *sT*, it is now possible to modify the E-Model to characterize the impact of delay on speech quality ratings in different conversational contexts. While both parameters are suitable in the fixed planning scenarios of [25], see Table 9.4, Schoenenberg [60] found out that, numerically, *mT* can lead to counterintuitive results if it is used as a free modeling parameter. For adapting the E-Model to new test contexts, these findings suggest to choose *mT* as a fixed value, e.g., according to [25], and then to use *sT* as the free modeling parameter.

Finally, the link to the conversation surface structure parameters is a further extension that allows to determine *mT* and *sT* from conversation recordings.

9.8 Conclusions

This chapter has given an overview of the challenges in evaluating media synchronicity in real-time communications and recommended appropriate methods for testing these systems.

It was shown how an in-depth analysis of such tests can be performed to gain a deep understanding of the effects of delay.

The results presented here point out the importance of choosing the right task for the test subjects to undertake. Overall, from the different types of conversation test scenarios, some form of free conversation seems to represent a good compromise for evaluating the impact of delay, since they allow test subjects to behave similarly to their everyday conversation situations. In cases where delay thresholds shall be identified, more sensitive but less ecologically valid tasks such as random number verification may be more suitable. Alternatives such as the short conversation test scenarios (SCT) described in [28] are much less delay sensitive owing to being scripted and responses need some consideration.

The impact of poor audio–video synchronization is most severe when the audio precedes the video. This reflects our everyday experience of light traveling faster than sound.

It is significant that subjects are often unaware of the presence of delay, even if they do experience problems communicating. This means that we cannot solely rely on subjective responses about aspects of speech quality to understand the effect of delay. This has given rise to the search for better testing methods such as tests which focus on understanding or measuring how subjects perform when carrying out tasks.

We have used an understanding of human interaction from the linguistic discipline of Conversation Analysis to show why latency can have a damaging effect on conversations that may not be attributed to the line, but to the conversation partner or conversation as such. The related method of Conversation Surface Structure Analysis provides a technique for characterizing a conversation in terms of a state model based on the speech activity of the participants.

We also looked at music interaction over networks. There are a number of different applications to be considered here, including music performance, rehearsal, and tuition. The degree to which these activities are affected by latency is strongly dependent on the tempo of the music; very slow passages of music or music with no rhythm can be quite delay tolerant, but as tempo increases the delays become a significant proportion of the rhythmic period and playing in time with each other becomes impossible. In practice latencies of 20–30 ms are required for music with medium tempi.

Ongoing development of codecs will lead to increased efficiency to be balanced with the resulting latency, and new developments in network transmission will help to reduce delays. However, there will always be some residual delay in multimedia communication. The question from an end user's perspective is whether this delay has an adverse impact on the communication, or is perceived to be annoying. This chapter has provided an overview of insights from related studies, and guidance on how to design subjective evaluations with appropriate tasks and evaluation questions.

Definitions

Telemeeting [ITU-T P.1301] A meeting in which participants are located at least two locations and the communication takes place via a telecommunication system. The term telemeeting is used to emphasize that a meeting is often more flexible and interactive than a conventional business teleconference and could also be a private meeting. The telemeeting could be audio-only, audiovisual, text-based, or a mix of these modes.

Conversational quality [ITU-T P.1301] The perceived quality when two or more test participants have a conversation.

Conversation analysis (CA) [Sacks] The study of social interaction, in particular focusing on conversations.

Turn construction unit (TCU) [Sacks] A conversation analysis term describing the fundamental segment of speech in a conversation – essentially a piece of speech that constitutes an entire 'turn'.

Transition relevance place (TRP) [Sacks] A conversation analysis term which indicates where a turn or floor exchange can take place between speakers.

References

1. Barger, R., Church, S., Fukuda, A., Grunke, J., Keislar, D., Moses, B., Novak, B., Pennycook, B., Settel, Z., Strawn, J., Wiser, P., Woszczyk, W.: AES white paper: networking audio and music using internet2 and next generation internet capabilities. Audio Engineering Society, New York (1998)
2. Bech, S., Zacharov, N.: Perceptual Audio Evaluation-Theory, Method and Application. Wiley
3. Berndtsson, G., Folkesson, M., Kulyk, V.: Subjective quality assessment of video conferences and telemeetings. Packet video workshop 2012, München, Germany (2012)
4. Berry, A.: Spanish and American Turn-Taking Styles: a Comparative Study (1994)
5. Biech, E.: The Pfeiffer book of successful team-building tools: best of the annuals. Pfeiffer (2007)
6. Brady, P.T.: A technique for investigating on-off patterns of speech. Bell Syst. Tech. J. **44**(1), 1–22 (1965)
7. Brady, P.T.: A statistical analysis of on-off patterns in 16 conversations. Bell Syst. Tech. J. **47** (1), 73–99 (1968)
8. Brady, P.T.: Effects of transmission delay on conversational behavior on echo-free telephone circuits. Bell Syst. Tech. J. **50**(1), 115–134 (1971)
9. Braun, A.M.: Qualitätsaspekte multimodaler Kommunikation: Subjektive und objektive Messungen [engl.: Qualityaspects of multimodal communication: Subjective and objective measurements]. Ph.D. thesis, Eidgenössische Technische Hochschule Zürich (2003)
10. Chafe, C., Wilson, S., Leistikow, R. Chisholm, D., Scavone, G.: A simplified approach to high quality music and sound over IP. In: Proceedings of the International Conference on Digital Audio Effects, Verona, Italy (2000)
11. Chafe, C., Gurevich, M.: Network Time Delay and Ensemble Accuracy: Effects of Latency, Asymmetry. J. Audio Eng. Soc. (2004)
12. Clift, R.: Conversation Analysis, Cambridge Textbooks in Linguistics (2016)
13. Cragan JF, Wright DW (1991) Communication in Small Group Discussions: An Integrated Approach. West Publishing Company
14. Daly-Jones, O., Monk, A., Watts, L.: Some advantages of video conferencing over high-quality audio conferencing: fluency and awareness of attentional focus. Int. J. Hum Comput Stud. **49**, 21–58 (1998)
15. Duffy, S.: Closer—A study of one-to-one instrumental music tuition through video conference. Media and Arts Technology Programme Project Report. Queen Mary, University of London (2011)
16. Egger, S., Schatz, R., Scherer, S.: It takes two to tango—assessing the impact of delay on conversational interactivity on perceived speech quality. In: Annual Conference of the Speech Communication Association, pp. 1321–1324 (2010)
17. Egger, S., Schatz, R., Schoenenberg, K., Raake, A., Kubin, G.: Same but different?—Using speech signal features for comparing conversational VoIP quality studies. In: International Conference on Communications, IEEE, pp. 1320–1324 (2012)
18. Farner, S., Solvang, A., Sæbø, A., Svensson, U.P.: Ensemble hand-clapping experiments under the influence of delay and various acoustic environments. J. Audio Eng. Soc. **57**, 1028–1041 (2009)
19. Geelhoed, E., Parker, A., Williams, D.J., Groen, M.: Effects of latency on telepresence. Technical report, HPL-2009-120, HP Laboratories (2009)
20. Guéguin, M., Bouquin-Jeans, R.L., Gautier-Turbin, V., Faucon, G., Barriac, V.: On the evaluation of the conversational speech quality in telecommunications. EURASIP J. Adv. Signal Process. (2008)
21. Hammer, F., Reichl, P., Raake, A.: Elements of interactivity in telephone conversations. In: International Conference Spoken Language, pp. 1741–1744 (2004)

22. Hammer, F.: Quality Aspects of Packet-Based Interactive Speech Communication, Ph.D. thesis, Technical University Graz, Vienna (2006)
23. Huynh-Thu, Q., Garcia, M-N., Speranza, F., Corriveau, P., Raake, A.: Study of rating scales for subjective quality assessment of high-definition video. IEEE Trans. Broadcast. **57**, 1–14 (2011). https://doi.org/10.1109/tbc.2010.2086750
24. ITU-T: Handbook on Telephonometry. International Telecommunication Union, Geneva (1993)
25. ITU-T Recommendation G.107: The E-model: a computational model for use in transmission planning. International Telecommunication Union, Geneva (2015)
26. ITU-T Recommendation G.114: One-way transmission time. International Telecommunication Union, Geneva (2003)
27. ITU-T Recommendation P.800: Methods for subjective determination of transmission quality. International Telecommunication Union, Geneva (1996)
28. ITU-T Recommendation P.805: Subjective evaluation of conversational quality. International Telecommunication Union, Geneva (2007)
29. ITU-T Recommendation P.831: Subjective performance evaluation of network echo cancellers. International Telecommunication Union, Geneva (1998)
30. ITU-T Recommendation P.832: Subjective performance evaluation of hands-free terminals. International Telecommunication Union, Geneva (2000)
31. ITU-T Recommendation P.920: Interactive test methods for audiovisual communications. International Telecommunication Union, Geneva (2000)
32. ITU-T Recommendation P.1301: Subjective quality evaluation of audio and audiovisual multiparty telemeetings. International Telecommunication Union, Geneva (2012)
33. ITU-T Recommendation P.1305: Effects of delays on the telemeeting quality. International Telecommunication Union, Geneva (2016)
34. ITU-T Recommendation P.1312: Method for the measurement of the communication effectiveness of multiparty telemeetings using task performance. International Telecommunication Union, Geneva (2015)
35. ITU-R Recommendation BT.1359-1: Relative timing of sound and vision for broadcasting. The ITU Radiocommunication Assembly, Geneva (1998)
36. International Telecommunications Union. https://www.itu.int/en/ITU-T/studygroups/2017-2020/12/Pages/default.aspx
37. Jekosch, U.: Voice and Speech Quality Perception—Assessment and Evaluation. Springer (2005)
38. Kapur, A., Wang, G., Davidson, P., Cook, P.R.: Interactive network performance: a dream worth dreaming? In: Organised Sound, vol. 10 (3), pp. 209–219. Cambridge University Press (2005)
39. Kitawaki, N., Itoh, K.: Pure delay effects on speech quality in telecommunications. IEEE J. Sel. Areas Commun. **9**(4), 586–593 (1991)
40. Knapp, M.L., Hall, J.A.: Nonverbal Communication in Human Interaction, 7th edn. Wadsworth, Cengage Learning, Boston, USA (2010)
41. Krauss, R.M., Bricker, P.E.: Effects of transmission delay and access delay on the efficiency of verbal communication. J. Acoust. Soc. Am. **41**(2), 286–292 (1967)
42. Krauss, R.M., Garlock, C., Bricker, P.D., McMahon, L.: The role of audible and visible back-channel responses in interpersonal communication. Journal of Personality and Social Psychology **35**(7), 523–529 (1977)
43. Le Callet, P., Perkis, A., Möller, S. (eds.): Qualinet White Paper on Definitions of Quality of Experience (2012). European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003). Version 1.2." Mar-2013
44. Matejka, J., Glueck, M., Grossman, T., Fitzmaurice, G.: The effect of visual appearance on the performance of continuous sliders and visual analogue scales. In: Proceedings of the 2016

CHI Conference on Human Factors in Computing Systems. ACM, New York, NY, USA, pp. 5421–5432 (2016)
45. Möller, S.: Assessment and Prediction of Speech Quality in Telecommunications. Springer (2000)
46. O'Conaill, B., Whittaker, S., Wilbur, S.: Conversations over videoconferences: an evaluation of the spoken aspects of video mediated interaction. Hum. Comput. Interact. **8**(4), 389–428 (1993). Olson JS, Olson GM, Meader
47. Raake, A.: Speech Quality of VOIP—Assessment and Prediction. Wiley (2006)
48. Raake, A., Hoeldtke, K., Schlegel, C., Ahrens, J., Geier, M.: Listening and conversational quality of spatial audio conferencing. In: International Conference of the Audio Engineering Society, pp. 4–7 (2010)
49. Raake, A., Schoenenberg, K., Skowronek, J., Egger, S.: Predicting speech quality based on interactivity and delay. In: Proceedings of 14th Annual Conference of the International Speech Communication Association (Interspeech 2013), Lyon, France (2013)
50. Richards, D.L.: Conversational performance of speech links subject to long propagation times. In: International Conference on Satellite Communication, IEEE, pp. 955–963 (1962)
51. Riesz, R.R., Klemmer, E.T.: Subjective evaluation of delay and echo suppressors in telephone communications. Bell Syst. Tech. J. **42**(6), 2919–2941 (1963)
52. Ruhleder, K., Jordan, B.: Co-constructing non-mutual realities: delay-generated trouble in distributed interaction. Comput. Support. Coop. Work **10**(1), 113–138 (2001)
53. Sacks, H., Schegloff, E.A., Jefferson, G.: A simplest systematics for the organisation of turn-taking in conversation. Language 50, pp. 696–735 (1974)
54. Sat, B., Huang, Z., Wah, B.W.: The design of a multi-party VoIP conferencing system over the internet. In: International Symposium on Multimedia, IEEE, pp. 3–10 (2007)
55. Schmitt, M., Gunkel, S., Cesar, P., Hughes, P.: A QoE testbed for socially-aware video-mediated group communication. In: Proceedings of the 2nd International Workshop on Socially-aware Multimedia, New York, NY, USA, pp. 37–42 (2013)
56. Schmitt, M., Gunkel, S., Cesar, P., Bulterman, D.: The influence of interactivity patterns on the Quality of Experience in multi-party video-mediated conversations under symmetric delay conditions. In: Proceedings of the 3rd International Workshop on Socially-aware Multimedia, New York, NY, USA (2014)
57. Schoenenberg, K., Raake, A., Egger, S., Schatz, R.: On interaction behaviour in telephone conversations under transmission delay. Speech Commun. **63–64**, 1–14 (2014)
58. Schoenenberg, K., Schmieder, M.: 3rnt—3-party random number verification (timed version) task (2014). https://doi.org/10.5281/zenodo.16133
59. Schoenenberg, K., Raake, A., Koeppe, J.: Why are you so slow?—Misattribution of transmission delay to attributes of the conversation partner at the far-end. Int. J. Hum.-Comput. Stud. **72**(5), 477–487, May 2014
60. Schoenenberg, K.: The Quality of Mediated-Conversations under Transmission Delay, Ph.D. Thesis, Technical University of Berlin (2016)
61. Skowronek, J., Schiffner, F., Schoenenberg, K., Raake, A.: 3sct 3-party short conversation test scenarios for conferencing assessment (version 03) (2014). https://doi.org/10.5281/zenodo.16136
62. Skowronek, J.: Quality of Experience of Multiparty Conferencing and Telemeeting Systems Methods and Models for Assessment and Prediction, Ph.D. thesis (2017). https://dx.doi.org/10.14279/depositonce-5811
63. https://www.eurescom.eu/services/management-of-european-rd-projects/past-projects/ta2.html
64. Tam, J., Carter, E., Kiesler, S., Hodgins, J.: Video increases the perception of naturalness during remote interactions with latency. In: Proceedings of CHI'12, New York, NY, USA, pp. 2045–2050 (2012)
65. Vartabedian, A.G.: The effects of transmission delay in four-wire teleconferencing. Bell Syst. Tech. J. **45**(10), 1673–1688 (1966)

66. Wah, B.W., Sat, B.: The design of VoIP systems with high perceptual conversational quality. J. Multimed. **4**(2), 49–62 (2009)
67. Wang, J., Yang, F., Xie, Z., Wan, S.: Evaluation on perceptual audiovisual delay using average talkspurts and delay. In: 2010 3rd International Congress on Image and Signal Processing (CISP), vol. 1, pp. 125–128 (2010)

Chapter 10
Synchronization for Secondary Screens and Social TV: User Experience Aspects

Jeroen Vanattenhoven and David Geerts

Abstract This chapter provides an in-depth discussion on the impact of synchronization of TV-related applications on the user experience. After all, applications meant to be used in conjunction with TV watching are created for the benefit and pleasure of the viewer. In order to explain the main user-related aspects of media synchronization, we will first sketch how the television and media landscape has evolved and which timing and synchronization aspects are important for the makers of TV-related applications. Then, we will delve into the core topic of this chapter by presenting the main user experience aspects involved in the creation of second-screen (Attentive readers will notice that 'second screen' is sometimes written with, and sometimes without a hyphen. When using 'second screen' without a hyphen, we use it as a substantive (the second screen). When using 'second-screen' with a hyphen, it is used as an adjective (second-screen applications).) applications—a specific type of TV-related application that has gained a lot of attention in the past years. Finally, we highlight the impact of synchronization issues on the user experience for Social TV applications. Our insights are gathered from our earlier publications in this area, results of research with users from the European research project TV-RING, and related literature. The chapter will, therefore, serve as an overview of media synchronization from the perspective of the viewer, based on our own insights and experiences, and complemented with the current state of the art.

Keywords Social TV · Synchronization · Second screen · User experience

J. Vanattenhoven (✉)
KU Leuven, Leuven, Belgium
e-mail: jeroen.vanattenhoven@kuleuven.be

D. Geerts
Mintlab, KU Leuven/imec, Leuven, Belgium
e-mail: david.geerts@kuleuven.be

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_10

10.1 Introduction

Time has always played a crucial role in the world of television. Most programming is scheduled at a specific time, and broadcasters assume specific groups of viewers to sit in front of their TVs at that time [1]. However, a number of relatively recent phenomena in the media landscape, such as digital recorders or tablets being used as a 'second screen', have created a more complex relationship between television and time, one where questions related to synchronization between different types of content emerge and need to be dealt with in the design of new applications.

A first critical change has been the ability to time-shift viewing experiences. Technically, we have been able to do so since the arrival of the video recorder or VCR. With the arrival of new technologies, however, time-shifting has been made possible on a larger scale by on-demand services, digital recording devices, and illegal downloads [34]. Many researchers have uncovered reasons why time-shifting is so popular and they all boil down to a transfer of control from the broadcaster to the consumer. People use time-shifting to skip advertising, catch-up on TV programs that they have missed, and to view programs when it suits them better in their daily life [4, 13, 25, 34]. The most comprehensive overview so far is offered by Abreu et al. [1], who constructed a detailed taxonomy to chart the existing ways in which people can watch TV nowadays as well as functionalities novel TV viewing applications offer, including various time-shift behaviours such as such as 'Pause TV', 'Start-over TV' or 'Catch-up TV'. Since time-shifting does not pose any synchronization challenges for either the developer or the viewer, we will not cover this topic in this chapter.

A second important phenomenon has been the proliferation of mobile and portable devices. To be more precise, it is their use in front of the television that has been of great interest to researchers and broadcasters. People are using tablets, smartphones and laptops, while they are watching TV [14, 24, 31, 35]. In most cases, this so-called second-screen use is not related to what is happening on the TV. Instead, people check their emails, chat via social networks, perform work-related activities, etc., during the dull moments of the television program. When something important is about to happen, they turn their attention back to the TV set. Many broadcasters and app developers have seen this as an opportunity to target the second screen in order to get the viewer's attention back. The fact that advertising revenue is related to the attention it receives from the audience is an important motivation for this. An important challenge for second-screen applications is to obtain accurate synchronization between the second-screen devices and the TV, and the coordination of the program's activities between the broadcaster and the viewer. These matters will be discussed in detail in the first part of this chapter.

Finally, timing and its impact on the user experience is also of the essence of Social TV applications. Early Social TV applications focused on supporting shared viewing experiences for people located in different locations [9, 10, 15], whereas some of the second-screen applications described in the previous paragraph focus

on supporting collocated viewers [36]. By providing specific functionalities in an application, developers can allow viewers in different locations to communicate and experience 'presence' [30], which is essential to support shared experiences. It is in these circumstances that the lack of synchronization can hinder communication between parties, and consequently provide a negative experience for at least one party. The synchronization of the communication channel with the content in the different locations is, therefore, an important prerequisite for an optimal, social experience, and will be covered in the second part of this chapter.

10.1.1 Studies and Experiments

Before discussing both aspects of how users experience media synchronization, we will give an overview of the various studies and experiments that have formed the basis for most of the insights we share in this chapter. Most of the studies we describe in the next paragraphs had a broader goal than the one dealt with in this chapter, for which we selected only those results that directly relate to synchronization, and are published or publicly available for finding out more details.

The first series of studies have been carried out on several Social TV systems as part of our effort in creating sociability heuristics for Social TV. A detailed description of the methodology and results are covered in Geerts and De Grooff [19], here we will give a brief summary. We performed a series of competitive user tests between April 2006 and March 2008. In total 149 unique users, of which 66 were male (44%) and 83 female (56%), participated in these tests. Ages ranged from 14 to 76 years old. Each user test was performed in two connected rooms, of which one was a usability lab with a one-way mirror. The applications were operated with a remote control and/or remote keyboard and used on a normal television set. All user tests followed the same pattern, including an introduction to the system, the test itself, a (group) interview and a couple of questionnaires, but as all applications were different, e.g. in set of features or stage of development, the test setup, questionnaires and interview questions used differed slightly from test to test. However, we made sure that in each test the focus was on social interaction with these systems. The systems we tested were AmigoTV [12], Windows Messenger, Social Television [21], Communication Systems on Interactive Television (CoSe, see Fig. 10.1) and Ambulant Annotator [8]. The latter was the only one that included a second screen.

A second source for the insights reported in this chapter was an experiment we carried out to study the synchronization requirements for watching online videos together. Again, a detailed description of the methodology can be found in Geerts et al. [16]; here we give a short summary. We set up a within-subjects lab-based experiment with 18 couples (partners, friends or family), with a total of 36 people taking part in the tests, consisting of 12 males and 24 females. The age ranged from 15 to 68 years old. Each participant from each couple was shown two episodes of a

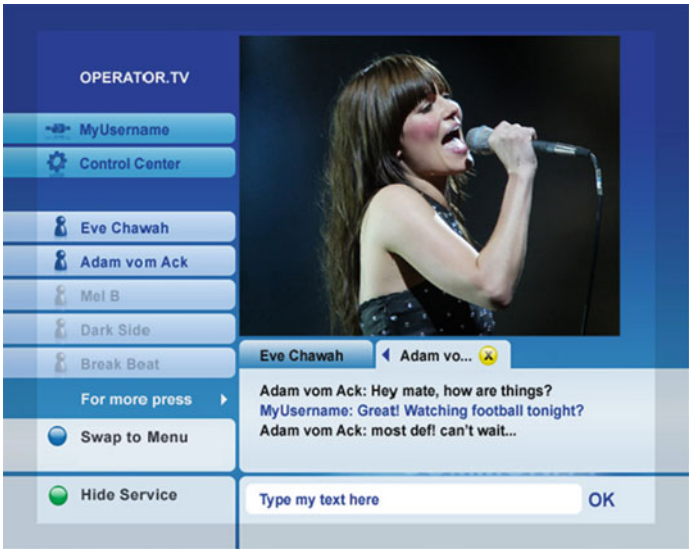


Fig. 10.1 Interface of the CoSe application (© Coeno Gmbh)

popular local quiz show at different locations. During the first episode, participants could voice chat with each other using a headset. The headset was also used for listening to the audio track of the video content, so the participants could not hear the audio track of their partner's show. During the second episode, participants could only text chat with each other. Without informing the participants, every 7 minutes the synchronization level of the videos was changed. In each condition (voice chat and text chat), five synchronization levels were presented to users: 0 s (perfect sync), 500 ms, 1, 2 and 4 s. These levels were presented in a randomized order for each set of participants and each condition. As a difference in synchronization between two participants implies that (the video of) one person is ahead ('leading'), and one person is behind ('lagging'), the order of who is leading and who is lagging was also randomly varied. After every 7 min (before the next synchronization change), the participants were asked to fill in a web-based questionnaire, asking a series of questions related to togetherness, noticeability and annoyance of the synchronization differences.

A third study has been extensively described in Geerts et al. [20]. To get a detailed picture of how people experience second screens in a social context, we recruited five couples (average age 33.6) and observed them in their own home environment while they were watching a television show and using a second-screen application. A camera was placed in their home before the day of the show (see Fig. 10.2), so no researcher had to be present. This was done to minimize the intrusion for the participants and to have them watch as naturally as possible. For the same reason, the participants were instructed to watch the show and use the second-screen application as they would normally do. One or two days after the

Fig. 10.2 Camera view from an observation of second-screen use

show the researcher returned to collect the footage and watch the recording together with the participant(s) using event-triggered retrospective think aloud. This means that when specific events took place in the recording, such as interaction with the second-screen device or social interaction between the participants, the researcher asked if the participants could clarify what they were doing and why. This in-depth interview lasted around two hours per couple. The show that was selected is a Belgian drama series about a young prosecutor called 'De Ridder'. The second-screen application features a timeline that shows new content related to the television show as the program progresses. This content can be quotes from the show, polls that users can respond to, information about specific terms used in the show, maps of the location of characters that the users can interact with, etc. All types of content can be 'liked' in the application or shared via Facebook or Twitter.

The fourth and final range of studies that served as the basis for this chapter were carried out as part of the European TV-RING project with the goal of developing a quiz/game companion application that mixes elements on the first and second screen to create a stimulating experience for the participating viewers. In this process, a co-creation session was held, as well as two iterations of prototype evaluation. These studies have not been published, but are available in a public deliverable of the project that can be requested from the authors. Each step in the process is explained in the next paragraphs.

To find out what type of play-along game users would create for themselves, we recruited nine groups of users and organized a co-creation session. There were three groups of two, four groups of three, and two groups of four participants. These were all groups of people who knew each other well and watched TV together at least occasionally. There were couples, groups of friends and families with an age range between 10 and 52 years old. Participants were instructed that they would view an episode of The Voice of Holland, a local version of a popular talent show format. Before they would start viewing the episode, they were asked to create a game that they could all participate in and that they could play while watching The Voice. To

help them create a game they were offered a selection of gaming attributes including items to facilitate timing, score keeping, representation and a randomizing element (e.g. dice). When they were done creating their game they were asked to explain it to the researcher. After the game was explained, the episode was started and the participants were instructed to simply watch the show and play their game. After about an hour of watching the show and playing their game, the participants were interviewed about their experience. Based on this co-creation session we created a set of requirements for second-screen companion apps for game shows.

In the second step, we created a paper prototype and tested this with potential end users. The goal of the paper prototype was to quickly test a few different variations of the different facets of an interactive quiz. We established four steps: set up (setting up the quiz, getting all participants registered); identification (letting all participants choose a way to represent themselves on the screen); questions (actually playing the quiz); and visualization (the way the scores would be displayed on the TV screen). Since one of the key features of the interactive quiz is stimulating social interaction, it was crucial to do the prototype session with groups who watch TV together regularly. Five groups of various compositions were recruited, in total 14 people. Ages ranged from 19 to 52 with 10 female and four male participants. We invited them to our lab that is set up like a living room. They were asked some questions upfront about their TV watching habits and then played through the different steps mentioned earlier. After each step participants were asked what they thought about the prototype variations and in the end to rank them according to their preference. There was a closing interview about the whole experience at the end.

In a final step, a hybrid paper/digital prototype was created, which used a fully interactive tablet application created in Axure and paper mock-ups for the visualizations on the TV. The key questions to be answered during the evaluation were: How would the users' attention be distributed over the first and second screen while there was an actual show running on the TV? And how would this impact the (social) interaction between the participants? Again, we recruited groups of people who watch TV together regularly. Five groups of various compositions were recruited, 12 people in total. Ages ranged from 21 to 60 with eight female and four male participants.

They were invited to our lab, set up the application and then were asked to play along with the episode as if they were playing at home. After the show ended, the participants were interviewed about the whole experience.

The design choices we made in these various user studies and experiments presented in this chapter were chosen in function of the technical possibilities, project constraints or best practices at that moment. There are certainly other ways of solving the issue of synchronization, which are not covered in this chapter. However, we think that through these various examples, this chapter offers sufficiently interesting insights and results that can inform other designers and developers that deal with synchronization.

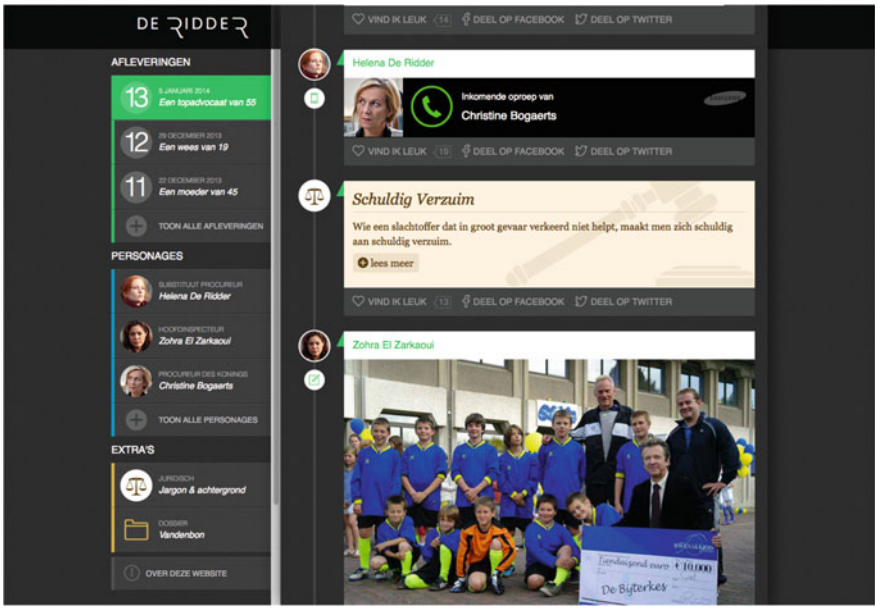


Fig. 10.3 Screenshot of the second-screen application for De Ridder

10.2 Synchronization in Second-Screen TV Applications

Second-screen applications are applications that offer content and interactivity related to television on a different device. These applications make use of the fact that many viewers are sitting in front of their TV with a smartphone, tablet or laptop. The TV is implicitly understood to be the first screen, or the most important screen, since the programs appear on this device.[1] Smartphones, tablets or laptops are explicitly called the second screen. In order to turn these second screens into an advantage and use them to augment the experience, dedicated 'companion' apps are created. In order to make this idea work, the first and the second screen need to know about each other. Moreover, they need to be synchronized. Figure 10.3 shows an example of a second-screen application augmenting the experience of watching a television drama titled 'De Ridder'. The main character of the show is a prosecuter. The second-screen application shows the actual text-messages characters in the drama receive; these messages cannot be seen by watching the TV. Furthermore, certain legal terms that are used are explained in the second-screen application when they are used on TV.

[1]Sometimes it is argued that in certain cases the tablet or smartphone is the first screen, and the TV is in the background as a second screen. While we agree with this perspective as a possible use of the term, our interest goes to those cases where TV is the first screen and other devices act as a second screen.

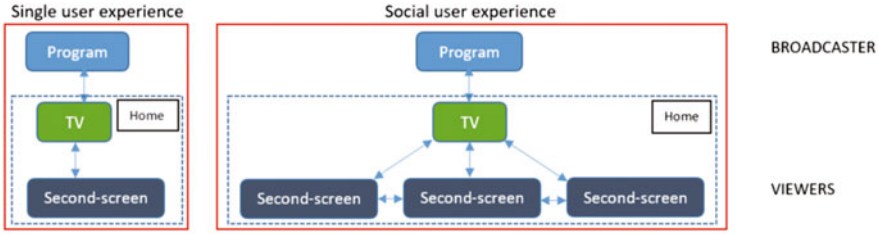


Fig. 10.4 Synchronization in second-screen scenarios

Figure 10.4 illustrates the place of synchronization in second-screen TV applications. On the left-hand side, a single user experience variant is shown. One example is the example we just gave of a companion app augmenting the user experience of a drama show. On the right-hand side, a social experience scheme is shown. An example of this scenario is a quiz program on TV that can be played on second-screen devices by several members in the home. The arrows indicate where, and in what direction synchronization needs to be addressed. In this section, we will discuss why this synchronization is important, how it impacts the user experience, and how it can be taken into account via the design of the application and the TV format.

10.2.1 Setting up the Synchronization Between the TV and the Second Screen

Any second-screen experience that requires synchronization starts with setting up the synchronization between the TV and the second screen. It is quite obvious to state that making this process user friendly is important. But how exactly can this be achieved? To start with, the procedure should consist of as few steps as possible. When users have to spend too much time at this stage of the experience, it is likely that they will lose interest, give up and do not make use of the second-screen possibilities. If no or little registration details are required for the functioning of the application, it is better not to ask the user to provide this information [20].

In the TV-RING project, using paper mock-ups and hybrid paper/digital mock-ups, we explored and evaluated three different options for setting up the synchronization between the TV and the second screen(s). In the first option, users would scan a QR code, which appeared on the TV, with their second-screen device. The second option was to enter a numerical code, which appeared on the TV, on their second-screen device. Finally, as a third option, we explored a 'Master' system, in which one user would scan a QR code from the TV, making his/her device the master device; the other users would then scan a QR code from the master device. Figure 10.5 shows an overview of the paper mock-ups for synchronizing the second screen with the TV at the start of the viewing experience.

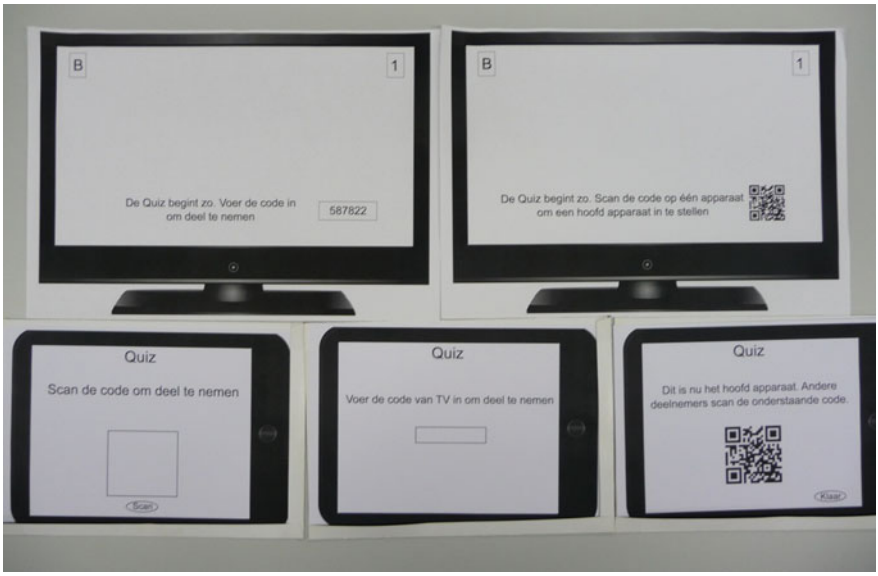


Fig. 10.5 Paper mock-ups for setting up synchronization with a second screen

The TV mock-ups are shown at the top: on the left with a numeric code and on the right with a QR code. The tablet screens are shown at the bottom. From left to right: scan the QR code, enter the numerical code from the TV, the QR code on the 'Master' tablet that should be scanned by the others.

We found that the option with the QR code was the fastest and the least error prone for participants with sufficient technical background. The latter is important to note because some participants were not familiar with QR codes and did not know how to proceed. During the evaluation with hybrid digital/paper prototypes, some participants with less technical knowledge even wanted to scan the QR code on the TV by pointing the screen of their tablet towards the TV. Consequently, when thinking about making a second-screen application for a larger audience, it is fairly realistic to assume that not everyone will have the same technological skills or literacy. The option with the numerical code worked for everyone, whereas the 'Master' system was perceived as overly complex.

10.2.2 Usability of Second-Screen Applications

It is important to note that the interfaces of the first and the second screen should be very easy to use, not only during the synchronization phase. When we consider TV programs that make use of a second-screen app to augment the experience, the fact that there is a second device and user interface automatically increases the cognitive

burden on the part of the user [2, 38]. One should avoid clutter on the interface and ensure that only the necessary elements are present. A study on the effects of second-screen use on comprehension of a news program, for example has shown that the cognitive load during the use of second screens is higher compared to during the use of just one screen [37]. The increased cognitive load leads to lower factual recall and comprehension of news content. One way to make second-screen designs more usable is to avoid too many instructions or text. We have found in our prototype evaluations that users do not read most of them anyway.

Another general usability guideline is consistency. In this case, the design of the user interface should be consistent on the TV as well as on the secondary device. A classic example is the use of the four colour buttons on a remote control, typically used in Teletext service and other interactive TV applications. These buttons are laid-out in a specific order on the remote control: red, green, yellow and blue. When a TV application would use these colour buttons to indicate the options that can be triggered by the buttons, a consistent design will feature all of them on the first screen in the same order as on the remote control. In the early user tests, the use of prototypes that excluded two of the colour buttons as there were only two options (yes and no) resulted in problems for colour-blind participants that did not know what colour buttons to use. With the increased complexity of digital interfaces on second-screen devices, designing consistent interfaces are even more of a concern.

10.2.3 Guiding Attention and Distraction to the Correct Screen

Once the user has signed on and is about to enjoy the second-screen experience, a delicate balance needs to be struck between offering sufficient opportunities for engagement and overburdening the viewer on the second screen causing him or her to be too much distracted from the program on TV. During our prototyping activities in the TV-RING project, we explored many interactive features to augment the second-screen TV viewing experience of a quiz format. The fact that there is an option to play along with a second screen makes viewers more motivated to pay attention. They become very excited during the program because they want to win the game. On the other hand, at times we introduced too many activities or game features on the second screen, causing participants to be overwhelmed. Consequently, they started paying less attention to the TV format and complained that this impacted their experience in a negative way. They were not engaged with the TV program anymore. Figure 10.6 shows two different philosophies of displaying information that requires input from the user. On the left, most information is shown on the tablet. On the right, the information is shown on the TV. Only the input is foreseen on the tablet. In the latter case, the attention of the viewers is mostly on the TV screen, which is felt by participants as providing a more social experience. In contrast, the first option causes every viewer to spend more time looking at their own tablet.

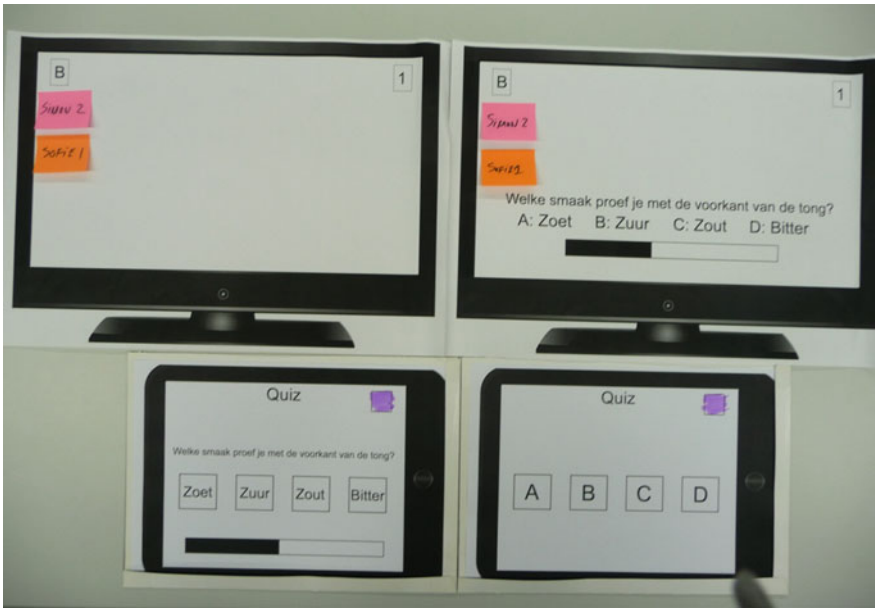


Fig. 10.6 Paper mock-ups with two approaches to providing input

An important way to avoid overburdening the viewer is to confine the second-screen interaction to a limited number of decisive moments during the show [3], and to limit the duration of this interaction. This design aspect was revealed during co-design sessions in which participants were asked to create their own second-screen application for an existing TV format. Most groups included a 30 s timer, so that the user, after performing an interaction on the second screen, would know that they could relax afterwards and just watch the program.

There are also broader issues to address. When input is expected from the user on the second-screen application, there should be sufficient time for him or her to grab the tablet from the coffee table, perhaps enter the pin code, start up the application again, wait for the necessary connections to be made, and then allowing the user to view the screen to understand what is required. In our evaluations in the actual pilot deployments of applications in the TV-RING project, a number of participants experienced difficulties because the questions appeared too early, or the timer ran out too fast. Through auditory or visual means (e.g. an announcement of the TV show presenter, a visual indicator on the first screen or an audio signal on the second screen), the application should, therefore, announce that the user will soon have the opportunity to use the second-screen application, giving him or her sufficient time to prepare. In addition, a timer could be displayed so that the user knows how much time he or she has left to enter the required input. This helps the user to be ready in time, and provides moments of calmness afterwards. Finally, the time synchronization between the first and the second screen has to be exact. Some

participants in the live pilot playing along in a quiz, reported that they could sometimes wait until the right answer was shown on TV, and still had the time on the second-screen device to change their answer. Obviously, this is something to avoid at all costs.

Even when both interfaces are very well designed, the user experience can still be hampered. Here, we are mainly talking about how attention diverts from one screen to the next. Human attention, even only taking into account classical television, is a very complex phenomenon [22]. Many researchers have expanded this line of research by focussing on the distribution of human attention on multiple screens, [2, 3, 6, 11, 23, 29, 33, 38]. Brown et al. [6] emphasized the importance of taking into account the relaxed nature of television viewing, and studies how and when attention shifts between screens in a natural setting using the eye-tracking equipment. They discovered that the TV received approximately five times more attention than the tablet. The length of the dwells on TV was considerably longer than those on the tablet. Their study seemed to indicate that attention towards the TV was caused by visual cues, and the attention towards the tablet by auditory cues. Furthermore, automatic updates on the tablet attracted the view of the user after a few seconds. A study by Holmes et al. [23] attributed 30% of the attention to the tablet. The difference with the study by Brown et al. [6] is probably due to the difference in program content. Both studies indicate that during less interesting content such as commercials, attention is driven to the tablet.

The main idea is that attention can be drawn to a screen by using auditory, visual or even haptic (in the case of tablets and smartphones) cues or stimuli. Neate et al. [29] designed such an experiment to investigate which stimuli, and even combination of stimuli, would work best to steer the attention of the user. An important finding was that auditory cues on the tablet attract the attention faster than visual cues on the TV. An explanation for this is that humans can process auditory information faster than visual information. However, the participants involved subjectively preferred a notification on the TV. The insight is that the auditory cue is more of a reflex in contrast with the notification on the TV. In the latter case, it is more of a signal after which the user voluntarily decides to look at their second screen. Neate et al. [29], therefore, recommend auditory cues when the new information on the second screen only lasts a couple of seconds, and visual cues when novel information on the tablet is available for a longer period of time. In another experiment, Almeida et al. [3] recommend the combination of a visual notification on TV and a haptic stimulus on the tablet.

Valuch et al. [33] studied yet another aspect of transferring the attention of one screen to the next. More specifically, they investigated what happens after a cut is made in the scene that is being shown. Their results indicate that after a cut, people are more likely to pick up the scene at another location (the second screen for example) when the scene on the new location is visually similar to the one made before the cut. When a different scene is shown, it takes more time for people to orient themselves after the cut.

An entirely different, but interesting approach to attention diversion in second-screen applications was devised by Centieiro et al. [7]. Their idea was to mostly avoid attention switches. They achieved this by designing a very easy-to-use betting app on a smartphone, which users could employ to make a bet when a goal was about to happen. Certainly, this will not be applicable to all types of interactions in second-screen applications. However, it might be an interesting idea when the response of the user is very time sensitive.

10.2.4 Synchronizing Time-Shifted Second-Screen Experiences

A crucial element to consider in this chapter is what kinds of synchronization are to be supported. After all, we do not only have linear broadcast television anymore. People are increasingly watching content when it suits them; they watch content on-demand. Furthermore, people also tend to go back in time to review what has been offered before on the second-screen application [20]. Therefore, we distinguish three types of synchronization that impact the use of second-screen companion apps: synchronization between the app and the TV show during the broadcast; delayed viewing or time-shifting; and reviewing content.

In the first case, the content and interactivity of the second-screen application need to be in sync with the content on TV. Viewers appreciate it a lot when updates on the second screen are well synchronized with the show, but are easily annoyed when content is badly synchronized as this interferes with their experience of the television show. An interesting feature discussed in Geerts et al. [20], is the displaying of a text message received by the main character on the TV program, on the second screen. If the message on the second screen arrives too soon or too late, the relevance of this message will be completely lost for the viewer.

Second, when viewing time-shifted content, viewers want to be able to use the companion app in a synchronized way with delayed viewings [20], similar to how it happens on live TV. An interesting approach to offering delayed viewing experiences was designed and evaluated by Basapur et al. [5]. The prototype they developed allows people to create contextual feeds that are synced to TV shows. This means that friends can attach extra information and their own social comments onto a feed that will be presented on a second screen. Later on, a viewer may watch this TV program, and experience the social comments and additional information at the same place in time as was originally created. In a way, social experiences are transferred in time.

Finally, Geerts et al. [20] observed during an evaluation of a second-screen prototype for a drama program on TV, that users sometimes wanted to break out of the synchronization of the second screen with the TV show. In these instances, they revisited second-screen content that was shown earlier on the TV show. At the time of the evaluation the application did not allow viewers to revisit content after the

end of the broadcast, although both viewers and broadcasters certainly were in favour of such functionality.

10.2.5 Types of Content in Second-Screen Scenarios

Augmenting the experience with a second-screen app can be achieved in several ways: interactive elements such as questions, polls, puzzles, etc., provide a very active way of involving the audience. Figure 10.7 shows such a second-screen application that was designed for a quiz on the Dutch public broadcaster's show 'Een tegen 100' (one against one hundred). Viewers at home could play along using their second-screen application (in this case a smartphone as shown on the right). Viewers in the same room in the home could then compete against each other; the scores of each player at home is shown on the TV screen using the HbbTV standard. So, this application augments the traditional viewing experience of a quiz by actually allowing you to play along using the second-screen application. Moreover, it is possible to add a social component if other people in the home want to play along. In this section, we will describe the main types of content that can be used in second-screen scenarios, and how their use can influence the experience.

One of the more popular means of providing additional interactivity for the viewer is adding polls. In the TV-RING project, we investigated the use of polls in an analysis of the second-screen application that was created for the drama program 'De Ridder' (see earlier). Participants indicated that they felt that the polls led to more discussion inside the household. They also argued for more interaction between the results of the poll and the storyline on the program. In other words, it would be interesting if the results of the polls would determine at least parts of the storyline. Obviously, this is not so easy to realize and requires a significant amount

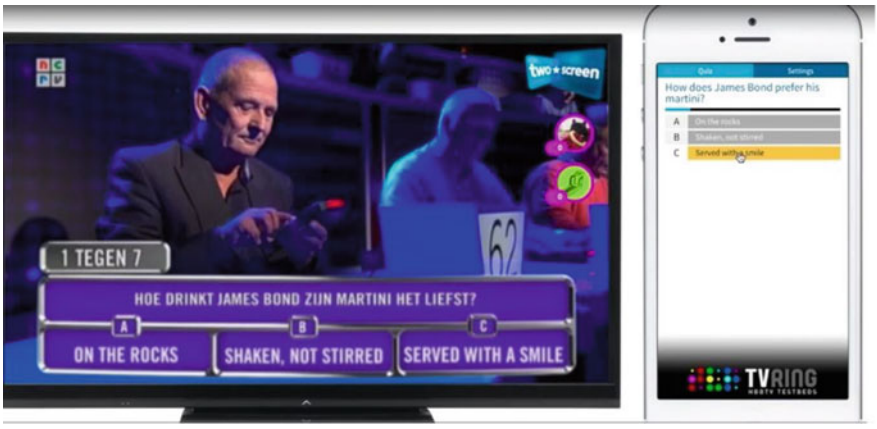


Fig. 10.7 Screenshot of a second-screen application to play along with a quiz show

of effort on the production side. Furthermore, we recommend to only include polls that are really interesting, and we recommend to avoid showing them too often. Some of our participants felt that they lost interest because of this. Therefore, it is better to provide polls about topics that are not too trivial, and to avoid polls about information that is too directly available by watching the program. For example, in the drama program 'De Ridder', the second-screen application offered a timeline of the episode, and showed quotes that were made by the characters in the show. Participants in the study did not find this compelling at all, since it was just repeating information that they just saw [20]. Polls or questions that are provocative or polarizing are preferred. In the evaluation of the second-screen application for the reality TV show 'De Rijdende Rechter', about a judge who drives around The Netherlands to solve neighbourly disputes, the second-screen application included a question that would ask the viewer which party in the dispute they thought was right. Participants who evaluated the second-screen application when the program was broadcast reported that this led to a lot of discussion in the household. As mentioned earlier, allowing the poll results to have a real impact on, or presence, in the actual TV program can greatly improve the second-screen experience. In those cases, the results have to be analysed quite rapidly after the closing of the polls, this analysis needs to happen behind the cameras, and the answers or input from the viewers need to be edited. Then, great coordination is required to bring the processed results from behind the cameras to the foreground of the program.

The genre of the program for which a second-screen application is created plays an important role. Earlier research already showed its influence on the sociability of Social TV applications [18]. For drama programs, such as 'De Ridder' which we discussed earlier, it is quite difficult to provide interaction during the broadcast. Viewers want to pay attention to the show. Therefore, they can become distracted when the second-screen application interrupts the experience too often via polls or information updates. When you offer very few updates, viewers reported to lose track of the second screen entirely. The balance between offering sufficient updates and too many updates is very difficult to realize in drama programs. In such genres, we recommend to provide updates in-between broadcasts so as to keep the viewer engaged for the next episode. This can be achieved by dropping hints, featuring small trailers or providing elements of the back story.

Interesting explorations concerning second-screen applications in support of drama programs on TV were conducted by Murray et al. [27] and Nandakumar and Murray [28]. Their objective was to support drama programs with highly complex story worlds. They designed and evaluated a synchronized, context-sensitive, and character-focused app for the drama 'Justified'. During an episode, the second-screen app would display character info, based on the characters in the scene on TV, and spatial cues that indicated the relationship between those characters. Clicking the characters would display more detailed information. In addition, short video clips were available that could help viewers understand difficult parts of the story. They found that users indeed were able to comprehend the storyline better. But they were not always appreciative of the way the synchronization was implemented. In the prototype, the TV program would automatically

pause when a video on the second screen was started. The participants preferred to be in control of when the TV program would pause. It is also clear from this example that such support requires significant effort. After all, the second-screen app should be in sync with the TV program, in order to know when to display the relevant information.

Another type of content is available in the form of social media. In most cases for second-screen use, integrating social media might distract the viewer from the show. However, in a number of instances, viewers are actually interested to know how other viewers feel about certain statements made in a program or on a second-screen application. This was expressed by participants taking part in the live pilot application of 'De Rijdende Rechter'. When talking about this aspect, these participants also mentioned other programs such as 'Kassa', a consumer program critically investigating claims made by companies about products or services that were not under evaluation in our project. Such programs, and perhaps also politically oriented programs, might provide a good opportunity to embrace the user-generated content from Twitter for example. It is then critical to provide sufficient editing of foul language, etc., if one wants to include such content into the second-screen application. It will be also a challenge to make a workable selection of interesting views to augment the discussion without making this aspect outgrow the second-screen experience. It is possible in that case, that the viewer becomes even more distracted from the TV program.

10.3 Synchronization in Social TV Applications

In this section, we present the main issues regarding good synchronization for Social TV applications. The main objective is to support the social interaction between participants using Social TV applications; their TV experience should be

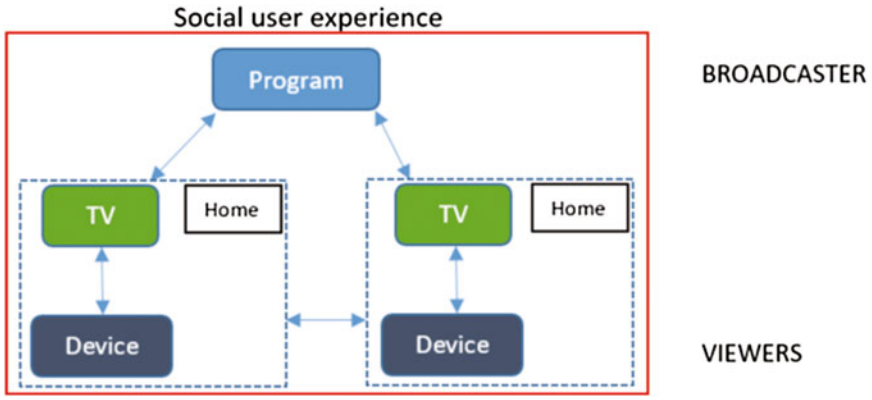


Fig. 10.8 Synchronization in Social TV applications

augmented, not impaired. Figure 10.8 illustrates the main elements in Social TV experiences, and the lines along which synchronization needs to be addressed.

In Social TV applications, two types of synchronization issues are at play. On the one hand, there is the synchronization of the communication channels. While people are watching online video together, they can use voice chat and/or text chat to engage in social interaction. These channels of communication resemble a 'backchannel' through which people can communicate about what happens on the TV program. Similar to the second-screen phenomenon discussed in the previous section, the TV is here considered to be the main focal point for the viewer's attention while communication tools, such as voice and text chat, form the secondary focus of attention. In the first part of this section, we will discuss how synchronization plays a role regarding these communication channels.

On the other hand, there is the synchronization of content between remote locations, while people are communicating. This is especially relevant for Social TV as people want to create a shared watching experience, where bad synchronization of the content they are watching can hinder the social interaction as well as the overall experience. Issues related to synchronizing content while communicating is discussed in the second part of this section.

10.3.1 Synchronizing Communication Channels

The two most important communication channels that Social TV applications offer to enable TV viewers to interact with each other are voice chat (audio only) and text chat, which are fundamentally different in nature. Communication channels can be synchronized in two ways: either in relationship to the content (where users have to carefully balance their attention between communicating and following the TV program), or in relationship to other communication channels (i.e. synchronizing the interaction with remote viewers). In both cases, the synchronization is more in the hands of the viewers themselves, supported by the technology at hand. We will discuss both perspectives in the following paragraphs.

In one of our earlier studies [17], voice chat has been found to feel more natural and direct, compared to text chat. Moreover, people found that voice chat allows them to follow the program more easily as they do not have to look at a keyboard or other device while communicating. However, using voice chat during TV can cause difficulties because viewers hear the sound of the TV and the sound of the person with whom they are engaged in conversation. It becomes even more difficult when the sound of the other person's voice is transmitted through the same speakers as the sound from the TV show [17]. Such problems make it harder for people to really comprehend what happens on the TV program. Note that this is similar to what happens when several people are engaged in conversation while watching TV in the same room. A solution could be to provide a separate set of speakers for the voice chat conversation so it is easier to distinguish it from the TV source.

On the other hand, for an audience that has much more experience using text chat applications in general, text chat showed to be much easier to combine with watching TV, and would a very useful addition. Even so, when we evaluated text chat in Social TV applications, we noticed that the act of communication requires more (cognitive) effort, because all the messages have to be typed into an application. Furthermore, people are required to look down for the typing (input), and to see what they have typed and what their conversational partner has typed (output). Therefore, they might sometimes miss what happens on the TV show—especially, visual events that are not supported by any kind of audio. Dividing one's attention between the TV program and the text chat application is the cause of most problems. This is similar to what we have discussed earlier in the section on second-screen applications.

While our lab study showed quite some benefits of voice chat over text chat, mainly in relation to synchronize viewers' communication activities with watching the content, participants in a field study by Tullio et al. [32] preferred text chat over voice chat. The main reason for this preference had to do with viewers synchronizing their reactions with the communication of remote viewers. Using text chat, the participants could decide when to react to a message from remote viewers. Using voice chat, an immediate response is required as it is very unnatural or even impolite not to respond when someone is talking. In practice, the participants of the field study mixed both communication channels, depending on their needs, which show that a flexible system offering various modes of communication is preferable in a Social TV application.

10.3.2 Synchronizing Content While Communicating

When the challenge of synchronizing communication channels is solved, another important challenge for Social TV is synchronizing the content that is being watched while communicating. Social TV applications aim to provide social viewing experience across locations (Cesar and Geerts 2011). Although it is possible to chat while watching different or unsynchronized content, this would not create a shared social watching experience, so both locations would need to view the same events at the same time [19]. In practice, this can be cumbersome to realize. The question, therefore, becomes: how large can the timing differences become between the two locations? This is especially relevant as not only on-demand content needs to be synchronized, but live broadcast content can have quite big differences of multiple seconds as well, e.g. because of the digitalization process. During big sports events such as the FIFA world championship football this becomes very tangible, when two adjacent pubs broadcast the same match, one using analogue TV and another digital TV. When one of the teams score, the cheers can be heard in the first pub, and only seconds later viewers in the second pub see the goal. This clearly creates a completely different experience in both pubs [26]. Another synchronization problem occurs when the communication channel, text chat or voice chat, has some

delay in relation to the TV program that is shown. Such synchronization issues have a severe impact on the user experience. In the following paragraphs, we will discuss these kinds of synchronization issues.

There are three important measures that can be used to evaluate synchronization for Social TV applications: togetherness [39], noticeability and annoyance [16]. Togetherness refers to the goal we are trying to achieve via Social TV applications, namely that people in different locations have an experience as if they are located in the same room. Noticeability refers to the point at which the synchronization difference becomes noticeable. Finally, annoyance refers to the point at which the synchronization difference becomes a problem for the people using the application.

A first important insight is that voice chatters feel slightly more together that text chatters [16]. This might be caused by the fact that voice chat feels more natural than text chat and creates more synchronized communication, as discussed earlier. Second, a distinction needs to be made between active and non-active chatters. This distinction is quite important since active chatters experience a higher level of togetherness. Results have shown that there is no difference in togetherness between voice chatters and active text chatters. Third, when evaluating synchronization differences from 0 to 4 s, no significant differences for both noticeability and annoyance are found when participants use text chat. This is perhaps due to the asynchronous nature of text chat: people might attribute later responses of their conversational partner to the fact that they just respond later. Fourth, when we evaluate synchronization differences from 0 to 4 s for voice chat, there is a point after which the synchronization difference becomes problematic. Up to 2 s people tend not to notice, or be annoyed. When the synchronization difference becomes larger than 2 s, the conversational partners will experience severe difficulties (see Fig. 10.9). Finally, as the highest levels of togetherness were found at up to 1 s of delay, the overall recommendation is that the maximum synchronization difference

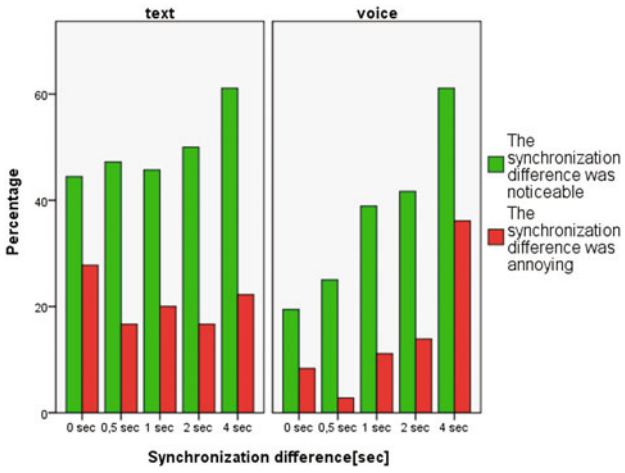


Fig. 10.9 Noticeability and annoyance of play-out differences

in Social TV applications should be 1 s. This result is partly in line with the study of Mekuria et al. [26] on synchronization while watching a football match that showed that 1 s is noticeable but not so annoying, and 4 s or more are very annoying [26]. It is interesting that the threshold for noticeability of the delay is lower for football matches than for quiz shows, which again highlights the importance of program genres in relation to interactive features [18].

10.4 Conclusions

In this chapter, we have discussed the user experience of media synchronization from various angles and in very specific scenarios. While the studies and results are of course directly related to the applications we worked with, to conclude, we would like to frame these issues in a larger perspective by drawing some more general lessons learnt, as well as highlight some open questions that need to be addressed still. The main lessons learnt we can identify, relate to usability, the role of attention, the relationship of viewers with time, and the impact of genres.

First of all, and not surprisingly when talking about user experience, the usability of any technical solution needs to be taken care of. Throughout the various user studies we have conducted, bad usability always hindered the user experience, even if the social interaction (or more broadly speaking 'sociability') was very well designed. So in order to create a good user experience of synchronization, step one is to make sure the setup and interaction related to synchronization are usable as well.

Second, adding interactivity to television watching inevitably leads to a need for attention management by the viewer. A smooth synchronization experience can support this user task, by providing the right tools and context for users to manage their attention between the different devices (in a second-screen scenario) or different activities (in a Social TV scenario).

Third, there are many different ways that timing and synchronization in media consumption play a role, especially when adding social interaction between viewers. There is synchronization of content on the first and the second screen, complicated by the fact that watching TV and video is no longer live but increasingly time-shifted. There is synchronization by viewers themselves between communication channels when social interaction is involved. And there is synchronization of content in remote locations when such a communication link is provided. As such, there is not one way to think about the user experience of media synchronization, and often these different types of synchronization need to be combined—increasing the complexity of the solution offered.

Finally, all of our own studies, and many other studies in the related literature, show the importance of program genres for the design of applications for interactive television and online video. Synchronization is not an exception here, so any solution will need to carefully consider the genre of the programs that are being watched while adding interactivity to this viewing experience.

Looking forward, we think that there is a need for more user research, both qualitative user studies for in-depth knowledge on how to design better media synchronization and quantitative experiments for teasing out specific aspects such as tolerance for delays or preferences for certain solutions. The topic of genre seems an area that still needs to be researched further, as most studies focus on one specific TV program. Sports programs and quiz shows have been researched quite well, but many other genres that also create social experiences or that can benefit from second-screen interactivity such as soap operas or interactive documentaries might bring up other synchronization challenges.

We hope that this chapter, by bringing together more than a decade of research in the user experience of media synchronization, provided useful insights as well as inspiration for further research into this exciting and quickly evolving area.

Acknowledgments A significant part of the results presented in this chapter was obtained in the TV-Ring project (EC grant agreement ICT PSP-325209). We would like to thank all partners in the TV-Ring project. Special thanks to NPO of the TV-Ring consortium with whom we have conducted this research.

Definitions

Time-shifting watching broadcasted video content at a different moment; (before or after) it is being broadcast.

Second screen screen-based device such as a smartphone, tablet or laptop that is used while watching television (the first screen).

Second-screen application an application that can be used on a laptop, tablet or smartphone in conjunction with the television program.

Social TV Interactive application that allows television viewers in remote locations to socially interact with each other.

References

1. Abreu, J., Nogueira, J., Becker, V., Cardoso, B.: Survey of catch-up TV and other time-shift services: a comprehensive analysis and taxonomy of linear and nonlinear television. Telecommun. Syst. 1–18 (2016)
2. Adamczyk, P.D., Bailey, B.P.: If not now, when?: The effects of interruption at different moments within task execution. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 271–278. ACM, New York, USA (2004)
3. Almeida, P., Abreu, J., Silva, T., Duro, L., Aresta, M., Oliveira, R.: Notification Mechanisms In Second-Screen Scenarios (2015)
4. Barkhuus, L., Brown, B.: Unpacking the television: User practices around a changing technology. ACM Trans. Comput. Hum. Interact. (TOCHI) **16**, 15 (2009)

5. Basapur, S., Mandalia, H., Chaysinh, S., Lee, Y., Venkitaraman, N., Metcalf, C.: FANFEEDS: evaluation of socially generated information feed on second screen as a TV show companion. In: Proceedings of the 10th European Conference on Interactive TV and Video, pp. 87–96. ACM, New York, USA (2012)
6. Brown, A., Evans, M., Jay, C., Glancy, M., Jones, R., Harper, S.: HCI over multiple screens. In: CHI '14 Extended Abstracts on Human Factors in Computing Systems, pp. 665–674. ACM, New York, USA (2014)
7. Centieiro, P., Romão, T., Dias, A.E.: From the lab to the world: studying real-time second-screen interaction with live sports. In: Proceedings of the 11th Conference on Advances in Computer Entertainment Technology, pp. 14:1–14:10. ACM, New York, USA (2014)
8. Cesar, P., Bulterman, D.C.A., Jansen, A.J.: The ambulant annotator: empowering viewer-side enrichment of multimedia content. In: Proceedings of the 2006 ACM Symposium on Document Engineering, pp. 186–187. ACM, New York, USA (2006)
9. Cesar, P., Geerts, D.: Past, present, and future of social TV: A categorization. In: Proceedings of Consumer Communications and Networking Conference (CCNC), pp. 347–351. IEEE (2011)
10. Cesar, P., Geerts, D.: Social interaction design for online video and television. In: Nakatsu, R., Rauterberg, M., Ciancarini, P. (eds.) Handbook of Digital Games and Entertainment Technologies, pp. 1–37. Springer, Singapore (2015)
11. Chorianopoulos, K., Fernández, F.J.B., Salcines, E.G., de Castro Lozano, C.: Delegating the visual interface between a tablet and a TV. In: Proceedings of the International Conference on Advanced Visual Interfaces, pp. 418–418. ACM, New York, USA (2010)
12. Coppens, T., Trappeniers, L., Godon, M.: AmigoTV: Towards a social TV experience. In: Masthoff, J. Griffiths, R., Pemberton, L. (eds.) Proceedings of EuroITV2003, University of Brighton (2004)
13. Darnell, M.J.: How do people really interact with TV?: naturalistic observations of digital TV and digital video recorder users. Comput. Entertain. (CIE) **5**, 10 (2007)
14. De Meulenaere, J., Bleumers, L., Van den Broeck, W.: An audience perspective on the 2nd screen phenomenon. J. Media Innovations **2**, 6–22 (2015)
15. Ducheneaut, N., Moore, R.J., Oehlberg, L., Thornton, J.D., Nickell, E.: Social TV: designing for distributed, sociable television viewing. Int. J. Hum. Comput. Interact. **24**, 136–154 (2008)
16. Geerts, D., Vaishnavi, I., Mekuria, R., van Deventer, O., Cesar, P.: Are we in sync?: synchronization requirements for watching online video together. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 311–314. ACM, New York, USA (2011)
17. Geerts, D.: Comparing voice chat and text chat in a communication tool for interactive television. In: Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles, pp. 461–464. ACM, New York, USA (2006)
18. Geerts, D., Cesar, P., Bulterman, D.: The implications of program genres for the design of social television systems. In: Proceedings of the 1st International Conference on Designing Interactive User Experiences for TV and Video, pp. 71–80. ACM, New York, USA (2008)
19. Geerts, D., De Grooff, D.: Supporting the social uses of television: Sociability heuristics for social TV. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 595–604. ACM, New York, USA (2009)
20. Geerts, D., Leenheer, R., De Grooff, D., Negenman, J., Heijstraten, S.: In front of and behind the second screen: viewer and producer perspectives on a companion app. In: Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video, pp. 95–102. ACM, New York, USA (2014)
21. Harboe, G., Massey, N., Metcalf, C., Wheatley, D., Romano, G. The uses of social television. ACM Comput. Entertain. (CIE) **6**, 1 (2008)
22. Hawkins, R.P., Pingree, S., Hitchon, J., Radler, B., Gorham, B.W., Kahlor, L., Gilligan, E., Serlin, R.C., Schmidt, T., Kannaovakun, P., Kolbeins, G.H.: What produces television attention and attention Style? Hum. Commun. Res. **31**(1), 162–187 (2005)

23. Holmes, M.E., Josephson, S., Carney, R.E.: Visual attention to television programs with a second-screen application. In: Proceedings of the Symposium on Eye Tracking Research and Applications, pp. 397–400. ACM, New York, USA (2012)
24. Holz, C., Bentley, F., Church, K., Patel, M.: "I'M just on my phone and They'Re watching TV": quantifying mobile device use while watching television. In: Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video, pp. 93–102. ACM, New York, USA (2015)
25. Ley, B., Ogonowski, C., Hess, J., Reichling, T., Wan, L., Wulf, V.: Impacts of new technologies on media usage and social behaviour in domestic environments. Behav. Inf. Technol. **33**, 815–828 (2014)
26. Mekuria, R., Cesar, P., Bulterman, D.: Digital TV: The effect of delay when watching football. In Proceedings of the 10th European Conference on Interactive TV and Video, pp. 71–74. ACM, New York, USA (2012)
27. Murray, J., Goldenberg, S., Agarwal, K., Chakravorty, T., Cutrell, J., Doris-Down, A., Kothandaraman, H.: Story-map: IPad companion for long form TV narratives. In: Proceedings of the 10th European Conference on Interactive TV and Video, pp. 223–226. ACM, New York, USA (2012)
28. Nandakumar, A., Murray, J.: Companion apps for long arc TV series: supporting new viewers in complex storyworlds with tightly synchronized context-sensitive annotations. In: Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video, pp. 3–10. ACM, New York, USA (2014)
29. Neate, T., Jones, M., Evans, M.: Mediating attention for second-screen companion content. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 3103–3106. ACM, New York, USA (2015)
30. Palviainen, J., Kuusinen, K., Väänänänen-Vainio-Mattila, K.: Designing for presence in social television interaction. In: Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedi, pp. 9:1–9:10. ACM, New York, USA (2013)
31. Rooksby, J., Rost, M., Morrison, A., Bell, M., Chalmers, M., Smith, T.: Practices of parallel media: using mobile devices when watching television. Presented at the Designing with Users for Domestic Environments: Methods, Challenges: Lessons Learned, Baltimore, MD, USA (2014)
32. Tullio, J., Harboe, G., Massey, N.: Investigating the use of voice and text chat in a social television system. In: Proceedings of the 6th European Conference on Changing Television Environments, pp. 163–167. Berlin, Heidelberg: Springer (2008)
33. Valuch, C., Ansorge, U., Buchinger, S., Patrone, A.R., Scherzer, O.: The effect of cinematic cuts on human attention. In: Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video, pp. 119–122. ACM, New York, USA (2014)
34. Vanattenhoven, J., Geerts, D.: Broadcast, video-on-demand, and other ways to watch television content: a household perspective. In: Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video, pp. 73–82. ACM, New York, USA (2015)
35. Vanattenhoven, J., Geerts, D.: Second-screen use in the home: an ethnographic study. In: Proceedings 3rd International Workshop on Future Television, EuroITV 2012, p. 12. Springer, Berlin (2012)
36. Vanattenhoven, J., Geerts, D.: Social experiences within the home using second-screen TV applications. Multimed. Tools Appl. 1–29 (2016)
37. Van Cauwenberge, A., Schaap, G., van Roy, R.: "TV no longer commands our full attention": Effects of second-screen viewing and task relevance on cognitive load and learning from news. Comput. Hum. Behav. **38**, 100–109 (2014)
38. Vatavu, R.-D., Mancas, M.: Visual attention measures for multi-screen TV. In: Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video, pp. 111–118. ACM, New York, USA (2014)

39. Weisz, J.D., Kiesler, S., Zhang, H., Ren, Y., Kraut, R.E., Konstan, J.A.: Watching together: integrating text chat with video. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 877–886. ACM (2007)

Chapter 11
Media Synchronization in Networked Multisensory Applications with Haptics

Pingguo Huang, Mya Sithu and Yutaka Ishibashi

Abstract In this chapter, we explain the present status of studies for media synchronization in networked multisensory applications with haptics. We also specify the characteristics of haptic media and different features of haptic media from other media such as olfactory, auditory, and visual media. By using such other media together with haptic media, we can get higher realistic sensation when we use the applications for various purposes such as remote education, entertainment, and networked games. When multisensory media streams are transmitted over a network like the Internet, the temporal relationships among the media streams may be disturbed owing to the network delay, delay jitter, and packet loss. Thus, the quality of experience (QoE) may seriously be degraded. To solve this problem, we need to carry out media synchronization control. To achieve a high quality of media synchronization, a number of media synchronization algorithms have been proposed so far. Especially, in networked multisensory applications, we need to take account of human perception of media synchronization errors in the algorithms. This is because the requirements of media synchronization quality depend on types of media. Some algorithms such as Virtual-Time Rendering (VTR) take human perception of the errors into account. For example, VTR tries to accomplish the synchronization by changing the buffering time of each media stream dynamically according to the network delay jitter with several threshold values about the errors. In the algorithms, instead of synchronizing the output timings of media streams exactly, ranges of human perception such as *the allowable range*, in which users feel that the synchronization error is allowable, *the imperceptible range*, in which users cannot perceive the error, and the *operation range*, which is narrower than the imperceptible range and should be kept usually, are taken into account for the sake

P. Huang (✉)
Seijoh University, Aichi 476-8588, Japan
e-mail: huangpg11@gmail.com; huangpg@seijoh-u.ac.jp

M. Sithu · Y. Ishibashi
Nagoya Institute of Technology, Nagoya 466-8555, Japan

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_11

of high synchronization quality. In this chapter, we explain the algorithms taking account of human perception of the media synchronization errors and enhance other algorithms such as the group (or inter-destination) synchronization control and the adaptive Δ-causality control algorithms for simultaneous output-timing control among multiple terminals by taking account of the perception. It is indispensable to clarify the ranges by QoE assessment in networked multisensory applications. In this chapter, we further make a survey of studies on the assessment. Finally, we discuss the future directions of media synchronization in networked multisensory applications with haptics.

Keywords Media synchronization algorithm · Networked multisensory applications · Haptics · Human perception

11.1 Introduction

Over the last decade, users around the world communicate with each other by using high realistic multisensory applications which employ two or more senses for various kinds of activities such as remote education, networked entertainment, and networked games. In the recent years, the networked multisensory applications are becoming more advanced by using haptic sense (i.e., the sense of touch) and olfactory sense in addition to traditional visual and auditory senses [1–8]. Especially, haptic sense attracts the attention of users, researchers, and developers because of its important contribution to high realistic networked multisensory applications. In the networked multisensory applications with haptics, users can feel the haptic sense when they touch objects by using haptic interface devices. They can also feel the shape, weight, and softness of the objects. There is a wide spectrum of applications that incorporate haptic technology in the areas of music performance, games, teleoperations, remote education, and so on [9–15].

When users at different places employ such applications together over a network, multisensory media streams need to be synchronized. However, when multisensory media streams are transmitted over a network which does not guarantee the Quality of Service (QoS) [16] like the Internet, the temporal relationships among the media streams may be disturbed owing to the network delay, delay jitter, and packet loss. Thus, the quality of experience (QoE) [17] may seriously be degraded. To solve this problem, we need to carry out media synchronization control [18–27].

To achieve a high quality of media synchronization, a number of media synchronization algorithms have been proposed so far [19, 20]. Especially, in the networked multisensory applications, we need to take account of human perception of media synchronization errors in the algorithms. This is because the requirements of media synchronization quality depend on types of media. Some algorithms such

as Virtual-Time Rendering (VTR) [21] and the algorithms in [22] and [23] take human perception of the errors into account for intra-stream synchronization and interstream synchronization [24]. For example, VTR tries to accomplish the intra-stream and interstream synchronization by changing the buffering time of each media stream dynamically according to the network delay jitter with several threshold values about the errors. In the algorithms, instead of synchronizing the output timings of media streams exactly, ranges of human perception such as *the allowable range* [22], in which users feel that the synchronization error is allowable, *the imperceptible range* [22], in which users cannot perceive the error, and the *operation range* [23], which is narrower than the imperceptible range and should be kept usually, are taken into account for the sake of high synchronization quality. We can apply this idea to algorithms for *simultaneous output-timing control* among multiple terminals which tries to output each media stream at the same time among the terminals such as the dynamic local lag control, the group (or inter-destination) synchronization control, and the adaptive Δ-causality control [25–27]. By taking account of the human perception in the simultaneous output-timing control algorithms, we can largely improve the simultaneous output quality.

In this chapter, we explain the algorithms taking account of human perception of the media synchronization errors. We also enhance the simultaneous output-timing control algorithms among multiple terminals by taking account of the perception.

In this chapter, first, we explain haptics in Sect. 11.2. Next, we make a survey of networked multisensory applications with haptics in Sect. 11.3. We also discuss the requirements for media synchronization in Sect. 11.4. We explain media synchronization algorithms taking account of human perception in Sect. 11.5, and enhance the simultaneous output-timing control algorithms in Sect. 11.6. It is indispensable to clarify the human perception ranges by QoE assessment in networked multisensory applications. Therefore, in Sect. 11.7, we further make a survey of studies on the assessment. Finally, we discuss the future directions of media synchronization in networked multisensory applications with haptics in Sect. 11.8 and conclude the chapter in Sect. 11.9.

11.2 Haptics

The term “haptics” was descended from the Greek word “haptesthai” meaning “relating to the sense of touch.” It was introduced by researchers in the area of psychology at the beginning of the twentieth century to refer to touch of real objects by humans. In the late 1980s, the term was redefined to include all aspects of human–computer touch interaction [28, 29]. Nowadays, the term “haptics” is being widely used in the areas of biomechanics, psychology, engineering, and computer science for the study of human touch and force feedback. Haptic sensation can be

perceived via haptic interface devices. Currently, there are a number of commercially available haptic interface devices. In this section, we briefly discuss the haptic interface devices and haptic media.

11.2.1 Haptic Interface Devices

By using haptic interface devices in networked virtual environments, users can get the force feedback via the devices when they touch objects in 3D virtual spaces. By comparing to video update rates, haptic interface devices need higher update rates for the feeling of realistic touch. Current haptic interface devices have 1 kHz or more input/output frequency [30, 31]. Because of this high-frequency rate, users can feel the sense of touch naturally in the virtual environments. There are several types of haptic interface devices such as 1-DoF (Degree-of-freedom), 2-DoF, 3-DoF, and 6-DoF devices [31, 32]. Degree-of-freedom means the freedom of movement of the virtual object in the 3D space. As the number of DoF increases, the device becomes more natural and intuitive, but it becomes more complicated because the number of haptic sensors and actuators increase. By using 6-DoF haptic interface devices, the virtual object can be manipulated in the directions of left/right, up/down, and forward/backward, combined with changes in orientation (i.e., pitch, yaw, and roll). Examples of haptic interface devices are Geomagic Touch (see Fig. 11.1a) [33], Geomagic Touch X (see Fig. 11.1b) [34], PHANTOM Premium 1.5 [35], Omega [36], SPIDAR-G AHS (see Fig. 11.1c) [37], Falcon (see Fig. 11.1d) [38], Virtuose [39], HapticMaster [40], CyberGrasp, CyberForce [41], and so on. All the devices excluding CyberGrasp and CyberForce are mainly manipulated by a single interaction point, which a user operates to touch an object.

11.2.1.1 Geomagic Touch

Geomagic Touch [33] (formerly called PHANToM Omni) was developed by Artificial Intelligence laboratory at MIT (Massachusetts Institute of Technology). Spring-damper model is used for calculation of reaction force. The Geomagic lineup of haptic interface devices includes Geomagic Touch, Geomagic Touch X

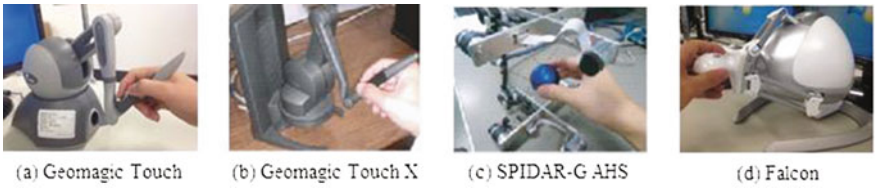


Fig. 11.1 Examples of haptic interface devices

[34] (formerly called PHANToM Desktop), the family of PHANToM Premium models [35], and the Touch 3D stylus [42].

The Geomagic Touch can work in the workspace of 6.4 Width, 4.8 Height, and 2.8 Diagonal in inches. The force feedback can be got in 3-DoF, the positional sensing can be got in 6-DoF. The maximum reaction force is 3.3 N. The workspace of Geomagic Touch X is within 6.4 Width, 4.8 Height, and 4.8 Diagonal in inches. Like the Geomagic Touch, the force feedback can also be got in 3-DoF, and the positional sensing can be got in 6-DoF. The maximum reaction force is 7.9 N. The Touch 3D stylus is a new product of Geomagic and it is mainly intended for making 3D designs. Instead of a cursor of the device in a virtual space, sculpting tools can be used for 3D design. The PHANToM Premium lineup models include two High Force (HF) models and three 6-DoF devices.

11.2.1.2 SPIDAR

SPIDAR [37], which was developed by Sato Laboratory of Tokyo Institute of Technology, is a string-based haptic interface device. It is a two-handed multi-finger haptic interface device. SPIDAR has eight fingertip attachment devices in which each fingertip attachment device is connected to three strings. The device calculates the positions of user's fingertips by using the lengths of three strings connecting to the fingertip attachment devices, and gives force feedback when the positions of user's fingers contact with a virtual object.

11.2.1.3 FALCON

FALCON [38] was developed by Novint Technologies, Inc. Three arms are connected to three motors in the main body, and a user can control a movable ball grip at the intersection of three arms. When the arms move, optical sensors attached to the three motors keep the track of the movements of the arms. Based on the positions of arms, the position of 3D cursor is calculated by a mathematical function called "Jacobian". The force feedback is applied to the grip in any direction at every millisecond.

11.2.2 Haptic Media

In networked virtual environments, information about haptic sense can be transmitted as *media units* (*MUs*) [21] between users' terminals. The MUs are information units for media synchronization, each of which includes the position information of haptic interface device, information about reaction force, timestamp

which represents the input time of the MU, and its sequence number. There are at least two control schemes in haptic media transmission [43, 44]. One is the position–position control scheme [45], and the other is the position–force control scheme [46]. In the position–position control scheme [45], each terminal calculates the reaction force applied to its haptic interface device or an object according to the positional information (i.e., position of haptic interface device or an object at the other terminal) provided from the other terminal and the position information of the device. In the position–force scheme [46], a terminal calculates the reaction force applied to its haptic interface device based on the information about the reaction force sent from the other terminal. The reaction force sent by the other terminal is multiplied by a gain coefficient.

11.3 Networked Multisensory Applications with Haptics

There are various kinds of networked multisensory applications with haptics in both virtual and real environments. In this section, we explain some examples of the applications.

11.3.1 Applications in Virtual Environment

In virtual environments with only visual and auditory senses, user capabilities are limited to interact with the other users or objects in the same virtual environment. By using haptic interface devices, users are able to touch, feel, and manipulate 3D objects in virtual environments.

There are various multisensory applications with haptics in networked virtual environments. Examples of the applications are remote education such as the remote ikebana (Japanese flower arrangement) (see Fig. 11.2a) [9], networked real-time games such as the networked fruit harvesting game (see Fig. 11.2b) [8] and the networked balloon bursting game (see Fig. 11.2c) [10], networked

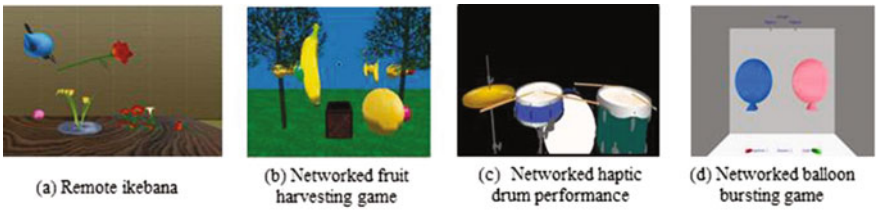


Fig. 11.2 Examples of applications

entertainment such as the networked haptic drum performance (see Fig. 11.2d) [11], and telemedicine such as the remote surgical training [12].

11.3.1.1 Remote Ikebana

In the remote ikebana [9], a teacher at a terminal can teach a student at a remote terminal how to arrange flowers. The teacher and student share a 3D virtual space surrounded by walls, a floor, and a ceiling. In the virtual space, there are flowers, one table, a flower vase which has a flower pin holder, and a pair of scissors. By manipulating a haptic interface device, the teacher or the student can hold a flower, adjust the length of the held flower's stem with the pair of scissors, and impale the flower on the flower vase. He/she perceives the reaction force via the haptic interface device when he/she touches the flower's stem or scissors, adjusts the flower, or impales the flower on the flower vase. He/she also perceives the smell of the flower via an olfactory display, when the flower is close to his/her viewpoint.

11.3.1.2 Networked Fruit Harvesting Game

In the networked fruit harvesting game [8], two users (a harvester and a harvest impeder) play the game together. In a virtual space of the game, there are some trees bearing three kinds of fruits. The smell of a fruit is randomly selected from among the three kinds of fruits. The harvester or harvest impeder can touch or pick a fruit by manipulating his/her haptic interface device. When he/she pulls a fruit, the reaction force is generated. If the reaction force exceeds a threshold value, the fruit pulled by him/her is picked off. He/she can also perceive the smell of fruit generated from an olfactory display by moving the fruit picked off toward his/her viewpoint.

The harvester picks a fruit, smells the fruit, and judges whether the smell is in agreement with its appearance or not. If the smell is in agreement with its appearance, he/she harvests the fruit. Otherwise, he/she discards the fruit. The harvest impeder picks a fruit and moves it toward the harvester's viewpoint to interfere with the harvester. By doing so, it is difficult for the harvester to judge the correct smell.

11.3.1.3 Networked Haptic Drum Performance

In the networked haptic drum performance [10], users share a drum set which consists of high-hat cymbals, a snare drum, a bass drum, and a floor tom in a 3D virtual space. The two haptic interface devices at each terminal are used to move a pair of drumsticks in the virtual space. When each drumstick hits a drum

component, the user can perceive the reaction force through the haptic interface device, and a sound depending on the drum component is generated.

11.3.1.4 Networked Balloon Bursting Game

In the networked balloon bursting game [11], each of the two players bursts balloons (i.e., soft objects) with his/her haptic interface device in a 3D virtual space. The two players compete with each other for the number of burst balloons. Each player employs his/her haptic interface device to move the virtual stylus in the 3D virtual space. When the player touches the balloon with the tip of the stylus, the reaction force is perceived through the haptic interface device; he/she can feel the softness of the balloon. The balloon is distorted when the player pushes it with the stylus. If he/she pushes it strongly, the balloon is largely distorted, and it is burst and disappeared. Then, he/she hears a sound of bursting it via his/her headset.

11.3.1.5 Remote Surgical Training

In a remote surgical training [12], an instructor and a student who are located in different remote places work together in the same virtual space, which contains 3D model of body organs. The instructor teaches the student how to operate a gall bladder removal. When they touch, grasp and stretch the body organs by using their respective haptic interface devices, they can feel the reaction force of the body organs according to the softness of the organs. The instructor can also control the student's hand. The student can feel the reaction force of the instructor's guiding hand and vice versa. They can apply diathermy to tissues and clip or cut ducts in the virtual space. If they accidentally cut the gall bladder or duct, it will bleed. Also, in the virtual space, the instructor is able to annotate the scene by drawing arrows, circles, and lines in 3D using the phantom's button.

11.3.2 Applications in Real Environment

11.3.2.1 Remote Penmanship

In the remote penmanship [13], a teacher at a terminal teaches a student at a remote terminal how to write characters or draw figures. The two terminals are connected via a network. The teacher and students can watch a video of the other terminal on a display at their respective terminal. A whiteboard marker is attached to the stylus of

each haptic interface device. The teacher can control the student's device by using his/her device.

11.3.2.2 Networked Ensemble

In the networked ensemble [14], a user at a terminal controls a haptic interface device at a remote terminal by his/her device. In this way, he/she can hit a tambourine placed at the remote terminal according to the sound of a keyboard harmonica played by a user of the remote terminal.

11.3.2.3 Remote Surgical Robot System

In a remote surgical robot system [15], a TCP/IP network connects the patient site and the surgeon site. At the patient site, two surgical manipulators are placed over the patient. The two haptic interface devices at the surgeon site control the manipulators at the patient site.

11.4 Requirements for Media Synchronization

As described in the previous section, in the networked multisensory applications, when MUs are transmitted over a network which does not guarantee the QoS, the temporal relationships among the MUs may be disturbed owing to the network delay and delay jitter (see Fig. 11.3 through Fig. 11.5). In Fig. 11.3, where a single media stream is transmitted from a source to a destination, the input intervals between MUs may be disturbed. In Fig. 11.4, where multiple media streams are transmitted, the temporal relationship among media streams may be disturbed. In Fig. 11.5, where a single media stream is transferred from a source to multiple destinations, the group synchronization may be disturbed (see Fig. 11.5).

In [47] through [49], the authors investigate the influences of interstream synchronization errors among multiple media streams in applications using haptics by experiment. Experiment results show that QoE is deteriorated as the network delay and delay jitter increase. Especially, when the delay differences (synchronization errors) among different media streams are large, QoE is seriously deteriorated.

In [50], the authors investigate the influence of network delay on the fairness in a networked real-time game. Experiment results show that the difference in network delay between different destinations affects the fairness.

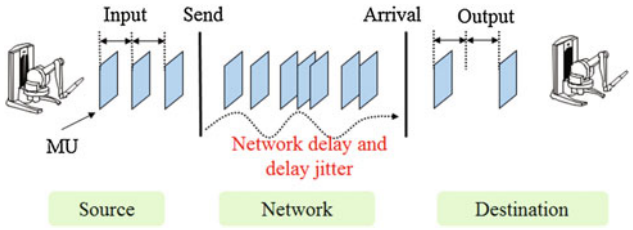


Fig. 11.3 Influence of network delay and delay jitters in single media stream

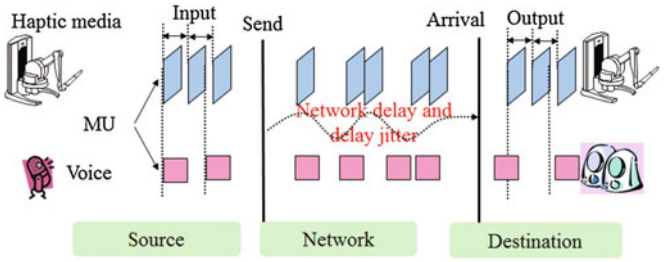


Fig. 11.4 Influence of network delay and delay jitters among multimedia streams

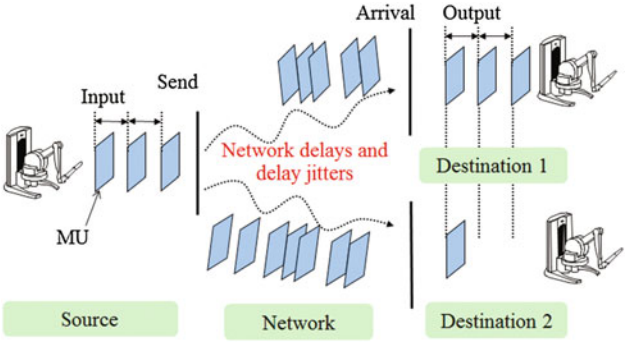


Fig. 11.5 Influence of network delay and delay jitters among different destinations

Therefore, it is necessary to carry out media synchronization control to preserve the timing relation between MUs (i.e., intra-stream synchronization control) for each media stream, to keep the temporal relationship among multiple media streams at each destination (interstream synchronization control), and to output each MU simultaneously at different destinations (group synchronization control).

11.5 Media Synchronization Algorithms Taking Account of Human Perception

There are many media synchronization algorithms which have been proposed to improve the quality of media synchronization. In this section, we introduce media synchronization algorithms taking account of human perception. Before introducing the algorithms, we will introduce the Virtual-Time Rendering (VTR) algorithm since it is employed in the algorithms.

11.5.1 Virtual-Time Rendering Algorithm

In [21], the authors employ the VTR algorithm for haptic media synchronization. In the VTR algorithm, when the network delay jitter is smaller than an estimated value of the maximum network delay jitter (i.e., the buffering time of the first MU [21]), the time at which the MU should be output is defined as the *ideal target output time*. When the network delay jitter is larger than the estimated value, the algorithm introduces the *target output time*, which is calculated by adding/subtracting a time (called the *slide time* (ΔS_n)) to/from the ideal target output time. The slide time is determined as follows.

Let T_n, A_n, x_n, t_n, d_n, ΔS_n denote the generation time, arrival time, ideal target output time, target output time, *scheduled output time* (an instant recommended for the algorithm), and slide time, respectively, of the n-th MU. When $d_n - t_n > T_{\mathrm{h2}}$ (> 0), the slide time is set to $d_n - t_n$, where T_{h2} is a threshold value which is used to judge whether the target output time should be delayed or not. When the target output time is later than the arrival time $(A_n \leq t_n)$, the algorithm determines the scheduled output time by advancing the target output time (i.e., the *virtual-time contraction*). That is, d_n is set to max $(t_n - r, x_n, A_n)$ and the slide time is set to –min $\left(r, \sum_{i=1}^{n-1} \Delta S_i\right)$ when $t_n - T_n > \Delta_{\mathrm{al}}$, or when a certain period of time has elapsed since the last late arrival or the last virtual-time contraction, where r is a positive constant and Δ_{al} is the *maximum allowable delay*. Otherwise, if $A_n > t_n$, the scheduled output time is set to the arrival time. When multiple MUs have the same scheduled output time (for example, when multiple MUs arrive at almost the same time), since the output duration of each haptic MU is 1 ms [21], the algorithm cannot shorten the output duration of an MU, and the virtual-time contraction brings discarding MUs.

11.5.2 Enhanced Virtual-Time Rendering Algorithm

In [22], the authors propose a media synchronization algorithm for haptic media and voice by enhancing the VTR algorithm. In order to keep high quality of intra-stream synchronization at the expense of slight deterioration in the interstream synchronization quality, two types of error ranges are employed in the proposed algorithm. One is the *imperceptible range,* in which users cannot perceive the error, and the other is the *allowable range*, in which users feel that the synchronization error is allowable.

In the proposed algorithm, for inter-synchronization, the haptic stream is selected as a master stream, which is the most important and/or the most sensitive to intra-stream synchronization error, and the voice stream is selected as a slave steam, which is synchronized with the master stream. This means that the output timing of master stream (haptic stream) is not affected by the slave stream (voice), and that intra-stream synchronization control is carried out over the master stream. The algorithm prioritizes the intra-stream synchronization quality of voice over the interstream synchronization quality. That is, it does not always start outputting the first haptic and voice MUs simultaneously. In order to keep high quality of inter-stream synchronization, it introduces two types of error ranges: the imperceptible range and allowable range of interstream synchronization error (see Fig. 11.6, where the *n*-th MU of the master stream has the same timestamp as the *m*-th MU of the slave stream (voice) and the two MUs should be output at the same time). If the interstream synchronization error is out of the allowable range as shown in Fig. 11.6, the algorithm tries to reduce the error gradually until the error enters the imperceptible range. The effects of the proposed algorithm is investigated by implementing the algorithm in a collaborative haptic work, and experiment results

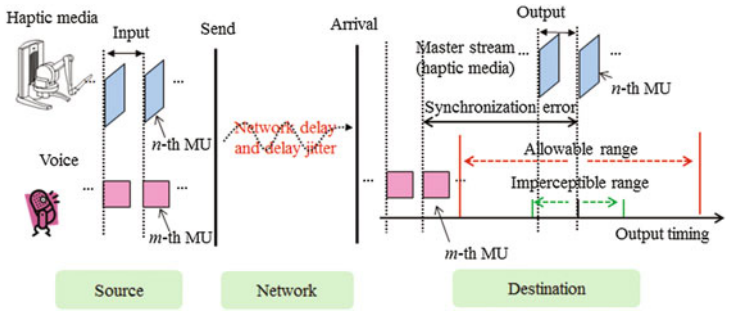


Fig. 11.6 Imperceptible range and allowable range in enhanced Virtual-Time Rendering algorithm

show that the algorithm can keep high quality of intra-stream synchronization at the expense of slight deterioration in the interstream synchronization quality.

11.5.3 *Interstream Synchronization Algorithm with Group Synchronization Control*

In [23], the authors propose interstream synchronization algorithm which uses group (inter-destination) synchronization control. The algorithm allows interstream synchronization errors among multiple media streams to some extent to obtain high quality of intra-stream synchronization of multiple media streams.

Different to the interstream synchronization algorithm described in the previous subsection, in the proposed algorithm, all the streams are handled as master streams (i.e., the interstream synchronization control of the VTR algorithm is not carried out). In order to achieve interstream synchronization among the streams, group synchronization control is carried out over all the streams. In the algorithm, all the master streams employ the VTR algorithm for intra-stream synchronization, and the destination gathers information about the output timing of each master stream when the output timing is changed by the VTR algorithm. Then, the destination determines the reference output timing by using the output timings of all the master streams. The proposed algorithm uses the two ranges; one is the imperceptible range, and the other is called the *operation range* (see Fig. 11.7). The operation range is defined as a range of the error which should be kept usually and is narrower than the imperceptible range for safety. When the interstream synchronization error is within the operation range, the current output timing is employed as the reference output timing. When the interstream synchronization error goes out of the imperceptible range, the control tries to change the output time gradually by modifying

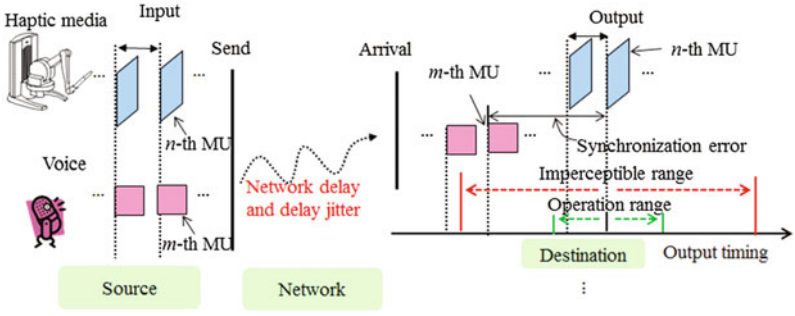


Fig. 11.7 Imperceptible range and allowable range in interstream synchronization algorithm with group synchronization control

the reference output timing so that the error enters the operation range. We explain how to modify the reference output timing in the latter case by using two media streams (say *streams 1* and *2*). When the current output timing of stream 1 $\left(\Delta^{(1)}\right)$ is later than that of stream 2 $\left(\Delta^{(2)}\right)$, and the interstream synchronization error exceeds the upper limit of the operation range $\left(R_{\text{operation}}^{\text{H}(1,2)}\right)$, the algorithm tries to delay $\Delta^{(2)}$ up to $\Delta^{(1)} - R_{\text{operation}}^{\text{H}(1,2)}$. However, in order to change the output timing gradually, for stream i ($i = 1$ or 2), there exist maximum and minimum values of $\Delta^{(i)}$ and denoted by $\Delta_{\text{H}}^{(i)}$ and $\Delta_{\text{L}}^{(i)}$, respectively. If $\Delta^{(1)} - R_{\text{operation}}^{\text{H}(1,2)} \leq \Delta_{\text{H}}^{(2)}$, the reference output timing of stream 2 is set to $\Delta^{(1)} - R_{\text{operation}}^{\text{H}(1,2)}$ and $\Delta^{(1)}$ is employed as the reference of output timing of stream 1. Otherwise, if $\Delta^{(1)} - R_{\text{operation}}^{\text{H}(1,2)} > \Delta_{\text{H}}^{(2)}$, $\Delta_{\text{H}}^{(2)}$ is employed as the reference output timing of stream 2, and the algorithm tries to advance $\Delta^{(1)}$ up to $R_{\text{operation}}^{\text{H}(1,2)} + \Delta_{\text{H}}^{(2)}$. If $R_{\text{operation}}^{\text{H}(1,2)} + \Delta_{\text{H}}^{(2)} \geq \Delta_{\text{L}}^{(1)}$, the reference output timing of stream 1 is advanced up to $R_{\text{operation}}^{\text{H}(1,2)} + \Delta_{\text{H}}^{(2)}$. Otherwise, the $\Delta_{\text{L}}^{(1)}$ is employed as the reference output timing of stream 1.

The authors also investigate the effect of the proposed algorithm in a remote control system with haptic media and video. Experiment results show that the algorithm behaves properly at a high level of quality.

11.6 Enhancement of Simultaneous Output-Timing Control

There are several types of simultaneous output-timing control such as the dynamic local lag control [25], group (or inter-destination) synchronization control [26], and the adaptive Δ-causality control [27]. When there are inter-destination errors, the types of control try to reduce the errors to zero. Normally, all the destinations try to synchronize with a destination which has the latest output timing (i.e., the destination has the largest delay and/or delay jitter, and the reference output timing is set to the latest output timing) so that all the destinations can output MUs simultaneously. This means that the destinations which receive MUs earlier need to delay the output of the MUs, and the interactivity and the quality of intra-stream and inter-stream synchronization may largely be degraded. Therefore, we can enhance the simultaneous output-timing control by taking account of human perception (i.e., the allowable range, imperceptible range, and/or operation range).

The dynamic local lag control is proposed for sound synchronization in networked virtual ensembles [25]. Under the control, each source buffers local information for Δ seconds (called the *local lag*) which is dynamically changed according to the network delay from the other terminal to the local terminal so that the user hears the sounds simultaneously. For example, in the networked haptic drum performance which employs the dynamic local lag control [25], two users try

to hit a drum set in a 3D virtual space synchronously. The sound information of local terminal is buffered by Δ seconds, which is set to the same value as the network delay from the other terminal to the local terminal. After buffering, the sound is output. At the time, the terminal receives the sound information transmitted from the other terminal and outputs the sound. Therefore, the user at the local terminal hears the sounds at the same time. We can take account of the human perceptions in the control. That is, the control adjusts the output timing of MUs only when the errors (differences) between the value of Δ calculated from the current network delay and the value of Δ used for the latest output MU are larger than the allowable or imperceptible range.

As for the group synchronization control, when each destination receives or determines the reference output timing, the destination gradually adjusts its output timing to the reference output timing [26]. As shown in Fig. 11.8, instead of synchronizing the output timings of MUs exactly, ranges about human perception such as the allowable range, in which users feel that the group synchronization error is allowable, and the imperceptible range, in which users cannot perceive the error, can be taken into account for the sake of high synchronization quality. Please note that in Fig. 11.8, the intra-stream synchronization control and group synchronization control are carried out for haptic media. Similar to the two types of control in [22] and [23], if the group synchronization errors are larger than the allowable range, the enhanced group synchronization control reduces the errors gradually until the errors enter the imperceptible range. If the group synchronization errors are within the allowable range but outside the imperceptible range, the enhanced control carries out only intra-stream synchronization control.

In the adaptive Δ-causality control, each MU has a time limit which is equal to the generation time of MU plus Δ seconds, and the value of Δ is dynamically changed according to the network delay and delay jitter [51]. However, the output of MUs are delayed by Δ seconds; this leads to the damage of interactivity at the destinations which receive MUs earlier. Therefore, we can also enhance the control by using the human perception. For example, when the errors (differences) between

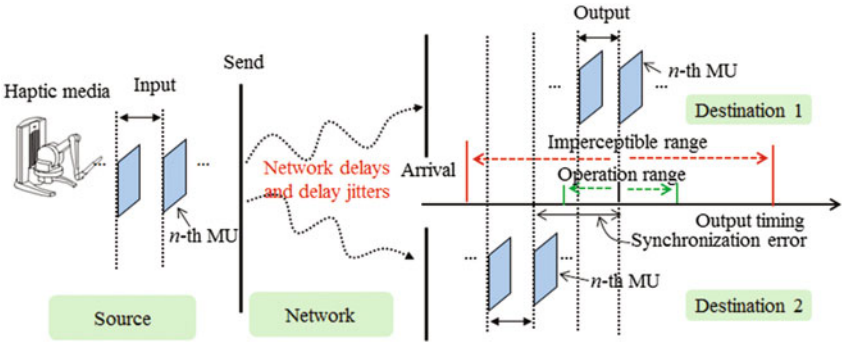


Fig. 11.8 Enhancement of group synchronization control

the values of Δ calculated from the current network delay and the value of Δ used for the latest output MU are outside of the allowable range or imperceptible range, the control starts to adjust the output timing of MUs.

11.7 Human Perception of Media Synchronization Errors

As described above, we can take account of human perception of media synchronization errors for high quality of media synchronization. There are a number of studies which focus on the influences of media synchronization errors on QoE [8, 47–50, 52, 53].

In [50], the authors investigate the fairness between two users when there exist delay differences in a networked real-time game with haptics. Experiment results show that the network delay differences larger than about 30 or 40 ms lead to unfairness; when the network delays of users are smaller than about 30 ms, it difficult to perceive unfairness.

In [47], the authors investigate the influence of synchronization error between haptic media and video in a system which transmits haptic media and video of a real object located at a remote place. Experiment results show that when the synchronization errors smaller than around 240–320 ms, as the network delay of haptic media increases, the synchronization error becomes easier to be perceived. In [48], the authors investigate the influence of interstream synchronization error in a teleoperation system with 3D (stereoscopic) video and haptic media. Experiment results show that the interstream synchronization quality is the highest when the 3D video and haptic media are output at the same time or the 3D video is output slightly ahead of haptic media. In [49], the authors investigate the influence of interstream synchronization errors among haptic media, sound, and video in a networked ensemble with a keyboard harmonica and a tambourine. Experiment results show that when the network delay of haptic media is 10 and 50 ms, in the case where one media stream's network delay is different from those of the other two media stream, interstream synchronization errors less than about 160 and 80 ms are allowable; in the case where the network delays of the three media streams are different from each other, interstream synchronization errors less than about 120 and 40 ms are allowable.

In [52], the authors investigate the influence of interstream synchronization error between olfactory and haptic media in a harvesting fruit system by QoE assessment. Experiment results show that (only in the case where the olfactory media are output earlier than haptic media) when the interstream synchronization errors are from 500 ms to around 1000 ms, users hardly perceive the errors; when the synchronization errors are from about 110 ms to around 1100 ms, the errors are allowable. In [8], the authors investigate the fairness between players in a fruit harvesting game using haptics and olfaction. Experiment results show that when the network delay at one terminal is 700 ms and the difference in network delay between two terminals is from about −200 to 200 ms, the fairness is high; when the network

Table 11.1 Human perception of media synchronization errors

Reference	Media	Synchronization	Work	Operation range	Imperceptible range	Allowable range	Others
Ishibashi and Tasaka [50]	Haptic media and CG (Computer Graphic)	Group	Networked real-time game	–	–	–	Synchronization errors > 30 ms: unfair
Fujimoto et al. [47]	Haptic media and video	Interstream	Collaborative work	–	–	–	Synchronization errors < around 240 to 320 ms: as the network delay of haptic media increase, the synchronization error becomes easier to be perceived
Tatematsu et al. [48]	Haptic media and 3D video	Interstream	Teleoperation system	–	–	–	Interstream synchronization quality is the highest when the 3D video and haptic media are output at the same time of the 3D video is output slightly ahead of haptic media
Zeng et al. [49]	Haptic media, sound and video	Interstream	Networked ensemble	–	–	(1) One media stream's network delay is different from those of the other two media stream: network delay of haptic: 10 ms, allowable range: 160 ms; network delay of haptic: 50 ms, allowable range: 80 ms (2) Three media streams are different from each other: 10 ms, allowable range: 120 ms; network delay of haptic; 50 ms, allowable range: 40 ms	

(continued)

Table 11.1 (continued)

Reference	Media	Synchronization	Work	Operation range	Imperceptible range	Allowable range	Others
Hoshino et al. [52]	Haptic media, CG, and olfactory	Interstream	Networked fruit harvesting game	–	From 500 to 1000 ms	From 110 to 1400 ms	
Ishibashi et al. [8]	Haptic media, CG, and olfactory	Group	Networked fruit harvesting game	–	–	–	(1) When the network delay at one terminal is 700 ms: synchronization errors from -200 to 200 ms, the fairness is high (2) When the network delay at one terminal is 1000 ms: other terminal is larger than 900 ms (smaller than 1500 ms in the experiment), the fairness is also high
Nakano et al. [53]	Haptic media, CG, and olfactory	Group	Networked balloon bursting game	–	–	From −150 to 150 ms	

delay at one terminal is 1000 ms and network delay at other terminal is larger than around 900 ms (smaller than 1500 ms in the experiment), the fairness is also high. In [53], the authors investigate the influence of network delay on the fairness between two players in a networked balloon bursting game with olfactory and haptic senses by QoE assessment. Experiment results show that the network delay differences (synchronization errors) from −150 to 150 ms are allowable.

From the above studies, we can see that there exists allowable range and imperceptible range of media synchronization errors, and the ranges depend on the media types. The human perception of media synchronization errors is summarized in Table 11.1.

11.8 Future Directions

In the future directions of media synchronization in networked multisensory applications with haptics, as we described above, it is important to clarify the human perception of media synchronization errors since most of the human perceptions are unclarified as shown in Table 11.1. Therefore, we can enhance the media synchronization control by using the human perception of media synchronization errors. Also, it is important to investigate the effects of the enhanced media synchronization control by implementing the control in different types of applications.

Furthermore, there is the mutually compensatory property among multiple media in [54] and [55]. Therefore, it is important to take the property into account for media synchronization.

11.9 Conclusions

In this chapter, we introduced the present status of studies for media synchronization in networked multisensory applications with haptics. We explained what haptic is, and introduced the applications and the media synchronization algorithms taking account of human perception in the applications. Furthermore, we enhanced simultaneous output-timing control by taking account of human perception. In addition, we made a survey of studies which investigate the human perception of media synchronization errors. Finally, we discussed the future direction of media synchronization in networked multisensory applications with haptics.

As our future direction, we need to clarify the human perception of media synchronization errors and enhance media synchronization control by taking account of the human perception. We also need to continue to make a survey of studies which focus on the media synchronization and clarify the human perception of different types of media in the future.

Definitions

Quality of Service (QoS) QoS is the quality of service which is provided from a layer to its upper layer in network layer model and it is defined as how much the service is faithful to the ideal situation.

Quality of Experience (QoE) QoE is also called the user-level QoS. QoE is the quality which is perceived subjectively and/or experienced objectively by end-users.

Human perception Human perception is the mental process in which human beings perceive or recognize an object or idea. In this chapter we quantitatively express human perception of synchronization error as ranges in which users can feel/perceive/allow the synchronization error.

References

1. Srinivasan, M.A., Basdogan, C.: Haptics in virtual environments: taxonomy, research status, and challenges. J. Comput. Graph. **21**(4), 393–404 (1997)
2. Kovacs, P.T., Murray, N., Rozinaj, G., Sulema, Y., Rybarova, R.: Application of immersive technologies for education: state of the art. In: Proceedings of 9th International Conference on Interactive Mobile Communication Technologies and Learning (IMCL), pp. 283–288, Nov 2015
3. Huang, P., Ishibashi, Y.: QoS control and QoE assessment in multi-sensory communications with haptics. IEICE Trans. Commun., Special Section on Quality of Communication Services Improving Quality of Life, **E96-B**(2), 392–403 (2013)
4. Ghinea, G., Ademoye, O.: Olfaction-enhanced multimedia: perspectives and challenges. Multimed. Tools Appl. **55**(3), 601–626 (2011)
5. Murray, N., Lee, B., Qiao, Y., Muntean, G.: Olfaction-enhanced multimedia: a survey application domains, displays, and research challenges. ACM Comput. Surv. **48**(4), 1–34 (2016)
6. Murray, N., Qiao, Y., Lee, B., Muntean, G.: Subjective evaluation of olfactory and visual media synchronization. In: Proceedings of ACM Multimedia Systems, pp. 162–171, Mar 2013
7. Sithu M., Arima, R., Ishibashi, Y.: Experiment on object recognition with haptic, olfactory, and audio senses in virtual environment. IEICE Technical Report, CQ2015, Mar 2016
8. Ishibashi, Y., Hoshino, S., Zeng, Q., Fukushima, N., Sugawara, S.: QoE assessment of fairness in networked game with olfaction: Influence of time it takes for smell to reach player. Springer's Multimed. Syst. J. (MMSJ), Special Issue on Network and Systems Support for Games **20**(5), 621–631 (2014)
9. Huang, P., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE assessment of olfactory media in remote ikebana with haptic media. In: Proceedings of IEEE 2012 International Communications Quality and Reliability (CQR) Workshop, May 2012
10. Sithu, M., Ishibashi, Y., Huang, P., Fukushima, N.: Influences of network delay on quality of experience for soft objects in networked real-time game with haptic sense. Int. J. Commun. Netw. Syst. Sci. (IJCNS) **8**(11), 440–455 (2015)

11. Sithu, M., Ishibashi, Y., Fukushima, N.: Effects of dynamic local lag control on sound synchronization and interactivity in joint musical performance. ITE Trans Media Technol. Appl., Special Section on Multimed. Transmission System and Services **2**(4), 299–309 (2014)
12. Gunn, C., Hutchins, M., Steveson, D., Adcock, M., Youngblood, P.: Using collaborative haptics in remote surgical training. In: Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 1–2, Mar 2005
13. Mizuno, H., Ishibashi, Y., Sugawara, S.: A comparison between transmission methods of sense of force in a remote haptic instruction system. In: Proceedings of IEEE ICME, pp. 1202–1205, June/July 2009
14. Zeng, Q., Iwata, K., Ishibashi, Y., Fukushima, N., Sugawara, S.: Influence of network delay in networked ensemble with haptic media, sound, and video. In: National Convention of IPSJ, 3Y-5, Mar 2012
15. Lun, M., Friedman, D., Sankaranarayanan, G., King, H., Fodero, K., Leuschke, R., Hannaford, B.: The RAVEN: design and validation of a telesurgery system. Int. J. Robot. Res. **1**(1), 1–16 (2009)
16. ITU E. 800, Terms and definitions related to quality of service and network performance including dependability, Aug 1994
17. ITU-T Rec. P. 10/G. 100 Amendment 1, New appendix I—definition of quality of experience (QoE). Int. Telecommun. Union, Jan 2007
18. Blakowski, G., Steinmetz, R.: A media synchronization survey: reference model, specification, and case studies. IEEE J. Sel. Areas Commun. **14**(1), 5–35 (1996)
19. Huang, Z., Nahrstedt, K., Steinmetz, R.: Evolution of temporal multimedia synchronization principles: a historical viewpoint. ACM Trans. Multimed. Comput. Commun. Appl. **9**(1), 34 (2013)
20. Sithu, M., Ishibashi, Y.: Media synchronization control in multimedia communication. IGI Global Emerg. Res. Netw. Multimed. Commun. Syst. Chapter **2**, 25–61 (2015)
21. Ishibashi, Y., Tasaka, S., Hasegawa, T.: The Virtual-Time Rendering algorithm for haptic media synchronization in networked virtual environments. In: Proceedings of the 16th International Workshop on Communications Quality and Reliability (CQR), pp. 213–217, May 2002
22. Ishibashi, Y., Kanbara, T., Tasaka, S.: Inter-stream synchronization between haptic media and voice in collaborative virtual environments. In: Proceedings of ACM Multimedia, pp. 604–611, Oct 2004
23. Sannomiya, H., Osada, J., Ishibashi, Y., Fukushima, N., Sugawara, S.: Inter-stream synchronization control with group synchronization algorithm. In: Proceedings of the 2nd IEEE Global Conference on Consumer Electronics (GCCE), pp. 520–524, Oct 2013
24. Ishibashi, Y., Tasaka, S.: A synchronization mechanism for continuous media in multimedia communications. In: Proceedings of IEEE INFOCOM, pp. 1010–1019, Apr 1995
25. Sithu, M., Ishibashi, Y., Fukushima, N.: Effects of dynamic local lag control on sound synchronization and interactivity in joint musical performance. ITE Trans Media Technol. Appl., Special Section on Multimedia Transmission System and Services **2**(4), 299–309 (2014)
26. Ishibashi, Y., Tasaka, S.: A distributed control scheme for group synchronization in multicast communications. In: Proceedings of International Symposium on Communications (ISCOM), pp. 317–323, Nov 1999
27. Ishibashi, Y., Hashimoto, Y., Ikedo, T., Sugawara, S.: Adaptive Δ-causality control with adaptive dead-reckoning in networked games. In: Proceedings of the 13th Annual Workshop on Network and Systems Support for Games (NetGames), pp. 75–80, Sep 2007
28. Marshall, A., Wai, Y.: Haptic virtual environment performance over IP networks: a case study. In: Proceedings of IEEE International Symposium on Distributed Simulation and Real-Time Applications, pp. 181–189, Oct 2003

29. El Saddik, A.: The potential of haptics technologies. IEEE Instrum. Meas. Mag. **10**(1), 10–17 (2007)
30. Lin, M., Salisbury, K.: Haptic rendering: beyond visual computing. IEEE Comput. Graph. Appl. **24**(2), 22–23 (2004)
31. Salisbury, K., Conti, F., Barbgli, F.: Haptic rendering: Introductory concepts. IEEE Comput. Graph. Appl. **24**(2), 24–32 (2004)
32. Weller, R., Zachmann, G.: User performance in complex bimanual haptic manipulation with 3DoFs vs. 6DoFs. In: Proceedings of IEEE Haptics, Mar 2012
33. http://geomagic.com/en/products/phantom-omni/overview
34. http://geomagic.com/en/products/phantom-desktop/overview
35. http://geomagic.com/en/products/phantom-premium/overview
36. http://www.forcedimension.com
37. Sato, M.: Development of string-based force display: SPIDAR. In: Proceedings of the 8th International Conference on Virtual System and Multi Media (VSMM), pp. 1034–1039, Sept 2002
38. Novint Technologies, Inc., Haptic device abstraction layer programmer's guide, version 1.1.9 Beta, Sept 2007
39. Gosselin, F., Riwan, A.: Design of virtuose 3D: a new haptic interface for teleoperation and virtual reality. In: Proceedings of ICAR, pp. 205–212, Aug 2001
40. Vanderlinde, R.Q., Lammertse, P., Frederiksen, E., Ruiter, B.: The hapticmaster: a new high-performance haptic interface. In: Proceedings of EuroHaptics, pp. 1–5 (2002)
41. Immersion Corporation: CyberGrasp. http://www.immersion.com/3d/products/cybergrasp.php
42. http://geomagic.com/en/products/sculpt/touch
43. Yoo,Y., Sung, M., Kim, N., Jun, K.: An experimental study on the performance of haptic data transmission in networked haptic collaboration. In: Proceedings of International Conference on Advanced Communication Technology, pp. 657–662, Feb 2007
44. Osada, J., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE comparison of haptic control schemes in remote instruction system with haptic media, video, and voice. In: Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), July 2012
45. Khademian, B., Hashtrudi-Zaad, K.: Performance issues in collaborative haptic training. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 3275–3263, Apr 2007
46. Willaert, B., Corteville, B., Reynaerts, D., Brussel, H., Poorten, E.: Bounded environment passivity of the classical position-force teleoperation controller. In: Proceedings of IEEE/RSJ IROS, Oct 2009
47. Fujimoto, T., Ishibashi, Y., Sugawara, S.: Influences of inter-stream synchronization error on collaborative work in haptic and visual environments. In: Proceedings of Haptics, pp. 113–119, Mar 2008
48. Tatematsu, A., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE assessment in tele-operation with 3D video and haptic media. In: Proceedings of IEEE ICME Workshop on Hot Topics in Multimedia, July 2011
49. Zeng, Q., Ishibashi, Y., Fukushima, N., Sugawara, S., Psannis, K.E.: Influences of inter-stream synchronization errors among haptic media, sound, and video on quality of experience in networked ensemble. In: Proceedings of IEEE GCCE, pp. 466–470, Oct 2013
50. Ishibashi, Y., Kaneoka, H.: Faieness among game players in networked haptic environments: influence of network latency. In: Proceedings of IEEE ICME, July 2005
51. Ishibashi, Y., Tasaka, S.: Causality and media synchronization control for networked multimedia games: centralized versus distributed. In: Proceedings of NetGames, pp. 42–51, May 2003

52. Hoshino, S., Ishibashi, Y., Fukushima, N., Sugawara, S.: QoE assessment in olfactory and haptic media transmission: Influence of inter-stream synchronization error. In: Proceedings of IEEE CQR, May 2011
53. Nakano, S., Maeda, Y., Ishibashi, Y., Fukushima, N., Huang, P., Psannis, K.E.: Influence of network delay on fairness between players in networked game with olfactory and haptic senses. IEICE Technical Report, CQ2014-94, Jan 2015
54. Tasaka, S. Ishibashi, Y.: Mutually compensatory property of multimedia QoS. In: Conference Record IEEE ICC, pp. 1105–1111, Apr/May 2002
55. Tasaka, S., Ito, Y.: Psychometric analysis of the mutually compensatory property of multimedia QoS. In: Conference Record IEEE ICC, pp. 1880–1886, May 2003

Chapter 12
Olfaction-Enhanced Multimedia Synchronization

Niall Murray, Gabriel-Miro Muntean, Yuansong Qiao and Brian Lee

Abstract This chapter introduces olfaction-enhanced multimedia synchronization and focuses on two key aspects: the specification of olfaction-enhanced multimedia, including the temporal relations between the media components; and secondly, the implementation of synchronized delivery of olfaction-enhanced multimedia. The relevance of this topic is supported by the fact that recently, multimedia researchers have begun to work with several new media components such as olfaction, haptic and gustation. The characteristics of these multisensory media differ significantly from traditional media. Multisensory media components cannot be classified as being continuous or discrete. Olfaction, the sense of smell, in particular, raises numerous research challenges. Synchronization, perceptual variability, sensor and display development are just some of the avenues among many others that require efforts from the research community. In terms of synchronization, implementing synchronized delivery as part of transmission across constrained networks is not the *key* research challenge (although adaptive mulsemedia delivery can play an important role here). Rather the principal problem, from a synchronization perspective, is understanding the experiential attributes of olfaction with respect to the temporal relations with other media components and the effect of these on the user's perceived Quality of Experience (QoE). This task is non-trivial. There are many facets unique to olfaction, which need to be understood in order to design and execute even the most basic of evaluations. In this chapter, we present and discuss the results of a subjective study which considered the above-mentioned "specification" and "implementation" challenges. In particular, we focus on analysing the user's ability to detect synchronization error and the resultant annoyance levels of synchronization error.

Keywords Olfaction · Multimedia · Synchronization · Quality of experience

N. Murray (✉) · G.-M. Muntean · Y. Qiao · B. Lee
Department of Electronics & Informatics, Athlone Institute of Technology,
Dublin Rd., Athlone, Co., Westmeath, Ireland
e-mail: nmurray@research.ait.ie

G.-M. Muntean
Department of Electronic Engineering, Dublin City University, Glasnevin, Dublin 9, Ireland

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_12

12.1 Introduction

Multimedia systems have generally transmitted various types of continuous and discrete media: audio, video, text and graphics [1]. These types of media components are now ubiquitous and are integrated as part of our daily lives as we both consume and produce such types of content. However, they have primarily focused on stimulation of the human senses of vision and audition. Sporadically over time and more recently, the research community has reported works that have stimulated other human senses: tactile [2], gustatory and olfaction [3]. Such efforts have been described with terms like: multi-modal media [4], sensory experiences [5], multiple sensorial media (mulsemedia) and multisensory media [6, 7].

Olfaction, the sense of smell, has been used along with other media components in the literature, as it is assumed that enhancing audiovisual content with scent will increase the viewer's comprehension and sense of reality. Applications of olfaction can also be found in domains such as gaming [8], health [9], education [10], training [11] and tourism [12]. Fundamental to any of these application areas is the synchronized delivery of media components from the user-perceived perspective. There are numerous suggested advantages of correctly presenting olfaction as part of a multisensory experience, such as: increased sense of presence, immersion and generally higher levels of user's perceived QoE. Claims have also been made with respect to benefits in terms of information recall [13] and as a form of therapy [9]. The purpose of this chapter is to highlight the potential of olfaction-enhanced multimedia experiences and present the existing findings in relation to synchronization evaluations.

12.1.1 Applications Domains for Olfaction-Enhanced Multimedia

This section introduces the reader to a number of different application domains where olfaction-based multisensory experiences can be of beneficial. It complements previous works [11, 14, 15, 16], which have highlighted the applications of olfaction-enhanced multimedia in areas such as health, education, tourism, entertainment, film and virtual reality (VR).

12.1.1.1 Olfaction in Entertainment, Film Industry, Gaming and Virtual Reality

Of all the domains where olfaction-enhanced multimedia experiences could be applied, a common belief is that first adopters will be in the areas of entertainment, gaming and film. Firstly, these genres have significant financial backing for exploring new avenues for revenue generation, but also these application domains

rely heavily on high degrees of user enjoyment. Hirota et al. in [12] reported on their work on multisensory theatre development. This multisensory theatre supported the sensory effects of: olfaction and wind as well as audiovisual media streams. Indeed, many franchises of the 4DX technology from CJ 4DPLEX [17], (a multisensory theatre system), are to be found across many cities around the world. The 4DX technology theatres support tactile, olfaction, mist and wind among many others.

More aligned to academic research and with respect to VR applications, [18] investigated user perception of smell in real and virtual environments. It highlighted the variable and perpetual nature of olfaction and the importance of considering the same. In [19, 20], the influence of audio, scent and temperature on viewer's ability to perceive the quality of graphics a virtual environment was investigated. They found significant impairment in the user's ability to detect degradations in visual quality due to the presence of multisensory components. Narumi et al. [21] evaluated a VR-enhanced multisensory system. Forty-four participants experienced six cookie appearances/scent combinations and were then queried in terms of their perception and ability to identify the flavour of a plain cookie when its appearance and scent were changed. In over 79% of the trials, a change in taste was reported. In [22], Covarrubias et al. presented an interactive and immersive multisensory VR system that could support upper-limb motor rehabilitation. As a form of interactive feedback, the system presented one of three odours (orange, chocolate and rose). The initial findings suggested such a multimodal system increased the users' sense of presence and attention.

12.1.1.2 Olfaction in Tourism

The area of digital heritage has gained significant interest in the last decade, and not surprisingly according to Hoven [23], olfaction has actually been incorporated into tourism experiences more than one might expect. One of the motivations for including multisensory components is to bring visitors beyond "browse mode". In the context of engagement and moving beyond the browse mode, the "Universal Scent Blackbox" reported in [24] provided a system that allowed visitors to experience pleasant and unpleasant scents as they moved between different city models in the museum. In [25], Hiroshi et al. also presented an interactive multisensory tourism experience which included visual, audition, tactile and olfaction. The aim of their system was to support multisensory experiences in addition to naturalistic interaction. They introduced the exciting olfactory display, the "micro-aroma-shooter", which has the benefit of being small yet still supporting "ejected" scent delivery. Fernstrom et al. [26] introduced the possibility of "hybrids" between arcades and museums allowing an interactive and multisensory experience. Addition of these multisensory components added value to the visitor experience. Tuan [27] presented the association between smells and locations and discussed how scent adds character making them easier to recall, whilst Porteous in [28] defined the term "smellscape", highlighting the relationship further between

places and smells. Furthermore, Hall et al. in [29] reflected that olfaction supports an "authenticity to an experience". In [30], Cheong et al. stated that future VR systems incorporating multisensory stimuli may be a "threat" to the traditional model of tourism. Dann et al. in [31] outlined and discussed the multisensory experiences at Walt Disney World. The use of olfactory data in other tourist sites, like the Guinness Storehouse and the bow street old Whiskey Distillery in Dublin, is discussed in [32].

12.1.1.3 Olfaction in Education and Training

Gardner's multiple intelligence theory proposed the idea of learning through various sensory channels and how such learning imitates learning in our natural environments [33]. The authors of [34] reviewed studies that indicate how "learning mechanisms operate optimally under multisensory conditions" concluding that multisensory experiences should be more effective than single modality learning as long as the congruency of the stimuli is consistent with real-world experiences. A similar theory was proposed by Haverkamp et al. in [35]. In terms of olfaction and memory, [36] proposed a basic apparatus for printed aromatic information. They employed a combination of scent and text to communicate information. They labelled this system "scented text". Brewster et al. in [4] evaluated any relationship between scent and information recall/memory. They developed a study that compared information recall via text and scent tagging. In a related study, Ademoye et al. [13] reported that users enjoyed the presence of scent, and that the presence of smell does not have any positive influence on ability to recall information. Tortell et al. [37] determined that scent has a positive effect on users' ability to recall the details of an environment. Herz et al. [38] looked at the influence of odours on mood and further discussed odour-assisted learning. As part of an initial study, Barros et al. analysed the impact of multisensory presentation on information processing and recall [39] and reported an improvement in performance via objective analysis.

Notwithstanding these works, the use of olfaction in education is still in its infancy relative to the other traditional modalities. Mikropoulos et al. [40] reviewed the use of education in virtual environments, but made reference to two works only with respect to olfaction: Rickard et al. [41] and Tijou et al. [42]. Tijou et al. described a VR application that investigated "the effect of olfaction on learning" and information retention [42]. Richard et al. [41, 43] described a VR platform that provides haptic, olfactory and auditory experiences. The authors stated that "odours activate the cerebellum, which is involved in motor learning". Dinh et al. evaluated the importance of olfaction among other modalities "on memory and sense of presence in virtual environments" [44]. Hughes et al. [45] included olfactory modality within their mixed reality training environments and applications. Ludvigson and Rottman [46] described and showed how the use of lavender and clove can improve cognitive processing. Mustonen [47] studied the effect of "sensory education" for food perception in terms of "taste and odour awareness in children",

Fig. 12.1 Cater fire training system [50]

with the result being an improvement in children's ability to describe sensory properties of food. Kwok et al. [48] described their "Smart Ambience for Affected Learning" (SAMAL) system, which uses olfactory data with a number of modalities to provide an effective evocative learning environment. The preliminary results suggest that the SAMAL system positively influences learning effectiveness. Childers et al. discussed the role of olfactory data in serious gaming, mental training and therapy in [49] and highlighted advantages including: reinforcement, real-life authentic learning, retention of information, improvement of concentration levels, promotion of independent thinking and familiarity. Cater reported a wearable multisensory experiences system in [50]. A backpack-mounted firefighter training device called advanced virtual environment real-time fire trainer (AVERT) delivered scents to the user through the oxygen mask; it also included heating affects.

As part of a collaborative learning environment that includes real, virtual and remote laboratory tools, Muller et al. proposed that "learners should be able to use multiple senses (e.g. visual, auditory, tangible, haptic and olfactory stimuli), when interacting with remote laboratory devices" [51]. Miyaura et al. [21] found "that presenting an odour when a user loses his/her concentration" is an effective method to decrease errors in addition tasks. Garcia-Ruiz et al. [52] reviewed works that integrated scent into VR applications during the task of learning a new language. For many years, the US military has been developing and using multisensory simulations [53]. They have investigated the use of olfaction dispensers (ODs) to provide the smell of "blood, cordite and other scents of the battlefield" (Fig. 12.1).

12.1.1.4 Olfaction in Health

Links between an ability to perceive olfactory stimuli and a number of neurodegenerative diseases exist in the literature: e.g. Alzheimer's [54], Huntington's [55]

and Parkinson's [56]. As part of medical training, doctors are trained to use their sense of smell in the recognition and diagnosis of disorders. Communication of information using senses other than vision and audition will be especially relevant to delivering high-quality experiences to those with vision-/audition-related disabilities [57, 58].

The authors of [59] used olfactory data as part of their work to "determine the characteristics of natural environments that are beneficial to humans". In [58], the use of vanilla fragrances was shown to reduce anxiety and distress. In addition, it enhanced the coping ability of patients who had undergone traumatic experiences. Baus and Bouchard [60] reviewed the literature on olfaction and outlined potential applications in VR environments from a healthcare perspective. Spencer discussed a number of olfactory and haptic training simulations [61] and recognized the educational benefits of exposing students to the various medically related odours associated with diverse disorders. In [62], Gerardi et al. used olfactory data delivered with a scent palette in conjunction with audio and visual stimuli to treat a soldier with post-traumatic stress disorder (PTSD).

Their findings indicate that following brief VR treatment, the veteran demonstrated improvement in PTSD symptoms. Rizzo et al. [63, 64] also used olfaction in addition to tactile, audio and visual to "create a more realistic multi-modal experience for the user to enhance the sense of presence" as part of VR exposure treatment to treat PTSD patients. Other examples document the use of VR to treat "PTSD in troops" [65, 66]. In [67], a detailed review of how VR applications were being used in military behavioural healthcare is provided. The use of sight, sound, touch and smell to "foster the anxiety modulation needed for therapeutic processing and habituation" was also discussed. Another work where "Virtual Reality Exposure Training" (VRET) was used is in [68], where the authors presented their rationale and description of their VR system to address PTSD therapy application (Virtual Iraq) and its use with active duty service members. Richard et al. [69] reported the design of an augmented reality (AR) application that uses olfactory as one of a number of media components to assist "children with cognitive disabilities". Borromeo et al. in [70] described a device to stimulate and scan olfactory areas as an initial step in the development of a clinical assessment tool for neurodegenerative and neurological premonitory symptoms. In the application area of home care reminders, [71] investigated the workload of responding to multisensory notifications, whilst carrying out a memory game.

Reflecting on the works discussed in this section, olfaction has a significant position to fill in the health, education and tourism industries as well as the more salient areas of TV, gaming and VR. The potential in each of these areas will drive the demand for enabling technologies in the form of sensors and ODs. This will potentially fuel the development of new applications across each of these disciplines as well as in areas such as film, alerting systems and entertainment. The next section introduces standardization efforts in the area of multisensory experiences, including olfaction-enhanced multimedia.

12.2 MPEG-V and the Standardization Efforts for Olfaction-Enhanced Multimedia

The ISO/IEC International Standard 23005, called "MPEG-V: Media Context and Control" is a seven-part standard which provides architecture and information representation to enable interoperability between virtual worlds [72–78]. In this standard, the term "sensory effects" describes what in this paper has been called multisensory experiences. As already mentioned, the MPEG-V standard states that addition of this multi-sensorial multimedia content leads to "even more realistic experiences in the consumption of audiovisual contents". MPEG-V contains two key areas: control information and sensory information. Control information is concerned with retrieving information from various sensory devices and also to control the presentation from various sensory devices. The sensory information area deals with the descriptions of the various sensory effects. To support these two areas, the standard comprises seven parts and describes: metadata to describe the various multisensory effects (through sensory effect metadata (SEM)) and commands to control the devices that can present the various types of multisensory content (audition, vision, olfaction, thermoception, mechanoreception, etc.); and various delivery formats. Considering these, what MPEG-V offers is a set of tools for enriching multimedia content with multisensory metadata.

The standard itself consists of the following parts under the general title of "Information technology—Media context and control":

- Architecture [72]: It describes the architecture and different instantiations of MPEG-V.
- Control Information [73]: It introduces the control information description language (CIDL) for controlling devices (sensors and displays).
- Sensory Information [74]: It presents the sensory effect description language (SEDL) and the sensory effect vocabulary (SEV) for describing sensory effects.
- Virtual World Object Characteristics [75]: It provides tools for describing the characteristics of virtual world objects.
- Data formats for Interaction Devices [76]: It describes the data format for exchanging information between interaction devices (e.g. controlling devices or retrieving information from sensors).
- Common Types and Tools [77]: It defines tools and common data types which are used by the different parts in common.
- Conformance and Reference Software [78]: It defines tools for checking the conformance of an MPEG-V description and for generating MPEG-V descriptions.

Figure 12.2 illustrates the MPEG-V architecture. Detailed information on the architecture of the standard can be obtained from the associated architecture document [72]. On top of the excerpt of the architecture, the digital content provider is located. This provider offers, for example, enriched multimedia content

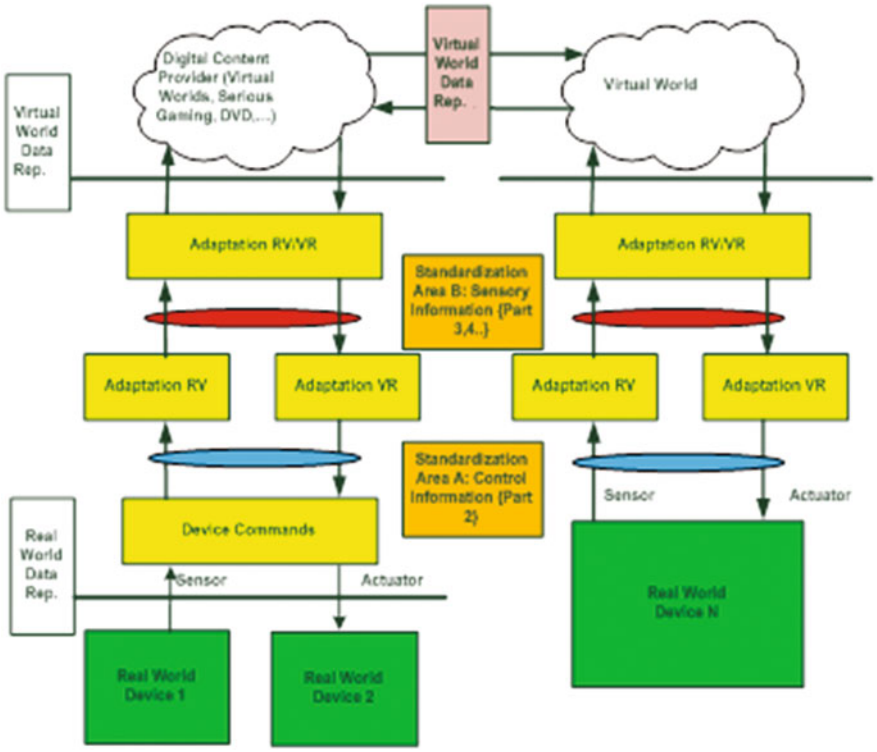


Fig. 12.2 MPEG-V part 1: architecture (Adapted from [72])

(e.g. broadcast, games, DVDs). This content can either be exchanged with other virtual worlds through the virtual world (VW) objects (i.e. Part 4) or the enriched content can be presented in the real world (RW) through actuators (e.g. vibration chairs, lamps, olfactory displays) and displays. The scope of Part 3 is the description of effects that can be rendered on the previously mentioned actuators. The resulting effects are transformed to commands for controlling the actuators (i.e. Part 2). The scope of MPEG-V Part 2 covers the interfaces between the adaptation engine and the capability descriptions of actuators/sensors.

The user's sensory preferences, sensory device capabilities and sensor capabilities are within the scope of this part of MPEG-V. The control information includes user's sensory preference information, device capability description and sensor capability description. These can be used to fine-tune the sensed information and the device command for the control of virtual/real world by providing extra information to the adaptation engine.

The key aspects of Part 2 [73] are as follows:

- The Control Information Description Language (CIDL) "provides a basic structure of the description of control information".
- The Device Capability Description Vocabulary (DCDV) "is an interface to describe functionality of sensory devices".
- The Sensor Capability Description Vocabulary (SCDV) "is an interface to describe the functionality of various sensors".
- The Sensory Effect Preference Vocabulary (SEPV) "is an interface for describing preference of individual users towards individual sensorial media types".

Figure 12.3 reflects the scope of MPEG-V: Part 3. On the left side, one can see the content provider or source (e.g. a DVD, Blu-ray Disc or the Internet). From the source, the traditional audio/visual content is streamed or offered to the consumer. In addition, the so-called SEM description can be sent to the consumer. At the consumer side, there is a media processing engine (MPE), which handles both the multimedia content and the SEM description. The MPE parses the SEM description and activates MPEG-V capable devices, such as vibration chairs, lamps, olfactory displays, haptic devices in a synchronized manner with the audiovisual media components.

SEDL is defined in Part 3. It provides the tools for describing sensory effects (e.g. wind, vibration, light, scent, haptic feedback, thermoception). The actual sensory effects are not provided via SEDL. They are defined by SEV. MPEG-V provides a separate SEV schema to allow for extensibility and flexibility. This allows application domains to define their own sensory effects. A description conforming to SEDL (and implicitly to SEV) is called SEM description. The SEM description can accompany any kind of multimedia data (e.g. movies, music, games). MPEG-V capable processing engines (e.g. a set-top box) can use the SEM description to steer appropriate sensory display interfaces (e.g. olfactory displays). The current version of SEDL is specified in [74].

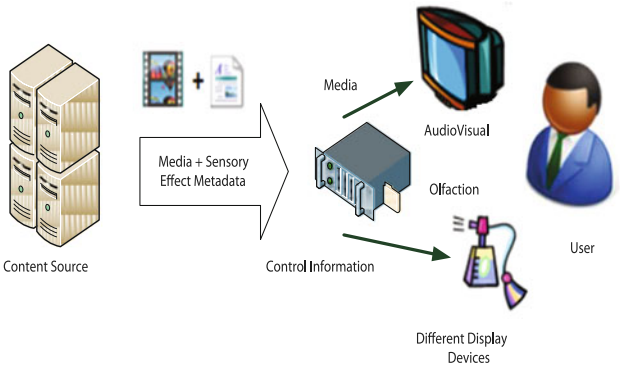


Fig. 12.3 Example of MPEG-V: part 3 (Adapted from [74])

SEV defines the sensory effects as opposed to SEDL, which defines the basic structure of the SEM description. SEV is also XML-based and can be extended with additional effects not yet defined. The mapping of annotated sensory effects to the actual devices is performed by the media processing engine. The first edition of the MPEG-V standard reference software [78] supports the following 15 sensory effects: light, flashlight, temperature, wind, vibration, water sprayer, scent, fog, colour correction, tactile and kinaesthetic.

12.2.1 Multisensory Multimedia Synchronisation Based on MPEG-V

Much of the work specific to olfactory media synchronization falls under perceived or subjective synchronization. In terms of analysing works that have looked at the streaming of sensorial media, just a few works are available in the literature. Many of these works have emerged based on the MPEG-V standard. Choi et al. [79] introduced Single Media Multiple Devices (SMMD) media controller, which processes MPEG-V SEM descriptions and controls various sensory devices. Pyo et al. [80] described an earlier version of the SMMD controller but takes also the context of Universal Plug and Play (UPnP) into account. Yoon et al. in [81] introduced a framework for broadcasting of sensory effects based on MPEG-V. The authors called it 4-D broadcasting. This framework mainly focused on SEM delivery via the MPEG-2 transport stream and its decoding at the destination. Additionally, in [82], another broadcasting system was introduced for streaming additional sensory effects together with the audio/video content.

Interestingly given the scope of this research, Yun et al. [83] introduced a synchronization algorithm for multimedia content accompanied by sensory media. In the paper, the authors presented evaluation results for their algorithm. They defined a number of time points in their model, so that a global time between all the devices in the mulsemedia presentation exists. The media start time *M(t)* and current system time *C(t)* were used to define the current media time *MC(t)* for the player. Every second, each player stores the time gap (C(t) − P(t)), which is calculated from the time difference between the current received message event C(t) and the previous one P(t), and delay media play time MC(t) with the time gap C(t) − P(t). *MC(t)* for the player can be calculated by Eq. (12.1).

$$MC(t) = M(t) + C(t) - P(t) \quad [80] \tag{12.1}$$

To ensure synchronization, the effect needs to be delivered to the user as the video objects are presented on screen. Considering this, the effect device should be activated such that the effect is delivered as the scene is played. Hence, an algorithm which calculates the device activation time *D(t)* is needed. The devices presenting sensorial media, e.g. olfactory display, need to be activated ahead of presentation time by subtracting device execution time $\delta(t)$ and network delay *N(t)*. In this case, we can use Eqs. (12.2)–(12.4). Figure 12.4 illustrates the effect device synchronization algorithm. More details for the interested reader are available in [83]

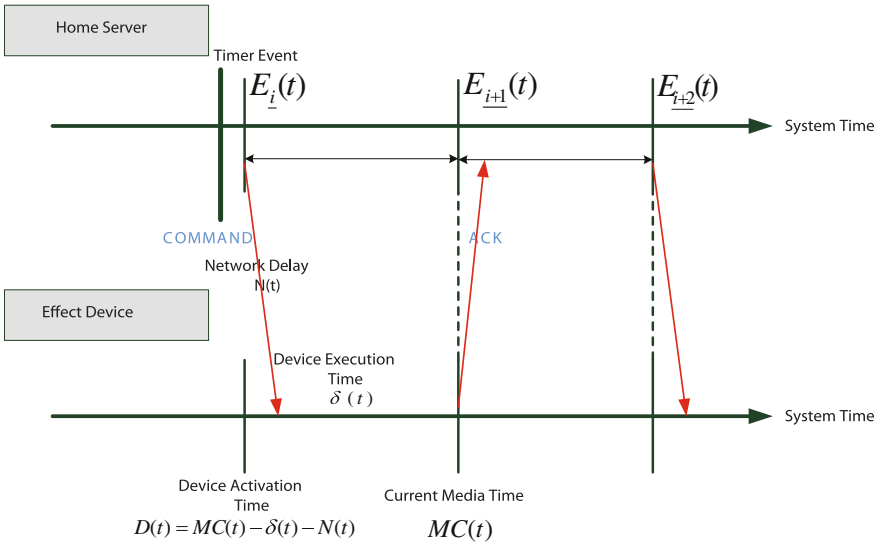


Fig. 12.4 Sensory effect device synchronization algorithm [83]

$$D(t) = MC(t) - \delta(t) - N(t) \text{ [80]} \tag{12.2}$$

$$\delta(t) = \frac{1}{n}\sum_{i=1}^{n} \delta_i(t) \text{ [80]} \tag{12.3}$$

$$N(t) = \frac{1}{k}\sum_{k=1}^{k} N_i(t) \text{ [80]} \tag{12.4}$$

In order to deliver synchronized multisensory media experiences, an algorithm is required which delivers multimedia content enriched by multisensory media with the timing relations intact, or such that any errors are not noticeable to the user. In this way, any synchronization problems for olfaction-enhanced multisensory multimedia do not have an adverse effect on user QoE. In the next section, olfaction-based mulsemedia delivery will be discussed in terms of architecture and algorithm.

12.3 Olfaction-Based Mulsemedia Synchronization: Implementation and Specification

In this section, the authors highlight some findings from their own works in relation to the implementation and specification of olfaction-based mulsemedia synchronization.

12.3.1 Olfaction-Based Mulsemedia Synchronization: Implementation

In the context of an increasing amount of data traffic, the communication networks are often subject to very high and variable loads. These affect the quality of the delivered multimedia content. Existing delivery approaches cannot cope with these highly variable situations, and adaptation solutions have been considered to be employed. These solutions perform multimedia content adjustments dynamically [84] to match the transferred content bitrate to the available bandwidth and decrease the loss rate. Despite the adaptation efforts, the reduction in encoding multimedia quality is observed and the end-user QoE decreases. However, mulsemedia perceptual tests described in [85] have shown that in the presence of additional sensorial inputs, the overall user QoE is higher than in their absence during adaptive multimedia content delivery. Consequently, this section describes an adaptive mulsemedia delivery system architecture for end-user QoE enhancement, which considers multi-sensorial content in the network-based content delivery adaptation process. The block-level architecture presented in Fig. 12.5 generalizes the adaptive mulsemedia framework (ADAMS) architecture proposed in [85] and enables the use of MPEG-V.

The architecture involves a feedback-based client–server approach and several components (blocks) which help enable adaptive mulsemedia content delivery, including *Packet Priority Scheduling* and *Mulsemedia Flow Adaptation* at the server and *Client Monitoring Unit* and *User Profile* at the client. During the content delivery sessions, the server exchanges multi-sensorial data with the client, which in turn passes feedback information back to the server. Specific adaptive delivery-related mulsemedia information processing is performed in the coloured blocks, whereas the other blocks employ already existing solutions.

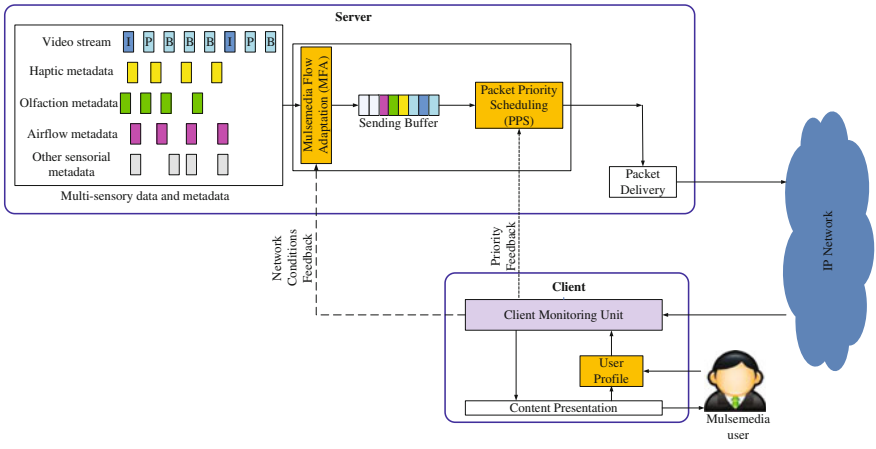


Fig. 12.5 Block-level architecture of an adaptive mulsemedia delivery system

The **server** is composed of four major blocks. The *Adaptation Module* gets regular feedback information from the client, and based on the received feedback, it takes multi-sensorial media adaptation decisions according to the adaptation algorithm. The adaptation algorithm is implemented in two sub-modules: **Mulsemedia Flow Adaptation (MFA)** and **Packet Priority Scheduling (PPS)**. The multi-sensorial data and metadata block stores the relevant content and associated information in order to be able to perform the delivery. The delivery to the client is performed by the *Packet Delivery* Unit.

The *MFA* module provides flow-based coarse-grained adaptation which transmits proper multi-sensorial content and performs video content transcoding if required, according to client feedback. The feedback includes both network conditions and user profile (i.e. priority level of sensorial effects). The network conditions are indicated using off-the-shelf bandwidth estimation techniques, such as the model-based bandwidth estimation (MBE), introduced in our previous paper [86]. MBE computes the estimated bandwidth using the following parameters: number of mobile stations, packet loss and packet size. Equation (12.5) gives the computation of estimated available bandwidth (B_A) for TCP flows based on MBE. The parameter b is the number of packets acknowledged by a received ACK, P_{retr} denotes the probability of packet retransmission, *MRTT* is the transport layer round-trip time between sender and receiver, and *MSS* is the maximum segment size. T_o is the timeout value used by the congestion control. The estimated available bandwidth for UDP flows used in this paper is also given in [86].

$$B_A = \frac{MSS}{MRTT \times \sqrt{\frac{2bP_{retr}}{3}} + T_o \times \min\left(1, 3\sqrt{\frac{3bP_{retr}}{8}}\right) \times P_{retr} \times \left(1 + 32P_{retr}^2\right)} \tag{12.5}$$

User profiles are configured and updated by the client in the *User Profile* unit.

MFA can deploy any adaptive solution. As per [86], it involves three states to perform mulsemedia flow adaptation. B_{MS}, B_{sense} and B_{video} represent the bitrate of mulsemedia flow, sensorial data flow and video flow, respectively. B_{MS} is defined in Eq. (12.6).

$$B_{MS} = B_{sense} + B_{video} \tag{12.6}$$

MFA maintains a *state* parameter to dynamically control the *MFA* process according to network conditions. Three states are considered in the design of *MFA*, as illustrated in Fig. 12.6:

(1) The first state (*State 1*) is active if $B_{MS} \leq B_A$. *State 1* indicates that the available bandwidth (B_A) is enough to deliver both video and sensorial data flows and there is no need to perform any content quality adaptation.
(2) The second state (*State 2*) is active if $B_{sense} \leq B_A \leq B_{MS}$ and $B_A \geq B_{video}^{min}$ where B_{video}^{min} is the bandwidth threshold associated with good video quality level. In *State 2,* the available bandwidth is between the bitrate of the sensorial data flow and the bitrate of the video flow, and therefore, the video flow is adapted

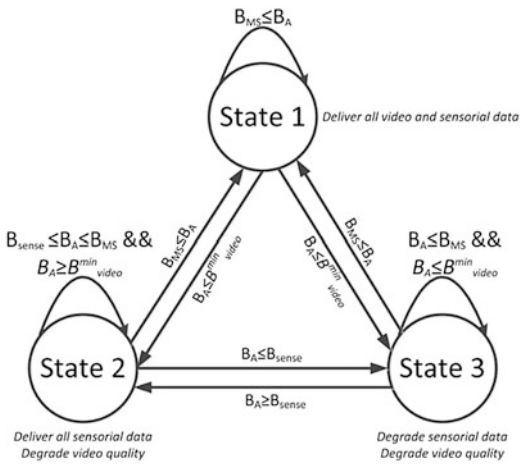


Fig. 12.6 States transition of the MFA module

(i.e. involves quality reduction and therefore bitrate decrease) whilst all the sensorial data flows are still transmitted. ADAMS adjusts the video bitrate to meet the available network bandwidth following the feedback reports. This is based on an additive increase–multiplicative decrease policy and on N granularity quality levels defined in inverse order of video quality. Each such quality level is defined in terms of a triplet <*resolution, frame rate, colour depth*>, directly related to a video bitrate value. When increased traffic in the network affects the client feedback scores, ADAMS switches fast to a lower quality level and accordingly adjusts the values of some of the triplet's components. This action results in a reduction in the bitrate of the video sent, easing the pressure on the network and helping it to recover from congestion. This eventually determines lower loss rates and consequently better end-user-perceived quality. In improved delivery conditions, as reported in terms of the client feedback scores, ADAMS cautiously and gradually increases the transmitted video quality level and, therefore, improves the values of some of the triplet's components. In the absence of loss, this determines an increase in end-user-perceived quality.

(3) The third state (*State 3*) is active if $B_A \leq B_{MS}$ and $B_A \leq B^{min}_{video}$. *State 3* indicates that the available bandwidth has reached very low values, and therefore, the video flow is degraded as indicated in *State 2*. Additionally, following delivery quality feedback reports, ADAMS removes sensorial media components from the mulsemedia stream, in inverse order of user interest in their corresponding sensorial effects. This decision is taken based on user profile information if it includes user preference for some sensorial media objects, or explicit user feedback. When such information is not available, a default preference order is assumed. The results reported in [86] have shown a definite preference of the test subjects for haptic, air motion and olfaction effects, respectively, in this order.

The *PPS* module provides packet-based fine-grained adaptation using a priority model which specifies that packets with higher priority are scheduled earlier than those with lower priority. Any priority model can be employed, including one derived based on the results of the subjective tests described in [86]. It was hypothesized that a lower quality video sequence integrated with mulsemedia effects is capable of producing as good a user experience as that of a higher quality video sequence.

Let W_v, W_h, W_o and W_a denote the weight factors associated with the priority levels of video, haptic, olfaction and airflow data packets, respectively. According to results from the subjective tests [86], on average 63%, 31% and 6% of users prefer haptic, airflow and olfaction sensorial effects, respectively. The importance or priority of each sensorial effect is normalized according to the ratio in Eq. (12.7). This might not be the perfect model for the priority levels of these sensorial data packets, but it initializes the mulsemedia adaptive system using low complexity computation and based on the average opinions of the subjects tested. To the best of our knowledge, this is the first equation that models the relationship between haptic, olfaction and airflow in terms of human preferences and is incorporated in the ADAMS adaptation strategy. Future work will extend Eq. (12.7) to improve the solution in terms of flexibility and scalability.

$$W_h : W_a : W_o = 0.63 : 0.31 : 0.06 \tag{12.7}$$

Additionally, the subjective tests show that the user enjoyment levels were maintained high when lower multimedia quality sequences were used in conjunction with mulsemedia effects [86]. Naturally, we assign sensorial data packets an equal or higher priority level than that of the video packets in terms of the user-perceived experience. Based on these subjective tests results, it can be concluded that sensorial data packets have equal or higher priority level (in terms of the impact on user perception) than that of the video packets. Therefore, Eq. (12.8) is derived to describe the priority relationship between these sensorial data packets.

$$\{W_h, W_o, W_a\}_{\min} \geq W_v \tag{12.8}$$

Equation (12.8) is a general approximation of the priority model between sensorial packets (i.e. haptic, olfaction, air flowing) and video packets. In order to obtain the initial values of the weighted factors for different packet types, it is assumed that olfaction packets have the same priority as the video packets, which results in W_o equals W_v. This assumption is supported by the fact that, in terms of user perception, olfaction data have lower priority than both haptic and airflow data. According to Eq. (12.8), by normalization, W_h, W_a, W_o *and* W_v values are 0.595, 0.293, 0.056 and 0.056, respectively.

The probability of scheduling the next packet in the queue is computed by Eq. (12.9), which takes into account both packet priority and flow bitrate. Parameters i and j refer to the ith packet of flow j in the queue, and N is the number of queued packets. $Bitrate_j$ denotes the bitrate of the jth flow. The value of packet

weight factor W_i^j is set based on the packet type (i.e. video, haptic, olfaction, airflow). For instance, if the *i*th packet is a haptic packet, then W_i^j equals W_h which is 0.293 according to the previously described default configuration.

$$P_i = \frac{W_i^j \times Bitrate_j}{\sum_i^N W_i^j \times Bitrate_j} \tag{12.9}$$

The **client** consists of three major blocks. When it receives the multi-sensorial content via the network, the multi-sensorial media components are passed to the *Content Presentation* unit which performs synchronized presentation of the various content items. Apart from the regular screen and speakers necessary to present multimedia content, this unit makes use of various devices, such as haptic vests, fans, smell-releasing devices, heaters for presentation of other sensorial effects. The client maintains a *User Profile* in order to enable both automatic feedback gathering and explicit feedback (if users desire to provide) in terms of user multi-sensorial adaptive preferences. The performance of network delivery is assessed by the *Client Monitoring Unit*, which has to map QoS-related parameters such as loss, delay and jitter and their variations and estimations of viewer-perceived quality on application level scores that describe the quality of the delivery session. This delivery quality is monitored over both short-term and long-term. Short-term monitoring is important for learning quickly about transient effects, such as sudden traffic changes, and for quickly reacting to them. Long-term variations are monitored in order to track slow changes in the overall delivery environment, such as new flows over the network. These short-term and long-term periods are set to be an order and two orders of magnitude (respectively) greater than the feedback-reporting interval. The Quality-Oriented Adaptive Scheme (QOAS) [85] describes an instantiation of such a *Client Monitoring Unit* which uses 100 ms inter-feedback intervals and short- and long-term monitoring periods of 1 s and 10 s, respectively.

In the next section, we discuss the experimental testing involving olfaction-enhanced multimedia content performed in order to assess the influence of synchronization on user-perceived quality.

12.3.2 Olfaction-Enhanced Multimedia Synchronization: Evaluation

In this section, we present some results of experimental evaluations performed to evaluate user ability to detect skew and their perception of skew. In this empirical evaluation of olfaction-enhanced multimedia clips, audiovisual media was enhanced with two olfactory components (i.e. the audiovisual media was presented with two different scents). We provide information on the olfaction-enhanced multimedia presentation system, laboratory design, assessors as well as video and scents used.

Rationale for Experimental Design

In terms of existing studies that have considered user perception of olfaction-enhanced multimedia synchronization, the majority of works have examined inter-stream synchronization between one scent and one video scene [14, 15, 87–93]. This represents conceptual network delay scenarios. These works defined temporal boundaries between the video and olfactory components based on: the user ability to detect skew and the annoyance levels associated with skew. Other research has focused on intra-stream synchronization of olfactory components. Okada [94, 95] and his team developed a scent emitter based on inkjet technology. In this olfactory display system, scented oil is stored in scent chamber. Tiny amounts of scent can be placed on a heated plate for vaporization and are presented via miniature fans in the device. From a synchronization perspective, this type of device can support directed and controlled amount of scents. In terms of evaluation of this device, a number of studies are available in the literature. Ohtsu et al. [94] report an evaluation of this device which controls emission to be synchronized with assessor breathing pattern, based on pulse ejection whilst another related study [95], considered the presentation of the minimum amounts of scent required, thus address lingering effects.

Here, we investigate how users perceive inter-stream synchronization between video and two scents. This work was reported in detail in [96]. The scenarios evaluated represent conceptual network delay and conceptual network jitter. We provide analysis on the user ability to detect and user's perception of skew in an olfaction-enhanced multimedia experience. These (detection and perception of skew) were identified as two critical considerations as discussed in [97].

Olfactory-Enhanced Video Presentation Equipment

Figure 12.7 illustrates the olfactory and video display system used which consists of the SBi4—radio v2 scent emitter (item Y sitting on the laptop) from Exhalia [98]. As per Fig. 12.8, the olfactory display can support four scent cartridges simultaneously. It presents scents by blowing air (using four inbuilt fans) through the scent cartridges. The SBi4 system is controlled using the Exhalia Java-based SDK. It is connected to the laptop via a USB port. The video content was played using the VLC

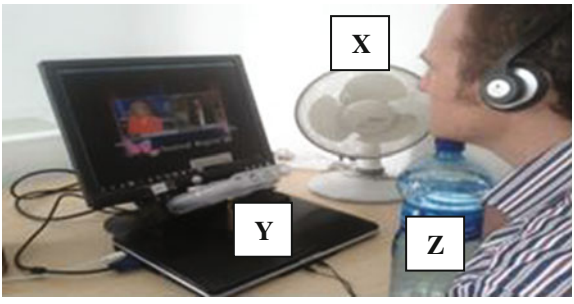


Fig. 12.7 Olfactory and video media display system [87]

Fig. 12.8 SBi4 V2, scent cartridges and bespoke extension [96]

media player 1.0.1 Goldeneye. A special control program was developed that controlled the synchronized presentation of olfactory data and video, including the introduction of artificial skews between the various media components presented in diverse step sizes. Figure 12.8 shows the SBi4, scent cartridges and the bespoke extension that was designed and added to the SBi4 as shown in Fig. 12.7 (Item Y). The fan (Item X) was turned on between participant trials (as they were completing the questionnaires) and also between assessments to remove any lingering scent. The purpose of this extension was to facilitate an accurate presentation of the scent to the users' olfactory field as opposed to a more general presentation. With the SBi4 positioned 0.5 m from the assessor, it was determined that it took assessors between 2.7 and 3.2 s to detect the scents as per Fig. 12.11. The laptop had an Intel Core™ 2 Duo CPU @ 1.66 GHz, 2 GB RAM and ran the Windows 7 professional operating system. In addition, Fig. 12.7 also presents a bottle of water (item Z) that the assessors placed under their chin during testing to ensure consistency across all assessors in terms of the location of their olfactory fields regardless of posture or physical size. Whilst the authors acknowledge the limitations of this configuration, we found it worked quite well during the experiments. This said, a more precise and stable configuration is presented in [95] for the interested reader.

Assessors, Screening and Training

A total of 100 assessors took part in this study (split into two groups: group 1 and group 2 with 50 participants in each). This group included assessors between the ages of 19–60 years, with an even distribution across the age range and gender. The assessors were from a wide variety of backgrounds. In order to be eligible, assessors could not be involved in any sensory analysis testing in the 24 h preceding the tests. In an attempt to provide contamination-free results, assessors must not have been affected by cold or flu and must avoid wearing perfume, aftershave or scented deodorants on the day of the testing. In addition, they were requested to avoid chewing gum, eating food, drinking tea or coffee in the 30 min prior to the test.

Assessors were also screened for anosmia as per ISO standard 5496:2006 [99]. Anosmia is an olfactory-related phenomenon, whereby there exists a lack of

sensitivity to olfactory stimuli. It can be total, partial, permanent or temporary. The purpose of this standard is to provide a methodology for the initiation and training of assessors in the detection and recognition of odours. The aim is to teach assessors to (1) evaluate, (2) identify odours and (3) use appropriate vocabulary and improve their aptitude. We have incorporated this standard for two key reasons (a) to address the key aims of the standard outlined above, but most importantly (b) to identify assessors who could have anosmia.

As part of the training phase, if assessors could detect, but were unable to correctly identify the scent presented, the assessors were given the name of the odour thereafter for familiarization purposes. As part of this screening, four potential assessors were not selected for participation in the tests reported below based on suspicion of partial anosmia. A detailed tutorial on the execution of testing involving olfaction-enhanced multimedia is available in [16].

The first group of 50 assessors (group 1) experienced synchronized and skew levels for scent A and scent B in step sizes of 5 s as per Fig. 12.9. These skew levels reflect conceptual delay. Once the presentation of the olfactory media was complete, a SBi4 fan with a non-odour cartridge was turned on to address scent lingering. Figure 12.9 shows how olfactory media is presented at different times relative to the video time axis. For olfactory media to be in sync (0 s skew) with the video, scent A should be presented for the middle 30–54 s block and scent B should be presented for the 60–84 s block as shown in Fig. 12.9. The six-second "non-odour" period is based on a recommendation that the time between presentations of two consecutive scents should be greater than 5 s [100]. Olfactory data before video content are represented by skew times of −20, −15, −10 and −5 s, and olfactory data after video content are represented by skews of +5, +10, +15 and +20 s, as per [96].

The second group of 50 assessors (group 2) experienced the set of skews as per Table 12.1. As shown, these test scenarios reflect typical packet jitter, i.e. variable latency across a network. Within this set of tests, there are two primary scenarios.

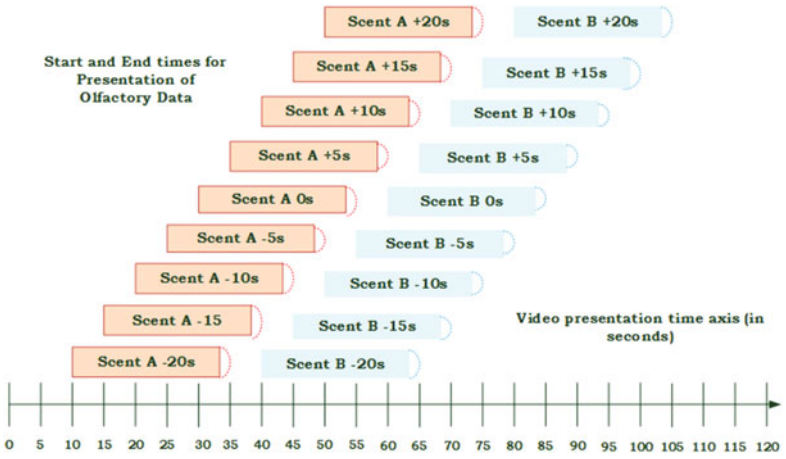


Fig. 12.9 Start and stop times for scent A and scent B and associated video presentation time for test group [96]

Table 12.1 Test group 2 (jitter scenarios): case 1 applies to participants 1, 7, etc., case 2 applies to participants 2, 8, etc. [96]

Case	Clip 1 skew fruit/ flower	Clip 2 skew forest/ burnt	Clip 3 skew fruit/ rubbish	Clip 4 skew rubbish/ burnt	Clip 5 skew orange/ chocolate	Clip 6 skew horse stable/ grass	Clip 7 skew forest/ sea water	Clip 8 skew grass/sea water
1	0 s/0 s	−15 s/ 0 s	**−10 s/ −20 s**	−5 s/ 0 s	**0 s/ −20 s**	**+5 s/ −10 s**	+10 s/ + 20 s	−20 s/ −10 s
2	+15 s/ −15 s	0 s/0 s	**0 s/ −10 s**	−20 s/ + 10 s	−15 s/ + 15 s	**+10 s/ −10 s**	−5 s/ + 5 s	**+20 s/ −10 s**
3	**+10 s/ −20 s**	**+15 s/ 0 s**	0 s/0 s	0 s/ + 10 s	**+5 s/ −5 s**	0 s/+ 5 s	−10 s/ + 10 s	**+20 s/ −20 s**
4	**+10 s/ −20 s**	0 s/ + 15 s	−5 s/ −10 s	0 s/0 s	+5 s/0 s	−10 s/ + 20 s	**+15 s/ + 5 s**	0 s/ + 20 s
5	−20 s/ 0 s	**0 s/ −15 s**	−15 s/ −10 s	**−5 s/ −20 s**	0 s/0 s	−15 s/ −5 s	+5 s/ + 15 s	0 s/−5 s
6	**+20 s/ + 10 s**	−15 s/ + 5 s	−20 s/ + 20 s	**+15 s/ −5 s**	−10 s/ 0 s	0 s/0 s	**−5 s/ −20 s**	**+10 s/ 0 s**

The variance of delay can result in either (1) the gap between the presentations of the two olfactory streams being extended beyond 6 s or (2) the two olfactory streams can overlap. Considering the latter case, for typical media transmission such as audio, the simple result is to "drop" packets if overlap occurs. However, considering that in these set of tests we are dealing with olfaction, in certain cases it may actually make sense if two scents "mix". Hence, the set of tests experienced for the group 2 reflect scenarios where both scent A and scent B are presented early and late and many of the possibilities in between. For clarity in Table 12.1, the mixing scenarios are highlighted in bold.

Laboratory Design

The design of the test laboratory is in accordance with ISO standard ISO/IEC 8589 [101], on "Sensory analysis—General guidance for the design of test rooms". The aim of this standard is to design test rooms such that it is possible: (1) to conduct sensory evaluations under known and controlled conditions with minimum distraction; and (2) to reduce the effects that psychological factors and physical conditions can have on human judgment. The laboratory design included: a testing area in which work may be carried out individually in testing booths; a preparation area and storage room. Walls in the test room are matt off-white. This size of the room and positioning allowed scents to diffuse between tests, minimized adaptation and gave assessors a break between each judgement. The laboratory is described in detail in [90, 96].

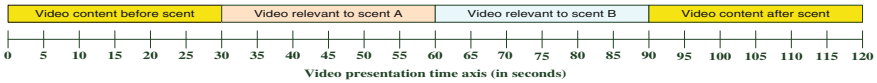


Fig. 12.10 Breakdown of video content into four key segments [96]

Video Sequences, Scents

Eight videos of 120 s duration were used. Each of the video clips can be divided into four 30 s blocks, whereby the two middle 30 s block contains content related specifically to the scent being presented as per Fig. 12.10. The clips are in the form of documentaries, cookery programs and movies and were chosen and altered such that the two middle 30 s segments corresponded to the content relating to the olfactory media. These clips were also chosen as they contain a balance of video content that reflects a mix of pleasant and unpleasant smells and combinations thereof. Ten scents in total were used in the testing, complying with [99] in terms of the number of scents that should be used in subjective tests. The scents of fruit, flowery, forest, burnt, orange, rubbish, chocolate, horse stable, grass and sea water. Descriptions of the scent combinations are provided in Tables 12.1 and 12.2. Scent A was always the first scent presented, and scent B was always the second scent presented. The scents outlined above were stored in sealable plastic bags and kept in a cool box at approximately 5 °C as per recommendations of [101].

Assessment Methodology and Questions

On arrival, assessors were provided with an information sheet on the tests. Any questions were addressed, and assessors were required to sign a consent form. Screening as outlined above was performed for all participants. Following the screening, assessors were asked to review the questionnaire they would answer on each olfaction-enhanced multimedia test clip. Assessors were asked to engage for the duration of each test sequence. On completion of the tests, windows in the room were opened and the fan was turned to remove any lingering scents. There was always a minimum of 15 min between consecutive executions of tests between assessors. This gave ample time for removal of any lingering scents, collection of questionnaire sheets and preparation for subsequent assessor testing. It also included time for the new assessor to read and sign consent forms, ask questions, etc. The entire testing time for a single subject was approximately 65 min. This comprised approximately 350 s per test sequence (i.e. reference sample, break, sample under test and voting). At the mid-point of the test, assessors were given a 15 min break to address any concerns over olfactory adaptation or fatigue. Assessors were permitted to drink water during the voting time periods and during the 15 min break [96, 16].

Assessors in group 1 and group 2 reported on their ability to detect skew and their perception of skew as per [102] and Table 12.3. Statements 1 and 2 aimed to

Table 12.2 Video categories and scents used [96]

Scent category	Fruit/flower	Forest/ burnt	Fruit/rubbish	Rotting/ burnt	Orange/chocolate	Horse stable/ grass	Forest/ sea water	Grass/sea water
Clip no:	Clip 1	Clip 2	Clip 3	Clip 4	Clip 5	Clip 6	Clip 7	Clip 8
Video description	Documentary on flower garden and orchards	Scene from Avatar Movie	Documentary about rotting fruit cocktail	Scene from Lord of the Rings Movie	Cookery documentary: chocolate orange biscuits	Documentary about horse stable cleaning	Scene from Avatar Movie	Documentary on grass plant and Sea life

Table 12.3 Questions, statements, responses and scales for group 1 and group 2 [96]

Statement/question #	Response options	Score
(1) Relative to the video content in the clip, smell "A" was released	Too late	5
	Late	4
	Neither early or late	3
	Early	2
	Too early	1
(2) Relative to the video content in the clip, smell "B" was released	Too late	5
	Late	4
	Neither early or late	3
	Early	2
	Too early	1
(3) In the event that you may have perceived the video clip and smell "A" being out of sync, please indicate the extent to which it impacted upon you?	Imperceptible	5
	Perceptible but not annoying	4
	Slightly annoying	3
	Annoying	2
	Very annoying	1
(4) In the event that you may have perceived the video clip and smell "B" being out of sync, please indicate the extent to which it impacted upon you?	Imperceptible	5
	Perceptible but not annoying	4
	Slightly annoying	3
	Annoying	2
	Very annoying	1

determine assessor ability to **detect** the existence of a synchronization error, for scent A and B, respectively. Assessors answered by selecting one of the five possible answers: Too Early, Early, Correct Time, Late or Too Late. Questions 3 and 4 aimed to determine how **tolerant** assessors were to different levels of skew for scent A and scent B. Hence, they were asked to qualify their annoyance of the inter-media skew as per: imperceptible, perceptible but not annoying, slightly annoying, annoying, very annoying.

Experimental Results

This section reports the results of the tests, presenting analysis on the assessor's ability to detect skew and also how they rated the presence of the same skew for both scents in terms of annoyance. In order to ensure that the scents arrived at the user's olfactory field at the correct time, we performed a pretest to determine, how long per scent it took to an assessor to detect the presence of scents once the fans

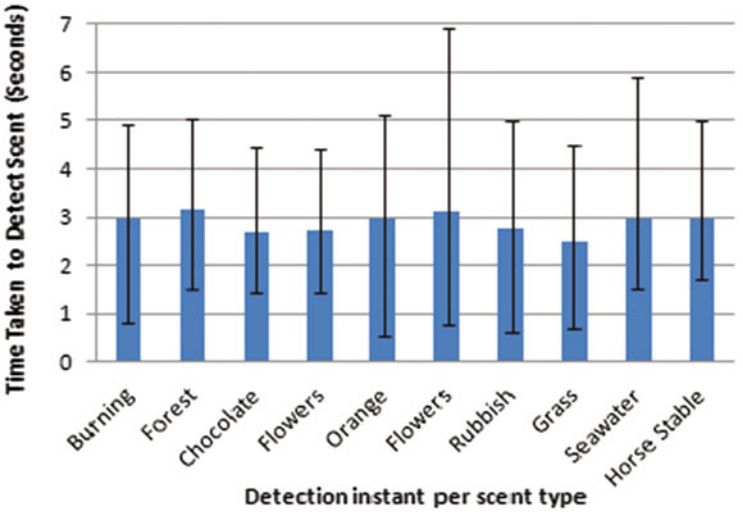


Fig. 12.11 Detection instant per scent average with maximum/minimum detection instants per scent type [96]

were turned on. The results are presented in Fig. 12.11. For each scent, the average time it takes an assessor to detect it is taken into account in terms of presenting the scent according to the above-mentioned skews, i.e. if it takes 3 s for a burnt scent to reach the assessor, the fan to emit the scent is turned on at time 27 s such that the scent reaches the assessor at time t = 30 s and as such is said to be synchronized with the video. The scent was presented continuously whilst corresponding to the video segment. To address the issue of the slow-moving nature of scent upon completion of the appropriate video sequence, a non-odour fan on the SBiX device was turned on in order to try to remove any lingering scent. We acknowledge that the lingering of scents remained a challenging issue throughout out testing.

Detection and Perception of Synchronization Error

In this section, we present the results of the user's ability to detect skew and perceive skew for group 1 and group 2. The group 1 results as discussed reflect conceptual delay scenarios, whereas the group 2 results reflect conceptual jitter.

Detection of Skew—"Conceptual Delay"

Figures 12.12 and 12.13 present the results of statements 1 and 2. The answers to statements 1 and 2 reflected the ability of group 1 users' ability to detect levels of inter-media skew between the video for both scents A and scent B, respectively. The vertical axes in each figure show the ratings related to the five possible answers

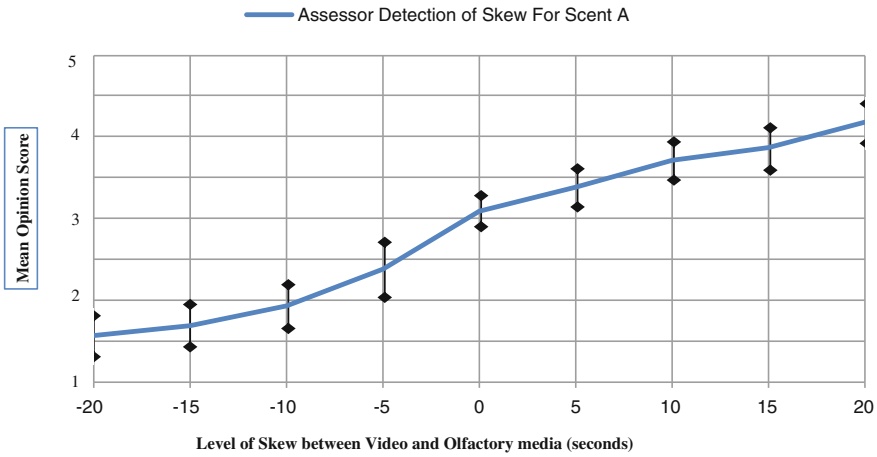


Fig. 12.12 Analysis of skew detection for scent A with confidence interval based on 95% confidence level for group 1 [96]

assessors provided to the statements. The horizontal axes indicate the level of skew artificially introduced between the olfactory and video media with the negative values representing olfactory media before video media. Both figures show assessors were able to identify the existence of inter-stream skew very well. The figures also indicate that the assessors were more sensitive to scent presented before video. Direct comparison of mean opinion scores (MOSs) presented in Fig. 12.12 at skews of +5 and −5 s show that the MOS for +5 s of 3.38 was closer to being at the "correct time" (represented by a value of 3) as opposed to the value of 2.38 for −5 s. Interestingly, based on MOS comparison, assessors viewed skews of +10 and −5 s similarly in terms of being Late or Early, respectively.

In order to analyse if significant differences existed in participants' perception between synchronized and unsynchronized scent and video, the data collected were analysed using independent sample t test with 95% confidence level. For all levels of skew between the olfactory data and video, statistically significant difference between assessors' opinion of mean of the synchronized and mean of participant responses for the "skewed" release times existed. With regard to scent B, interesting observations can be made. For the synchronized case (i.e. 0 s skew), assessors reported as being slightly early with a MOS score of 2.72 instead of a value of 3. As with scent A, assessors perceived olfaction presented after video as being closer to the correct time when compared with olfaction presented before video. Indeed, assessors viewed skews of +5 s (MOS 3.26) and +10 s (MOS 3.3) as being as close to the correct time as the correct presentation time of 0 s (2.72). Statistically, significant differences were found for all skew levels with the exception of −5 and +5 s.

Since assessors appear not to be as accurate in their detection of skew for scent B, analysis comparing the MOS scores between the same skew values of scent A and scent B was performed. As noted previously, the MOS values for 0 s, +5 and

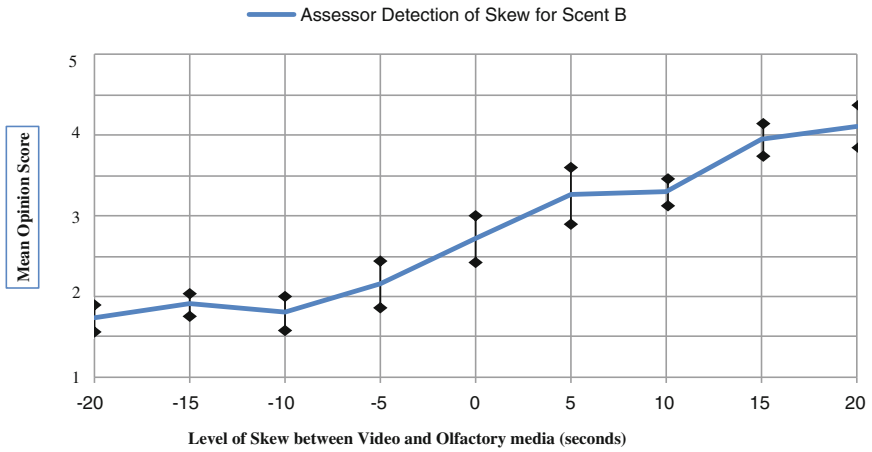


Fig. 12.13 Analysis of skew detection for scent B with confidence interval based on 95% confidence level for group 1 [96]

+10 s for scent B show that assessors perceived scent presented after video as being in sync. An independent sample t test was performed to determine if there were statistically significant differences between the assessor ability to detect skew for scent A and scent B. At skew values of +10 s, there was a statistically significant difference between scent A and scent B results. The ratings for other skews were not found to be statistically significant.

Perception of Skew—"Conceptual Delay"

The task of question 3 and 4 was to determine an assessor's perception of a skew if it existed for the olfactory-video clip. The effect an error has is key to determine temporal boundaries, as works involving other media have shown that users can tolerate certain levels of skew [87, 97, 103]. Hence, assessors were asked to qualify the level of impairment the inter-media skew had on the experience when comparing it to the synchronized reference sample. Figures 12.14 and 12.15 show, regarding group 1 assessors, the mean opinion score (MOS) for level of annoyance for inter-media skew for scent A and B, respectively. Based on the comparison of MOS scores for olfaction presented before and after video, assessors were clearly **less tolerant** to olfaction presented **before** video, than they were for olfaction presented **after** video between skews of −15 and +15 s.

For scent A, Fig. 12.14 shows a sharp increase in annoyance between 0 and −5 s with MOS values of 3.73 and 4.59, respectively. Although an increase in annoyance between 0 and +5 s (MOS 4.35) exists, the rating for +5 s remains in the imperceptible to perceptible to not annoying range. Skews of −5 and +10 s are perceived similarly by assessors as are skews of −10 and +15 s. For all skews with the exception of +5 s, statistically significant differences were reported.

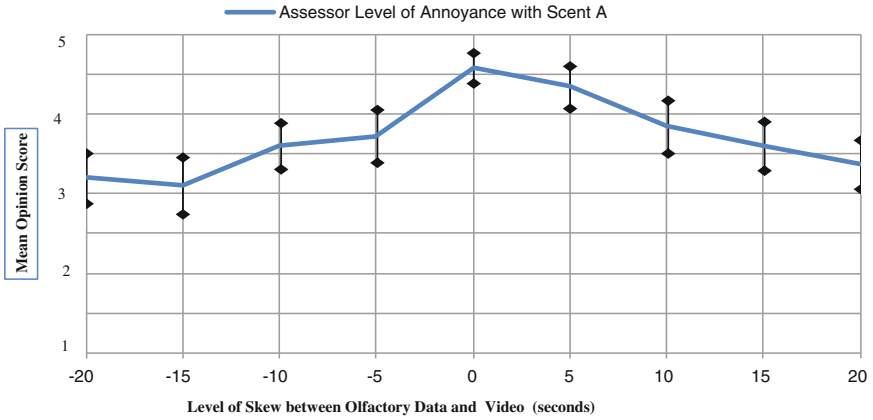


Fig. 12.14 Analysis of perception of skew for scent A with confidence interval based on 95% confidence level for group 1 [96]

For scent B, the findings are particularly interesting. Assessors reported skews of +5 s (MOS 4.35) and +10 s (MOS 4.2) as being less annoying than synchronized presentation 0 s (MOS 4), albeit all are in the range of perceptible but not annoying to imperceptible. Figure 12.15 clearly shows the decrease in annoyance as the time of presenting olfaction before video decreases. Statistically significant differences existed for all skew levels except +5, +10 and −5 s.

To compare ratings between scent A and scent B, an independent samples t test with 95% confidence interval was executed. Interestingly, only the synchronized presentation times of 0 s reported statistically significant differences. With this said, from Figs. 12.14 and 12.15, the trends towards lesser levels of annoyance for olfaction

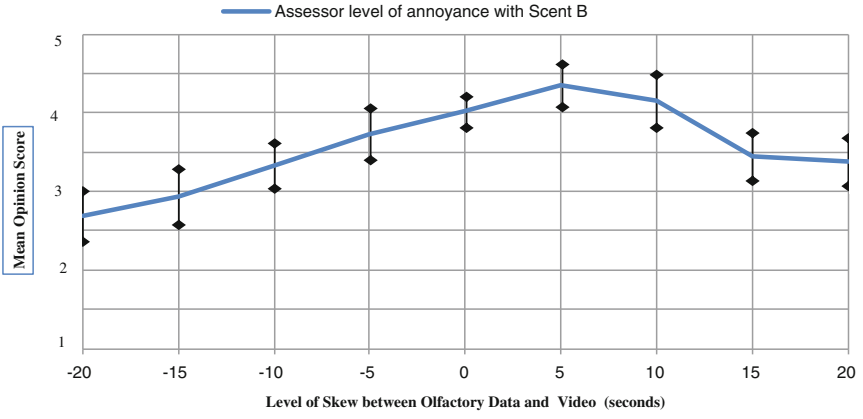


Fig. 12.15 Analysis of perception of skew for scent B with confidence interval based on 95% confidence level for group 1 [96]

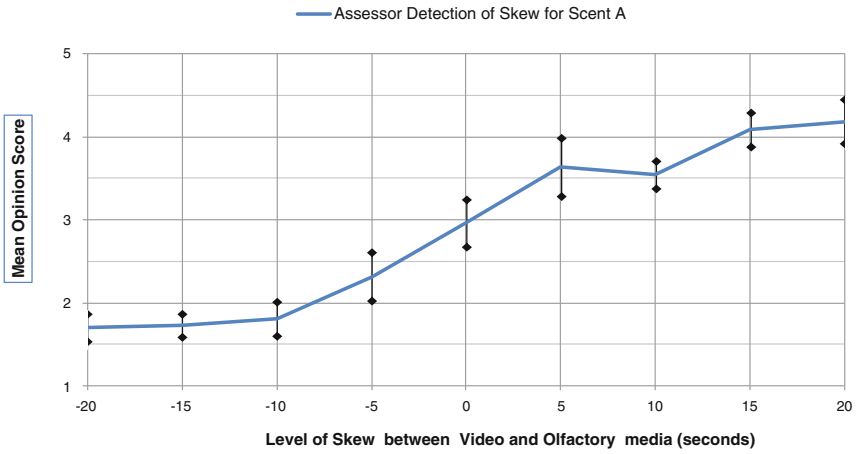


Fig. 12.16 Analysis of detection of skew for scent A with confidence interval based on 95% confidence level for group 2 [96]

presented after video are clear, particularly with assessor rating of annoyance for scent B. These findings are plausible, because in our everyday lives, we see first, smell after [87]. Olfaction presented without a visual cue acts as a distraction, which in other contexts highlights the potential for use of olfaction in warning systems.

Detection of Skew—"Conceptual Jitter"

Figures 12.16 and 12.17 present the general results of statements 1 and 2 for group 2 (again 50 participants). The group 2 participants experienced the conceptual jitter scenarios outlined above. Again the motivation was to determine users' ability to detect levels of inter-media skew between the video and the scents presented. They show assessors were able to identify the existence of inter-stream skew very well. Both figures also indicate that assessors were more sensitive to scent that was presented early rather than late in comparison with the video. Interestingly from Fig. 12.16, assessors rated skews of +5 s approximately the same as they rated skews of +10 s. Based on MOS comparison, ratings of skews at +10 and −10 s support the view that assessors are more sensitive to olfaction presented before the video than after it. To determine if statistically significant differences existed between the assessors' perception of synchronized and skewed presentation for scent A, independent samples t tests were executed. It reported that statistically significant differences existed between all skew levels with 95% confidence level.

Figure 12.17 reports assessor ratings of the various skews for scent B. The results indicate that assessors were not as accurate in their detection of skew for scent B as they were for scent A. We propose that the reason for this was the jitter between scent A and scent B (i.e. the skew level of scent A affected assessor

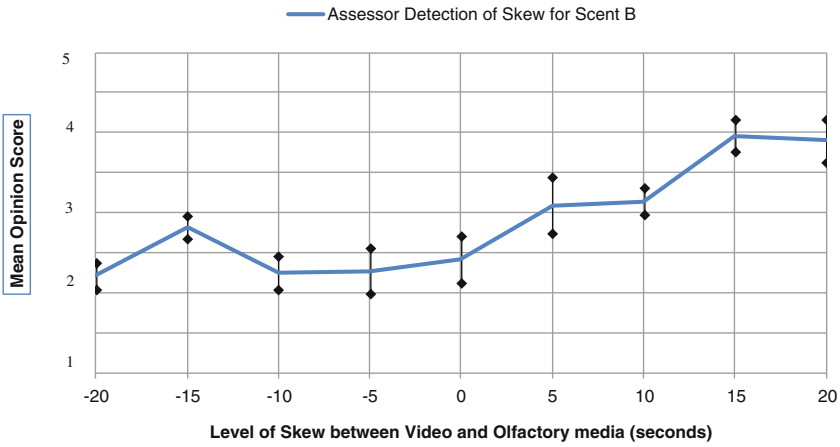


Fig. 12.17 Analysis of detection of skew for scent B with confidence interval based on 95% confidence level for group 2 [96]

detection of skew level for scent B). Interestingly, little difference exists between user detection at skews of 0, −5 and −10 s, as all of which were rated as being between early and at the correct time (MOS values of 2.41, 2.27 and 2.21, respectively). For olfaction presented after video, assessors rated skews of +5 and +10 s as being close to the correct time.

Statistically significant differences between skewed values and the synchronized presentation were found at skew levels of −15, +5, +10, +15 and 20 s with 95% confidence level following the t tests. To analyse the difference between the MOS values reported for detection of skews for scent A and scent B at the same skew levels, an independent samples t test was performed. The purpose of this analysis was to determine if there were significant differences between how assessors viewed the same skew levels when impacted by jitter. Based on 95% confidence interval, statistically significant differences at skews of −15, 0 and +5 s were found.

Perception of skew—"conceptual jitter"

Figures 12.18 and 12.19 reflect group 2 assessors' perception of skew for both scent A and B. As was the case for group 1, assessors are more sensitive to and more easily annoyed by scent presented before the video than scent presented after the video. Figure 12.18 presents assessor's perception of skews with scent A. It shows assessors found synchronized presentation as being the least annoying with MOS of 4.28. Assessors rated skews of −5 s similarly to +5 and +10 s with MOS scores of 3.92, 3.85 and 4, all close to or at a rating of perceptible but not annoying. They also rated skews of −10 s similar to +15 s with MOS values of 3.52 and 3.41. Via independent samples t test, skews of −20, −15, +15 and +20 s resulted in statistically significant differences with confidence level of 95%. For skews

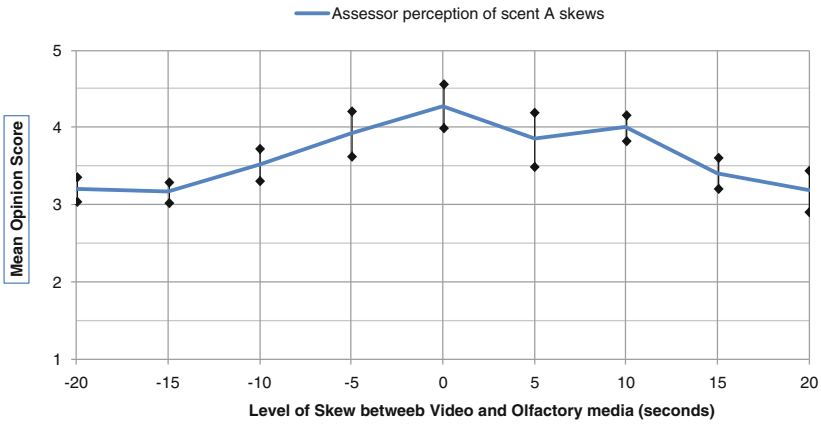


Fig. 12.18 Analysis of perception of skew for scent A with confidence interval based on 95% confidence level for group 2 [96]

of −10 s, the results were just on the border of being statistically significant, whereas for +10 s, it cannot be said that there is any statistical difference in the results with any significant confidence. For scent B, assessors reported a skew of +5 s to be the least annoying with MOS of 4.24. As per Fig. 12.19, the assessors rated skews of −5 and +20 s similarly. This again validates the belief that users were more tolerant of olfaction presented after the video than before it. Skews of −20 and −10 s were found to be statistically significant with confidence level of 95%. As was the case for detection, the perception of scent B with skew of −15 s appears at odds but as outlined above, given the assessors perception of this skew, the rating is understandable. An independent samples t test was run between MOS of the ratings for scent A and scent B. Statistically significant differences were found at −5 and 0 s with confidence level of 95%.

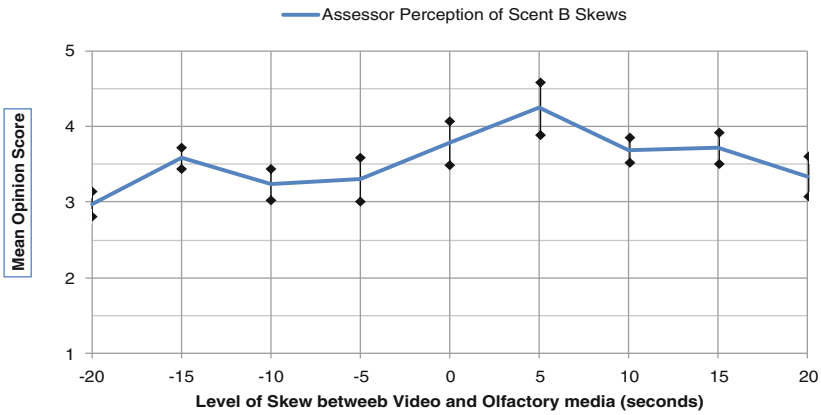


Fig. 12.19 Analysis of perception of skew for scent B with confidence interval based on 95% confidence level for group 2 [96]

For more specific analysis on the assessor perception of mixing of scents in terms of assessor annoyance, we performed grouping between the various test scenarios that have common mixing times, e.g. test cases which resulted in overlaps of 4, 9, 14 and 24 s. For the test cases where there were 4 and 9 s overlaps, approximately 40% of the test case assessors actually detected the mix. For 14 and 24 s overlap, the detection was approximately 60% and 70%, respectively. Interestingly, when the two scents were presented both at the same time (i.e. 24 s overlap), assessors only reported detecting one scent in three of the 24 test scenarios executed all when the scents of fruit and flower were mixed. Assessors found 4 s overlap the least annoying with a MOS of 3.52, on the mid-point between perceptible but not annoying and slightly annoying. They found overlaps of 9 and 24 s to be between slightly annoying and annoying (MOS 3.05 and 3.32), respectively. A 14 s overlap was rated as between slightly annoying and the rating for 4 s overlap with a MOS of 3.32. The results indicate that assessors found the experience of overlapping of scents as being somewhat annoying.

12.4 Conclusions

This chapter has introduced the idea of olfaction-enhanced multimedia synchronization and reported some initial findings in terms of user perception of olfaction-enhanced multimedia. It has discussed related proposals in terms of solutions for olfaction-enhanced multimedia and their applications in diverse areas such as entertainment, tourism. Additionally, the chapter has presented a current standard and a current research solution which can accommodate the delivery of olfaction-enhanced multimedia and their related architectural design. Next, the chapter has presented and discussed the results of a number of subjective tests which have analysed the ability of users to detect and their perception of synchronization of olfaction-enhanced multimedia. It highlighted how user's ability to detect skew was affected when more than one olfactory component was presented and in particular when two olfactory streams overlapped. In terms of acceptable skew levels for olfaction-enhanced multimedia, the following general temporal boundaries are proposed for olfaction-enhanced multimedia synchronization:

(1) −5 to +10 s as being the "in-sync" region for olfaction-enhanced multimedia [96]
(2) Skew values beyond this boundary are "out-of-sync" [96].

Outside of the "in-sync" range, assessors reported skews as being annoying to varying degrees, and also that the skews have a negative impact on assessor QoE. Finally, our future work will include implicit (physiological [104, 105] and Psychophysiology-based [106]) and explicit involve experiential evaluation of the applicability of olfaction for use in the areas of health and education, as well as considering the opportunities for employing other sensorial multimedia across these application domains.

Definitions

Olfaction-enhanced multimedia The presentation of olfactory stimuli in addition to traditional media content (audio-visual).

Quality of Experience the degree of delight or annoyance of a person whose experiencing involves an application, service, or system. It results from the person's evaluation of the fulfillment of his or her expectations and needs with respect to the utility and/or enjoyment in the light of the person's context, personality and current state.

Skew skew reflects the difference in presentation times between the related olfactory media component and the associated video media components (i.e. if both synchronized, there is a 0s skew). The skew levels are not the results of network transmission effects but are conceptual.

Mulsemedia multiple sensorial media applications are those that engage three (or more) of our senses.

References

1. Blakowski, G., Steinmetz, R.: A media synchronization survey: reference model, specification, and case studies. In: IEEE J. Sel. Areas Commun. **14**(1), 5–35 (1996). https://doi.org/10.1109/49.481691
2. Spencer, B.S.: Incorporating the sense of smell into patient and haptic surgical simulators. IEEE Trans. Inf Technol. Biomed. **10**(1), 168–173 (2006)
3. Murray, N., Lee, B., Qiao, Y., Muntean, G.-M.: The impact of scent type on olfaction-enhanced multimedia quality of experience. IEEE Trans. Systems, Man, and Cybernetic: System. **47**(9), 2503–2515 (2017). https://doi.org/10.1109/tsmc.2016.2531654
4. Brewster, S.A., McGoookin, D.K. Miller, C.A.: Olfoto: designing a smell-based interaction. In: ACM SIGCHI Conference on Human Factors in Computing Systems (2006)
5. Timmerer, C., Waltl, M., Rainer, B., Murray, N.: Sensory experience: quality of experience beyond audio-visual. In: Möller, S., Raake, A. (eds.) Quality of Experience: Advanced Concepts, Applications and Methods, pp. 351–365 (2014). Springer, Heidelberg, Germany. https://doi.org/10.1007/978-3-319-02681-7_24
6. Ghinea, G., Gulliver, S.R., Andres, F. (eds.) Multiple Sensorial Media Advances and Applications: New Developments in mulsemedia. IGI Global (2011). https://doi.org/10.4018/978-1-60960-821-7
7. Ghinea, G., Timmerer, C., Lin, W., Gulliver, S.R.: Mulsemedia: state of the art, perspectives, and challenges. ACM Trans. Multimedia Comput. Commun. Appl. **11**(1s), Article 17, 23 pp (2014). https://doi.org/10.1145/2617994 http://doi.acm.org/10.1145/2617994
8. Washburn, D.A.: Olfactory use in virtual environment training. Model. Simul. **2**(3), 19–25 (2003)
9. Gerardi, M., Rothbaum, BO., Ressler, K., Keekin, M., Rizzo, A.: Virtual reality exposure using a virtual iraq: case report. J. Trauma. Stress **21**(2) (2008)
10. Shams, L., Seitz, A.R.: Benefits of multisensory learning. Trends Cogn. Sci. **12**(11), 411–417 (2008)

11. Obrist, M., Tuch, A.N., Hornbaek, K.: Opportunities for odour: experiences with smell and implications for technology. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp 2843–2852 (2014). https://doi.org/10.1145/2556288.2557008
12. Hirota, K., Ebisawa, S., Amemiya, T., Ikei, Y.: A theater for viewing and editing multi-sensory content. In: IEEE International Symposium on VR Innovation (ISVRI), pp 239–244 (2011). https://doi.org/10.1109/isvri.2011.5759643
13. Ghinea, G., Ademoye, O.A.: Olfaction-enhanced multimedia: bad for information recall? In: International Conference on Multimedia and Expo (ICME), pp. 970–973 (2009). https://doi.org/10.1109/icme.2009.5202658
14. Ghinea, G., Ademoye, O.A.: Olfaction-enhanced multimedia: perspectives and challenges. Multimed. Tools Appl. **55**(3), 601–626 (2011)
15. Murray, N., Qiao, Y., Lee, B., Karunakar, A.K., Muntean, G.-M.: Olfaction enhanced multimedia: a survey of application domains, displays and research challenges. ACM Comput. Surv. **48**(4), Article No. 56 (2016). https://doi.org/10.1145/2816454
16. Murray, N., Ademoye, O.A., Ghinea, G., Muntean, G.: A tutorial for olfaction-based multisensory media application design and evaluation. Accepted for publication in ACM Computing Surveys 2017
17. http://www.cj4dx.com/. Accessed 01 July 2017
18. Ramic, B., Hasic, J., Rizvic, S., Chalmers, A.: Selective rendering in a multimodal environment: scent and graphics. In: Spring Conference on Computer Graphics, pp. 147–151 (2006). ACM SIGGraph. https://doi.org/10.1145/2614348.2614369
19. Brkic, B.R., Chalmers, A.: Virtual smell: authentic smell diffusion in virtual environments. In: Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, pp. 45–52, (2010). https://doi.org/10.1145/1811158.1811166
20. Brkic, B.R., Chalmers, A., Sadzak, A., Debattista, K., Sultanic, S.: Exploring multiple modalities for selective rendering of virtual Environments. In: Spring Conference on Computer Graphics, pp. 91–98 (2013). https://doi.org/10.1145/2508244.2508256
21. Miyaura, M., Narumi, T., Nishimura, K., Tanikawa, T., Hirose, M.: Olfactory feedback system to improve the concentration level based on biological information. In: IEEE Virtual Reality Conference, pp. 139–142 (2011). https://doi.org/10.1109/vr.2011.5759452
22. Covarrubias, M., Bordegoni, M., Rosini, M., Guanziroli, E., Cugini, U., Molteni, F.: VR system for rehabilitation based on hand gestural and olfactory interaction. In: Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology, pp. 117–120 (2015). https://doi.org/10.1145/2821592.2821619
23. van Hoven, B.: Multi-sensory tourism in the great bear rainforest. J. Assoc. Icel. Geogr. **25**, 31–49 (2011)
24. Lai, M.K.: Universal scent BlackBox—engaging visitors communication through creating olfactory experience at Art Museum. In: Proceedings of the 33rd Annual International Conference on the Design of Communication Article No. 27 (2015). https://doi.org/10.1145/2775441.2775483
25. Hiroshi, A., Liu, J., Kim, D.W.: Multi-sensory interaction technology and its system application. http://www.nict.go.jp/publication/shuppan/kihou-journal/journal-vol57no1_2/journal-vol57no1-2_0503.pdf
26. Fernstrom, M., Bannon, L.: Enabling technology for museum visitors: issues and experiences. In: Proceedings of the International Conference on Museums and the Web 1997, Los Angeles
27. Tuan, Y.F., Hoelscher, S.: Space and Place: The Perspective of Experience. University of Minnesota Press (2001). ISBN-10:0816638772
28. Porteous, J.D.: Smellscape. Progress Phys. Geogr. **9**(3), 356–378 (1985). https://doi.org/10.1177/030913338500900303
29. Hall, T., Bannon, L.: Designing ubiquitous computing to enhance children's interaction in museums. In: Proceedings of the 2005 conference on Interaction design and children, pp. 62–69 (2005). https://doi.org/10.1145/1109540.1109549

30. Cheong, R.: The virtual threat to travel and tourism. Tour. Manag. **16**(6), 417–422 (1995). https://doi.org/10.1016/0261-5177(95)00049-T
31. Dann, G., Jacobsen, J.K.S.: Leading the tourist by the nose. The tourist as a metaphor for the social world, pp 209–235 (2009). https://doi.org/10.1079/9780851996066.0209
32. Kaye, N.: Symbolic olfactory display, M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2001
33. Gardner, H.: Frames of Mind: The Theory of Multiple Intelligences (1983). Basic Book Inc., New York
34. Shams, L., Seitz, A.R.: Benefits of multisensory learning. Trends Cogn. Sci. **12**(11), 411–417 (2008). https://doi.org/10.1016/j.tics.2008.07.006
35. Haverkamp, M.: Synesthetic approach for evaluation of the cross-sensory quality of multi-media applications. In: International Workshop on Quality of Multimedia Experience (QoMEX), pp 136–141 (2010). https://doi.org/10.1109/QOMEX.2010.5516107
36. Hyacinthe, B.P.: Apparatus and methods for production of printed aromatic and gustative information. In: Proceedings of 6th ACM/IEEE-CS Joint Conference on Digital Libraries, pp 365 (2006). https://doi.org/10.1145/1141753.1141860
37. Tortell, R., Luigi, D.P., Dozois, A., Bouchard, S., Morie, J.F., Ilan, D.: The effects of scent and game play experience on memory of a virtual environment. IEEE Virtual Real. **11**(1), 61–68 (2007). https://doi.org/10.1007/s10055-006-0056-0
38. Herz, R.S.: Influences of odours on mood and affective cognition. In: Olfaction, Taste, and Cognition, pp. 160–177 (2002). https://doi.org/10.1017/cbo9780511546389.016
39. Barros, P.G.-D., Lindeman, R.: Performance effects of multi-sensory displays in virtual environments. In: Symposium on Spatial User Interaction (2013). https://doi.org/10.1145/2491367.2491371
40. Mikropoulos, T.A., Natsis, A.: Educational virtual environments: a ten-year review of empirical research (1999–2009). Comput. Educ. **56**(3), 769–780 (2011). https://doi.org/10.1016/j.compedu.2010.020
41. Tijou, A., Richard, E., Richard, P.: Using olfactive virtual environments for learning organic molecules. In: Technologies for E-Learning and Digital Entertainment, vol. 3942, pp. 1223–1233 (2006). Springer. https://doi.org/10.1007/11736639_152
42. Richard, E., Tijou, A., Richard, P., Ferrier., J.-L.: Multi-modal virtual environments for education with haptic and olfactory feedback. J. Virtual Real. **10**(3), 207–225 (2006). https://doi.org/10.1007/s10055-006-0040-8
43. Richard, E., Tijou, A., Richard, P.: Multi-modal virtual environments for education: from illusion to immersion. In: Technologies for E-Learning and Digital Entertainment, vol. 3942, pp. 1274–1279 (2006). Springer. https://doi.org/10.1007/11736639_158
44. Dinh, H.O., Walker, N., Song, C., Kobayashi, A., Hodges, L.F.: Evaluating the importance of multi-sensory input on memory and the sense of presence in virtual environments. In: IEEE Virtual Reality, pp. 222–228 (1999). https://doi.org/10.1109/vr.1999.756955
45. Hughes, C.E., Stapleton, C.B., Hughes, D.E., Smith, E.M.: Mixed reality in education, entertainment and training. IEEE Comput. Graph. Appl. **25**(6), 24–30 (2005). https://doi.org/10.1109/mcg.2005.139
46. Ludvigson, W., Rottman, T.R.: Effects of ambient odours of lavender and cloves on cognition, memory, affect and mood. Chem. Senses **14**(4), 525–536 (1989). https://doi.org/10.1093/chemse/14.4.525
47. Mustonen, S., Rantanen, R., Tuorila, H.: Effect of sensory education on school children's food perception: a 2-year follow-up study. Food Qual. Prefer. **20**(3), 230–240 (2009). https://doi.org/10.1016/j.foodqual.2008.10.003
48. Ron, C.-W.K., Cheng, S.H., Ip, H.H.-S., Joseph, S.-L.K.: Design of affectively evocative smart ambient media for learning. Comput. Educ. **56**(1), 101–111 (2011). https://doi.org/10.1016/j.compedu.2010.08.015
49. Childers, C., Coleman, M.: The role olfactory technology in serious gaming, mental training, and therapy. http://sensoryacumen.com. Accessed 11 June 2013

50. Cater, J.P.: Smell/taste: odours in virtual reality. In: IEEE International Conference Systems, Man and Cybernetics, Human Information and Technology, vol. 2 (1994). https://doi.org/10.1109/icsmc.1994.400108
51. Muller, D., Bruns, F.W., Erbe, H.H., Robben, B., Yoo, Y.H.: Mixed reality learning spaces for collaborative experimentation: a challenge for engineering education and training. Int. J. Online Eng. **3**(4) (2007)
52. Garcia-Ruiz, M.A., Edwards, A., Aquino-Santos, R., Alvarez-Cardenas, O., Mayoral-Baldivia, M.G.: Integrating the sense of smell in virtual reality for second language learning. In: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, pp. 2647–2652 (2008)
53. Fletcher, J.D.: Education and training technology in the military. Science **323**(5910), 72–75 (2009). https://doi.org/10.1126/science.1167778
54. Borromeo, S., Hernandez-Tamames, J.A., Luna, G., Machado, F., Malpica, N., Toledano, A.: Objective assessment of olfactory function using functional magnetic resonance imaging (fMRI). IEEE Trans. Instrum. Meas. **59**(10), 2602–2608 (2010). https://doi.org/10.1016/j.otoeng.2012.07.005
55. Haehner, A., Hummel, T., Reichmann, H.: Olfactory dysfunction as a diagnostic marker for Parkinson's disease. Expert Rev. Neurother. **9**(12), 1773–1779 (2009). https://doi.org/10.1586/ern.09.115
56. Bohner, N.I., Mueller, M.L., Kotagal, V., Keoppe, R.A., Kilbourn, M.A., Albin, R.L., Frey, K.A.: Olfactory dysfunction, central cholinergic integrity and cognitive impairment in Parkinson's disease. Brain **133**(6), 1747–1754 (2010). https://doi.org/10.1093/brain/awq079
57. Pereira, F., Burnett, I.: Universal multimedia experiences for tomorrow. IEEE Signal Process. Mag. **20**(2), 63–73 (2003). https://doi.org/10.1109/msp.2003.1184340
58. Redd, W., Manne, S.: Using aroma to reduce distress during magnetic resonance imaging. In: A. Gilbert (ed.) Compendium of Olfactory Research 1982–1994, Dubuque, pp. 47–52. Kendall/Hunt Publishing Co., Iowa (1995)
59. Depledge, M.H., Stone, R.J., Bird, W.J.: Can natural and virtual environments be used to promote improved human health and wellbeing? Environ. Sci. Technol. **45**(11), 4660–4665 (2011). https://doi.org/10.1021/es103907m
60. Baus, O., Bouchard, S.: The sense of olfaction: its characteristics and its possible applications in virtual environments. J. CyberTherapy Rehabil. **3**(1), 31–50 (2010)
61. Spenser, B.S.: Incorporating the sense of smell into patient and haptic surgical simulators. IEEE Trans. Inf. Technol. Biomed. **10**(1) (2006). https://doi.org/10.1109/titb.2005.856851
62. Gerardi, M., Rothbaum, B.O., Ressler, K., Heekin, M., Rizzo, A.: Virtual reality exposure therapy using a virtual Iraq: case report. J. Trauma. Stress **21**(2), 209–213 (2008). https://doi.org/10.1002/jts.20331
63. Rizzo, A., Newman, B., Parsons, T., Difede, J., Reger, G., Holloway, K., Gahm, G., McLay, R., Johnston, S., Rothbaum, B., Graap, K., Spitalnick, J., Bordnick, P.: Development and clinical results from the virtual Iraq exposure therapy application for PTSD. In: Virtual Rehabilitation International Conference, pp. 8–15 (2009)
64. Rizzo, A., Graap, K., Mclay, R.N., Perlman, K., Rothbaum, B.O., Reger, G., Parsons, T., Difede, J., Pair, J.: Virtual Iraq: initial case reports from a VR exposure therapy application for combat-related post traumatic stress disorder. Stud. Health Technol. Inform. **132**, 420–425 (2008)
65. Brewin, B.: Combat trauma theatre. http://www.govhealthit.com/news/combat-trauma-theater-3. Accessed 11 June 13
66. Pair, J., Allen, B., Dauticourt, M., Treskunov, A., Liewer, M., Graap, K., Reger, G.: A virtual reality exposure therapy application for Iraq war post traumatic stress disorder. In: IEEE Virtual Reality Conference, pp. 67–72 (2006). https://doi.org/10.1109/vr.2006.23
67. Rizzo, A., Parsons, TD., Lange, B., Kenny, P.,. Buckwalter, JG., Rothbaum, B., Difede, J., Frazier, J., Newman, B., Williams, J., Reger, G.: Virtual reality goes to war: a brief review of the future of military behavioral healthcare. J. Clin. Psychol. Med. Setting **18**(2), 176–187 (2011). https://doi.org/10.1007/s10880-011-9247-2

68. Yeh, S.-C., Newman, B., Liewer, M., Pair, J.: A virtual Iraq system for the treatment of combat-related post traumatic stress disorder. In: IEEE Virtual Reality Conference, pp 163–170 (2009). https://doi.org/10.1109/vr.2009.4811017
69. Richard, E., Billaudeau, V., Richard, P., Gaudin, G.: Augmented reality for rehabilitation of cognitive disabled children: a preliminary study. In: IEEE International Conference on Virtual Rehabilitation, pp. 102–108 (2007). https://doi.org/10.1109/icvr.2007.4362148
70. Borromeo, S., Hernandez-Tamames, J.A., Luna, G., Machado, F., Malpica, N., Toledano, A.: Objective assessment of olfactory function using functional magnetic resonance imaging (fMRI). IEEE Trans. Instrum. Measur. **59**(10), 2602–2608 (2010)
71. Warnock, D.: A subjective evaluation of multimodal notifications. In: 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth), pp 461–468 (2011). https://doi.org/10.4108/icst.pervasivehealth.2011.246001
72. ISO/IEC FDIS 23005-1 Information Technology—Media Context and Control—Part 1: Architecture, ISO Publication (2010)
73. ISO/IEC FDIS 23005-2 Information Technology—Media Context and Control—Part 2: Control Information, ISO Publication (2010)
74. ISO/IEC FDIS 23005-3 Information Technology—Media Context and Control—Part 3: Sensory Information, ISO Publication (2010)
75. ISO/IEC FDIS 23005-4 Information Technology—Media Context and Control—Part 4: Virtual World Object Characteristics, ISO Publication (2010)
76. ISO/IEC FDIS 23005-5 Information Technology—Media Context and Control—Part 5: Data formats for Interaction Devices, ISO Publication (2010)
77. ISO/IEC FDIS 23005-6 Information Technology—Media Context and Control—Part 6: Common Types and Tools, ISO Publication (2010)
78. ISO/IEC FDIS 23005-7 Information Technology—Media Context and Control—Part 7: Reference Software and Conformance, ISO Publication (2010)
79. Choi, B.S., Joo, S.H., Lee, H.Y.: Sensory effect metadata for SMMD media service. In: Proceedings of the Fourth International Conference on Internet and Web Applications and Services, pp. 649–654 (2009). https://doi.org/10.1109/iciw.2009.104
80. Pyo, S., Joo, S.H., Choi, B.S., Kim, M.: A metadata schema design on representation of sensory effect information for sensible media and its service framework using UPnP. In: Proceedings of the 10th International Conference on Advanced Communication Technology (ICACT'08) (2008). https://doi.org/10.1109/icact.2008.4493965
81. Yoon, K., Choi, B.S., Lee, E.S., Lim, T.B.: 4-D broadcasting with MPEG-V. In: IEEE International Workshop on Multimedia Signal Processing (MMSP'10), pp. 257–262 (2010). https://doi.org/10.1109/mmsp.2010.5662029
82. Yun, J., Jang, J., Moon, K.: Development of the real-sense media broadcasting service system based on the SMMD. In: Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), pp. 435–436 (2011). https://doi.org/10.1109/icce.2011.5722669
83. Yun, J., Jang, J., Park, K., Han, D.: Real-sense media representation technology using multiple devices synchronization. In: Software Technologies for Embedded and Ubiquitous Systems, vol. 5860, pp. 343–353 (2009). https://doi.org/10.1007/978-3-642-10265-3_31
84. Yuan, Z., Chen, S., Ghinea, G., Muntean, G.M.: Beyond multimedia adaptation: quality of experience-aware multi-sensorial media delivery. IEEE Trans. Multimed. **17**(1), 104–117 (2015). https://doi.org/10.1109/tmm.2014.2371240
85. Muntean, G.-M., Perry, P., Murphy, L.: A new adaptive multimedia streaming system for All-IP multi-service networks. IEEE Trans. Broadcast. **50**(1), 1–10 (2004). https://doi.org/10.1109/tbc.2004.824745
86. Yuan, Z., Venkataraman, H., Muntean, G.-M.: MBE: model-based available bandwidth estimation for IEEE 802.11 data communications. IEEE Trans. Veh. Technol **61**(5), 2158–2171 (2012). https://doi.org/10.1109/tvt.2012.2190760

87. Murray, N., Qiao, Y., Lee, B., Karunakar, A.K., Muntean, G.-M.: Subjective evaluation of olfactory and visual media synchronization. In: Proceedings of ACM Multimedia Systems conference, Feb 26–Mar 1, Oslo, Norway, pp. 162–171 (2013). https://doi.org/10.1145/2483977.2483999
88. Kovács, P.T., Murray, N., Rozinaj, G., Sulema, Y., Rybárová, R.: Application of immersive technologies for education: state of the art. In: 9th International Conference on Interactive Mobile Communication Technologies and Learning, IMCL2015, Special Session on Immersive Technologies for Effective Learning (2015). https://doi.org/10.1109/imctl.2015.7359604
89. Murray, N., Qiao, Y., Lee, B., Karunakar, A.K., Muntean, G.-M.: Age and gender influence on perceived olfactory and visual media synchronization. In: IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6, July 15–19, San Jose, California, USA (2013). https://doi.org/10.1109/icme.2013.6607467
90. Murray, N., Qiao, Y., Lee, B., Muntean, G.M.: User-profile-based perceived olfactory and visual media synchronization. In: ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **10**(1s), article 11 (2014). https://doi.org/10.1145/2540994
91. Yuan, Z., Chen, S., Ghinea, G., Muntean, G.M.: Beyond multimedia adaptation: quality of experience-aware multi-sensorial media delivery. IEEE Trans. Multimed. **17**(1), 104–117 (2015). https://doi.org/10.1109/tmm.2014.2371240
92. Ademoye, O.A., Murray, N., Muntean, G.-M., Ghinea, G.: Audio masking effect on inter-component skews in olfaction-enhanced multimedia presentations. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **12**(4), article 51 (2016), https://doi.org/10.1145/2957753
93. Murray, N., Ademoye, O.A., Ghinea, G., Qiao, Y., Muntean, G.M., Lee, B.: Olfactory-enhanced multimedia video clips datasets. Ninth International Conference on Quality of Multimedia Experience (QoMEX), 2017. https://doi:10.1109/QoMEX.2017.7965653
94. Ohtsu, K., Sato, J., Bannai, Y., Okada, K.: Scent presentation technique of pulse ejection synchronized with breathing. In: Ninth Annual International Symposium on Applications and the Internet, pp 125–128 (2009). https://doi.org/10.1109/saint.2009.28
95. Noguchi, D., Sugimoto, S., Bannai, Y., Okada, K.: Time characteristics of olfaction in a single breath. In: ACM CHI Conference on Human Factors in Computing Systems (CHI2011), pp. 83–92 (2011), https://doi.org/10.1145/1978942.1978956
96. Murray, N., Lee, B., Qiao, Y., Muntean, G.M.: Multiple-scent enhanced multimedia synchronization. In: ACM Trans. Multimed. Comput. Commun. Appl. (TOMM), Special Issue on MulseMedia **11**(1s), article no. 12 (2014). https://doi.org/10.1145/2637293
97. Steinmetz, R.: Human perception of jitter and media synchronization. IEEE J. Sel. Areas Commun. **14**(1), 61–72 (1996). https://doi.org/10.1109/49.481694
98. www.exhalia.com
99. ISO 5496:2006—Sensory analysis—Methodology—Initiation and training of assessors in the detection and recognition of odours
100. Nakamoto, T., Otaguro, S., Kinoshita, M., Nagahama, M., Ohinishi K., Ishida T.: Cooking up an interactive olfactory game display. IEEE Comput. Graph. Appl. **28**(1), 75–78 (2008). https://doi.org/10.1109/MCG.2008.3
101. ISO/IEC 8589 Sensory analysis—General guidance for the design of test rooms
102. Murray, N.: Questionnaire_TOMCCAP_Special_Issue_2013. www.niallmurray.info/Research/appendix
103. Ghinea, G., Ademoye, O.A.: Perceived synchronization of olfactory multimedia. In: IEEE Trans. Syst. Man Cybern.—Part A: Syst. Hum. **40**(4), 657–663 (2010). https://doi.org/10.1109/tsmca.2010.2041224
104. Keighrey, N., Flynn, R., Murray, S., Brennan, S., and Murray, N.: Comparing User QoE via Physiological and Interaction Measurements of Immersive AR and VR Speech and Language Therapy Applications. In: Proceeding Thematic Workshops '17 Proceedings of

the on Thematic Workshops of ACM Multimedia 2017 Pages 485–492, https://doi.10.1145/3126686.3126747

105. Keighrey, C., Flynn, R., Murray, S., Murray, N.: A QoE evaluation of immersive augmented and virtual reality speech & language assessment applications. 2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX), 2017 https://doi:10.1109/QoMEX.2017.7965656
106. Engelke, U., Darcy, D.P., Mulliken, G.H., Bosse, S., Martini, M.G., Arndt, S., Antons, J-.N., Chan, K.Y., Ramzan, N., Brunnstrom, K.: Psychophysiology-based QoE assessment: a survey. IEEE J. Sel. Top. Sign. Proces. **11**(1), 6–21

Part IV
Document Formats and Standards

Chapter 13
SMIL: Synchronized Multimedia Integration Language

Dick C. A. Bulterman

Abstract The period from 1995 to 2010 can be considered to be networked multimedia's Golden Age: Many formats were defined that allowed content to be captured, stored, retrieved, and presented in a networked, distributed environment. The Golden Age happened because network infrastructures had enough bandwidth available to meet the presentation needs for intramedia synchronization, and content codecs were making even complex audio/video objects storable on network servers. This period marked the end of the CD-ROM era for multimedia content distribution. Unlike the relative simplicity of CD-ROM multimedia, where timing constraints were well-understood and pre-delivery content customization was relatively simple, the network multimedia era demanded new languages that would allow content to be defined as a collection of independent media components that needed to be located, fetched, synchronized, and presented on a large collection of user devices (under greatly varying network characteristics). One of the most ambitious projects to define an open and commonly available multimedia content integration language was W3C's SMIL. In a period of approximately ten years, SMIL grew from a simple synchronization language to a full content integration and scheduling facility for a wide range of Web documents. This chapter considers the timing and synchronization aspects of SMIL.

Keywords Structured timing · Hierarchical synchronization · Web-based multimedia

D. C. A. Bulterman (✉)
Vrije Universiteit Amsterdam and CWI, Amsterdam, The Netherlands
e-mail: dick.bulterman@gmail.com

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_13

13.1 Introduction

This chapter provides an overview of the W3C Synchronized Multimedia Integration Language (SMIL), which was developed and is maintained by the World Wide Web Consortium (W3C) [1]. SMIL has been defined as a series of *W3C Recommendations* [2–5], the term used by W3C to indicate a member-reviewed and member-approved standard for use within the W3C's suite of protocols and formats. SMIL is a comprehensive presentation format that can be used to structure the timing, layout, and user control of a set of content objects. While SMIL can be used to address a wide array of temporal control applications, this chapter will concentrate on SMIL as a multimedia presentation language.

The SMIL Recommendation is structured as a set of *modules*, each providing a collection of elements, attributes, and attribute values that can be selectively included when designing XML languages. The SMIL modules have also been grouped into a number of *profiles* that each serves the implementation needs of individual use domains. Since this book is particularly concerned with media synchronization, the discussion of SMIL in this chapter will largely be limited to the timing and synchronization aspects of the language. We will also consider how timing and synchronization are influenced by user interaction: mostly by activating temporal hyperlinks, but also where parts of a presentation are activated by event-based user interaction.

In the sections below, we start with a short history of SMIL, tracing the roots of the formats on which SMIL was based. We then provide a short primer on how SMIL is structured, with enough background to understand the examples given later in the chapter. We then consider the details of SMIL timing and interaction. We close with a short reflection on the success and limitations of the language.[1]

13.2 A Brief History of SMIL

In 1996, W3C started an activity to determine how audio and video could best be supported in the context of the a-temporal HTML text and image content that was then dominant on the World Wide Web. This activity, lead by Philipp Hoschka, resulted in the definition of the Synchronized Multimedia (SYMM) working group [7]. The working group consisted of several major providers of media players and embedded technology (most notably Apple, Intel, Microsoft, Philips, RealNetworks), a collection of academic institutions (CWI, GMD, and INRIA), and organizations interested in accessibility (WGBH and the Daisy consortium). After about nine months of work, the group published the first version of SMIL in June 1998. The architecture of SMIL was based largely on CWI's CMIF format [8], with modifications, restrictions, and extensions flowing from the design-by-committee

[1]This chapter draws on material published in [6].

process. Subsequent editions of SMIL, up to SMIL 3.0, were produced through December 2008, when the working group ceased active development.

From its earliest version, SMIL differed significantly from other media support activities available at that time. Unlike Apple's Quicktime and the Windows Media Player—the two dominant content delivery sources at the time—SMIL was not a *content object delivery platform* in which a single video or audio item could be played, but a *presentation delivery platform*, in which several independent media objects could be gathered (potentially from multiple servers), scheduled, and then presented via an open Web interface. Treating a media presentation as a collection of objects rather than a single item remains a unique approach even in current (W3C) multimedia recommendations.

The general model used by SMIL was that a media presentation consisted of an XML-formatted scheduling file[2] (the.*smil* presentation) and a series of external media objects. A SMIL-compliant player would interpret the presentation file and then implement a scheduling algorithm that would correspond to the needs of the presentation specification. Different players could employ different implementation strategies to meet the needs of the SMIL specification. Another innovation of SMIL was a *declarative* approach to defining the media object interactions in a presentation. SMIL is not a scripting or programming language that implements the mechanics of content delivery and playout, but a specification language that allows an author to define what they would like to have happen. It is up to the playout engine to resolve any constraints at playback time, such as lack of network bandwidth, limited screen size, or lack of interaction facilities.

The focus on media scheduling meant that SMIL was not a low-level content creation language, but a media aggregation and synchronization language. In early versions of SMIL, there was a rigid separation between content creation and content scheduling: All objects scheduled by SMIL were required to be stored in external files. Later versions relaxed this limitation, allowing plain and synchronized text to be defined directly within the SMIL file.

In 1996, the Internet was a heterogeneous environment. SMIL was defined to interact within this environment, with different user agents, different network speeds, different client screen sizes, different user language preferences, and users of differing natural abilities (in the sense that users could be deaf, blind, or otherwise differently-abled). Interoperability was a key concern of SMIL, which meant that limiting the specification to any one scripting language or any one delivery platform would not meet the language's needs. SMIL was largely successful in providing an interoperable scheduling language. Unfortunately, unlike text and images, there was—and is—no universally available set of video and audio formats that could ensure interoperability of presentation content. Early commercial and open implementations of SMIL-compliant players did not support interoperability in content codecs. As a result, a SMIL presentation written for use with the

[2]SMIL was not only an XML-compliant language, it was the *first* XML language released as a W3C recommendation.

RealNetworks G2 player could not be played with Apple's Quicktime player, even though both were early adopters of the SMIL language. This severely limited the impact of SMIL.

13.3 SMIL Presentation Basics

A SMIL presentation is an XML-formatted specification containing references to media content objects, a temporal scheduling, and synchronization model that determines when these objects are presented (and how persistent they are) and a layout model that can help a playback agent determine where the objects should be placed relative to one another. The specification also allows transitions, metadata, temporal hyperlinks, and a host of other secondary presentation features to be defined. Most of these are beyond the scope of this chapter (but are treated extensively in [6]).

In order to understand the basic structure of a simple SMIL presentation, consider the following SMIL definition[3]:

```
0)  <!DOCTYPE SMIL PUBLIC "-//W3C//DTD SMIL 3.0 Language//EN"
                          "http://www.w3.org/2008/SMIL30/SMIL30Language.dtd">
1)  <SMIL xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
2)   <head>
3)      <meta name="title" content="SlideShow"/>
4)      <meta name="generator" content="GRiNS for SMIL 3.0"/>
5)      <meta name="author" content="Dick Bulterman"/>
6)      <layout>
7)          <root-layout xml:id="Winter" backgroundColor="#ffffcc" width="240" height="269"/>
8)          <region xml:id="Title" left="0" width="240" top="0" height="29"/>
9)          <region xml:id="Image" left="0" width="240" top="29" height="180" z-index="2"/>
10)         <region xml:id="Text" left="0" width="240" top="209" height="42" fit="meet"/>
11)         <region xml:id="Buttons" left="0" width="240" top="251" height="15"/>
12)         <region xml:id="Sound"/>
13)     </layout>
14)  </head>
15)  <body>
16)     <par>
17)         <img region="Title" src="M/title.gif"/>
18)         <seq>
19)             <video region="Image" src="M/v0.mp4"/>
20)             <video region="Image" src="M/v1.mp4"/>
21)             <video region="Image" src="M/v2.mp4"/>
22)             <video region="Image" src="M/v3.mp4"/>
23)         </seq>
24)     </par>
25)  <body>
26) </SMIL>
```

[3]Line numbers have been added to simplify references in the text. They are not part of the SMIL language.

(This structure is reused later in this chapter.) Line 0 defines some XML basics for this file: the DOCTYPE and the DTD. This defines the format of the SMIL file itself, not of the presentation defined in the file. Line 1 defines the dialect of SMIL used in this file and the SMIL profile (collection of modules) that is expected to be supported by the SMIL agent processing the presentation.

Lines 2–14 provide information that the agent can use to process the file. It may contain metadata defining the name, the author, and the system used to generate the presentation. It also includes a layout specification that determines where objects are placed and their relative layering. SMIL Layout was a contentious facility: Most members of W3C wanted SMIL to use CSS layout features [9]. These gave the user agent nearly total control over (relative) object placement. The SYMM group felt that multimedia presentations were different from text content and that the semantic meaning of object placement (such as having captions overlay or be close to content) justified having a separate layout model.

Lines 15–25 define the presentation body. In this presentation, an image containing a presentation title is presented in parallel with a sequence of video objects. Both the title image and the sequence start at the same time. The members of the sequence start when their predecessor is finished.

One design goal of SMIL was that simple things should be able to be done as simply as possible, but that complex scheduling operations should also be possible without having to resort to a scripting language. In that sense, the following minimalist SMIL presentation could be used to display a simple sequence of video objects:

```
0)  <SMIL>
1)  <body>
2)      <video src="M/v0.mp4"/>
3)      <video src="M/v1.mp4"/>
4)      <video src="M/v2.mp4"/>
5)      <video src="M/v3.mp4"/>
6)  <body>
7)  </SMIL>
```

In this presentation, a default DTD and SMIL profile is used, as determined by the user agent. The agent defines a default layout structure for the presentation. The <body> element, which in SMIL defaults to the behavior of a <seq> element, is used to structure the presentation of a series of videos. The implicit duration of the videos themselves determines the length of the presentation. Granted, there is not much control on where a user agent displays the content, but the authoring overload of creating the presentation is minimal.

Finally, as a basic XML refresher:

- Each line of text between "<" and ">" characters is an XML statement.
- Each statement begins with an element name, defined by the language (in this case, SMIL).

- Each element defines a number of attributes (such as "src"). Attributes may allow attribute values to be defined (constrained by the language).

Of course, the full XML specification is slightly more complex than this summary, but this should be enough to understand the examples below.

13.4 SMIL Timing and Synchronization

SMIL timing defines when elements in a presentation get scheduled and, once scheduled, how long they will be active. The SMIL timing facilities are the core contribution of the SMIL standard: using the elements and attributes defined in the SMIL Recommendation, time can be integrated into any XML language.

In a SMIL-based document, every media object and nearly every structural element has a specific timing scope. While media timing plays an important role in determining the overall duration of a presentation, the structure of the document is also used to simplify and optimize the rendering of media presentations. SMIL timing defines a collection of elements that determine the relative start and end times of document objects and a collection of attributes that control the duration, persistence, repetition, and accuracy of timing relations in a document. SMIL can be used directly as the host language for a document (as is done in the various SMIL profiles), but it can also serve as the basis for integrating time-based coordination of otherwise static elements (as is done in SVG animation and in XHTML + SMIL).

13.5 SMIL Timing Model Basics

The timing approach used by SMIL to specify the (relative) begin times of media objects and their durations is based on a *structured timing model*. This means that the nested presentation structure in a SMIL document—and not only hard-coded clock values—is used to define the high-level activation and synchronization of objects. For many simple SMIL documents, this timing is implied: The SMIL agent can figure them out at presentation time. If more precise control over a presentation is required (such as inserting delays or specifying interactive behavior), SMIL also provides more complex timing mechanisms.

13.5.1 A Simple Slideshow Presentation

We introduce the timing issues addressed by SMIL in terms of the slideshow presentation depicted in Fig. 13.1. This presentation contains a single background

image on top of which a sequence of slides is placed, each with an accompanying image containing a text label and an audio file containing spoken commentary. The presentation also contains a single background music object that is played throughout the presentation. The timing in this presentation is dominated by two object sets: a background music object, which determines the duration of the total presentation, and various spoken commentary objects, which determine the duration of each of the image slides. This means that an outer time base for the entire presentation is defined and a set of inner time bases for each slide in the presentation.

13.5.2 Media Object and Presentation Timing Definitions

A basic property of multimedia presentations is that they require some degree of temporal coordination among the objects being presented. The more complex a presentation—in terms of either number of simultaneous objects or number of synchronization control points—the greater the amount of control information required. SMIL uses *timing elements* and *timing attributes* to provide the activation and synchronization control information in a presentation. In general, timing attributes are used to control the timing behavior of media object, and timing elements are used to control the behavior of the presentation as a whole.

13.5.2.1 Media Timing

Multimedia presentations typically contain two types of media objects: *discrete* media and *continuous* media. Discrete media objects, such as the text labels, the background image, and each of the slide images in Fig. 13.1, have no implicit duration. If referenced in a SMIL file without any additional timing attributes, their duration will be 0 s—which is not very long. *Continuous* media, such as the background music object and each of the spoken commentaries in Fig. 13.1, have implicit durations that are defined within their media encodings. If referenced in a SMIL file without any additional timing attributes, they will be rendered for the full duration defined by the object.

In Fig. 13.1, each slide image and the associated text labels should be displayed for duration that is defined by the accompanying spoken commentaries. (Each slide/text/audio group will have different durations, since not all spoken commentary is equally long.) SMIL provides a range of attributes that allow the duration of objects to be explicitly defined and refined, and it provides a general inheritance model in which the durations of both discrete and continuous media can be obtained by the context in which a media object is presented in relation to other objects. This allows the durations of the images and text to match that of the spoken audio.

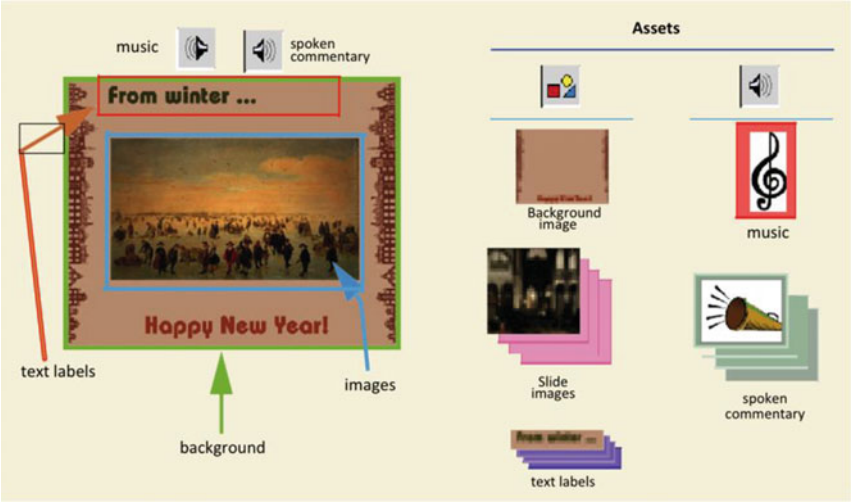


Fig. 13.1 Elements used in a slideshow presentation

13.5.2.2 Presentation Timing

A SMIL file contains references to one or more of media objects and a set of timing primitives that determine when these objects get started relative to one another. The total timing of each of the media objects, plus any additional timing control defined in the SMIL file, determines the duration of the composite presentation. Sometimes, this composite duration can be calculated in advance, but often it cannot.

The basic timing of the slideshow presentation described in Fig. 13.1 is deterministic: That is, we can determine the full timing in advance of the presentation's execution by evaluating the timing of each of the continuous media objects. It is important to understand, however, that the presentation is only deterministic if several potential presentation-time delays are ignored—these include any streaming delays associated with bringing the media object from a server to the presentation device, or any delays at the client associated with decoding and rendering individual media objects. For local presentations, such as CD-ROM multimedia, it is safe to assume that all the delays in the system can be predicted in advance and factored into the presentation timing. For Web-based multimedia, where the delays when obtaining media may be considerable (and unpredictable) and where there may be wide variability in the performance of end-user devices, assuming that presentations are fully deterministic which is a dangerous strategy.

A presentation with uncertain timing characteristics is *non-deterministic*. In addition to the network streaming delays discussed above, non-deterministic timing can also be the result of content substitution within the presentation or as a result of using interactive, event-based presentation timing. SMIL has elements and attributes to handle these cases as well.

13.5.3 SMIL and Timelines

A timeline metaphor is often used to model presentations. A timeline is a simple graph showing time on one axis and one or more media objects on the other axis. An example timeline, showing the elements and objects in Fig. 13.1, is shown in Fig. 13.2. This timeline shows the media sorted by layout: The left axis shows the various classes of media objects used, and the bottom axis shows the cumulative duration. It is also possible to define separate lines for each media object, but this is usually less space efficient.

A timeline exposes the exact temporal relationships among media items. These can translated to a text format by assigning explicit begin times for each media object and defining durations to discrete objects. One such encoding is the time-list structure of the following code fragment. The background audio and image objects (lines 0 and 1) are followed by the set of slide images (lines 2–8), the spoken commentary (lines 9–15), and the image-encoded text labels (lines 16–22). The begin times are determined by the duration of the spoken commentary objects. While this example could be used as the basis for SMIL timing, this would be unwise, since it is insensitive to delays and requires that all timing relationships be pre-calculated.

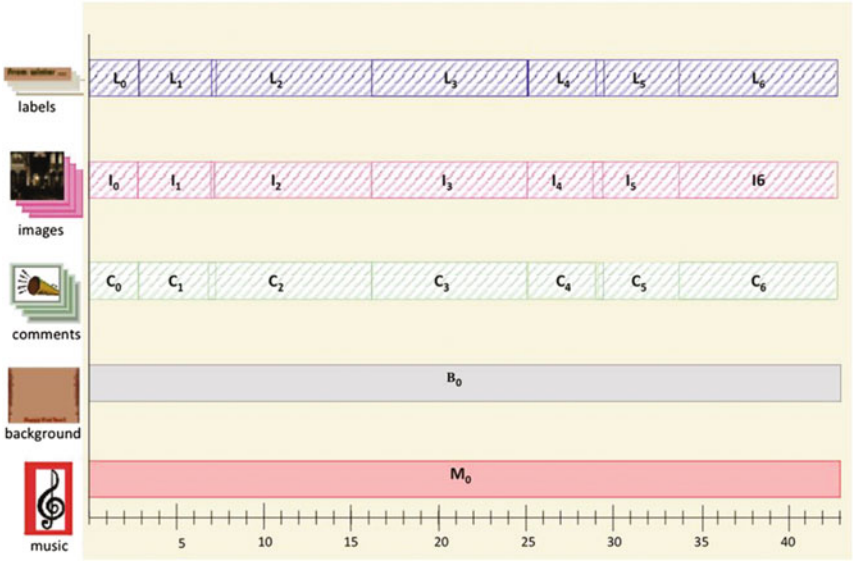


Fig. 13.2 Timeline representation of presentation in Fig. 13.1

```
0)   <audio xml:id="M0" begin="0s"    ... />
1)   <img xml:id="B0" begin="0s" dur="43s"... />
2)   <img xml:id="I0" begin="0s" dur="3s" ... />
3)   <img xml:id="I1" begin="3s" dur="4s" ... />
4)   <img xml:id="I2" begin="7s" dur="9s" ... />
5)   <img xml:id="I3" begin="16s" dur="9s" ... />
6)   <img xml:id="I4" begin="25s" dur="4s" ... />
7)   <img xml:id="I5" begin="29s" dur="5s" ... />
8)   <img xml:id="I6" begin="34s" dur="9s" ... />
9)   <audio xml:id="C0" begin="0s"    .../>
10) <audio xml:id="C1" begin="3s"   ... />
11)  <audio xml:id="C2" begin="7s"   ... />
12) <audio xml:id="C3" begin="16s"   ... />
13) <audio xml:id="C4" begin="25s"   ... />
14) <audio xml:id="C5" begin="29s"   ... />
15) <audio xml:id="C6" begin="34s"   ... />
16)  <img xml:id="L0" begin="0s" dur="3s" ... />
17) <img xml:id="L1" begin="3s" dur="4s" ... />
18)  <img xml:id="L2" begin="7s" dur="9s" ... />
19)  <img xml:id="L3" begin="16s" dur="9s" ... />
20) <img xml:id="L4" begin="25s" dur="4s" ... />
21) <img xml:id="L5" begin="29s" dur="5s" ... />
22) <img xml:id="L6" begin="34s" dur="9s" .../>
```

A better approach is to use structure-based timing primitives, since this allows timing to be deduced from the content rather than duplicating content and timing information.

13.5.4 SMIL and Structure-Based Timing

The main advantage of the timeline model is that it is an easy-to-understand representation of continuous media objects under deterministic timing conditions. As such, it is a representation used often in video and audio media editors. Unfortunately, while deterministic timing is good for modeling video tape, it does not scale well to most Web environments: If one of the image objects arrives later than planned, or if the presentation agent is slow in rendering the audio, the timeline does not really help in maintaining order among objects. Things become even more troublesome if we don't know the implicit duration of the audio items when constructing the timeline or if the duration of the object changes over the lifetime of the presentation. (Since the SMIL file does not contain the media—it contains a pointer to the media—the timing and the update histories of the media object are decoupled from its use.) Finally, if we add content substitution or interactive timing to the presentation (such as having the follow-on slide begin on a mouse click rather than at a fixed time), the timeline representation loses almost all of its utility.

In order to provide a more realistic framework for Web documents, SMIL is based on a structured timing framework in which the structured relationships among objects can be used to define most timing. SMIL encodes its timing

relationships by defining a logical timing hierarchy rather than an exact timeline. The hierarchy for Fig. 13.1 is shown in Fig. 13.3. Here, we see a set of yellow logical parallel nodes (P0–P7) and one blue logical sequential node (S0). The parallel components say activate the sub-components together as a unit, and the sequential component says activate the sub-components sequentially. The SMIL textual encoding of the presentation hierarchy is as follows:

```
0)   <par xml:id="P0" >
1)   <audio xml:id="M0" .../>
2)   <image xml:id="B0" .../>
3)   <seq xml:id="S0" >
4)       <par xml:id="P1" >
5)           <img xml:id="I0" .../>
6)           <audio xml:id="C0" ... />
7)           <img xml:id="L0" ... />
8)       </par>
9)       <par xml:id="P2" >
10)          <img xml:id="I1"  ... />
11)          <audio xml:id="C1" ... />
12)          <img xml:id="L1" ... />
13)      </par>
14)      ...
15)      <par xml:id="P7" >
16)          <img xml:id="I6"  ... />
17)          <audio xml:id="C6" ... />
18)          <img xml:id="L6" ... />
19)      </par>
20)  </seq>
21)   </par>
```

While a timeline can state that objects I_2, C_2, and L_2 all start at 7 s into the presentation and that they each have duration of 9 s, the SMIL hierarchy can state what is really going on logically:

- That objects C_i, I_i, and L_i are to be treated as a logical group that get scheduled together (that is, they begin and end together);
- That the duration of I_i and L_i depends on the duration of C_i,
- That all three objects are to begin after object C_{i-1}—and, by extension, I_{i-1} and L_{i-1}—end.

Note that none of these relationships depends on the exact duration of any of the objects—you can construct a SMIL file before you know anything about the actual media being used.

A single timeline for one instance of a SMIL specification (that is, for one of the—potentially many—run-time uses of the presentation) can be constructed by combining the structured composition of objects with a model of the execution environment that contains information on the performance of the network

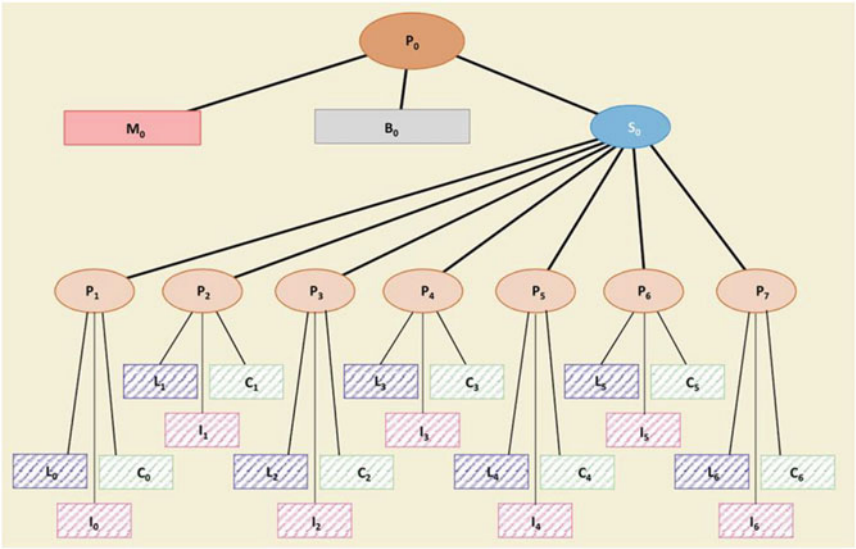


Fig. 13.3 The SMIL structured representation of the presentation in Fig. 13.1

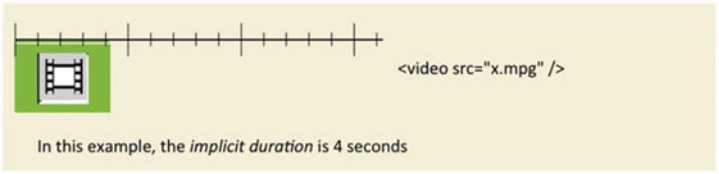
connection, the preferences of the user, etc. A timeline model based on the explicit media timings alone is not rich enough to model the various structured paths throughout a SMIL presentation.

13.5.5 Durations, Time, and Timebases

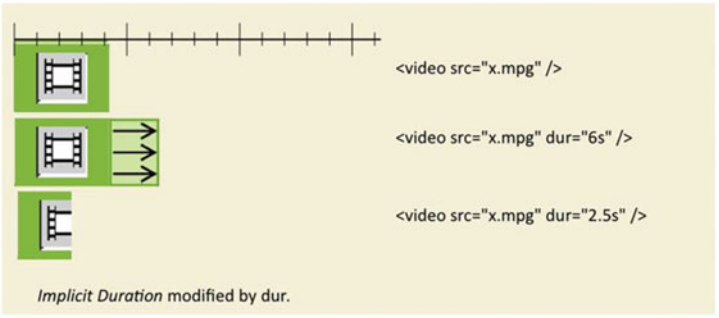
One of the most powerful features of SMIL is a flexible time model in which various aspects of an element's behavior can be determined by the context in which it is being presented. In order to use this model, it is important to understand a number of temporal distinctions and constraints applied by the SMIL model. These include defining the active period of an element and defining the way that delays and relative starting/ending times can be expressed in a document.

13.5.5.1 Defining the Active Period of an Element

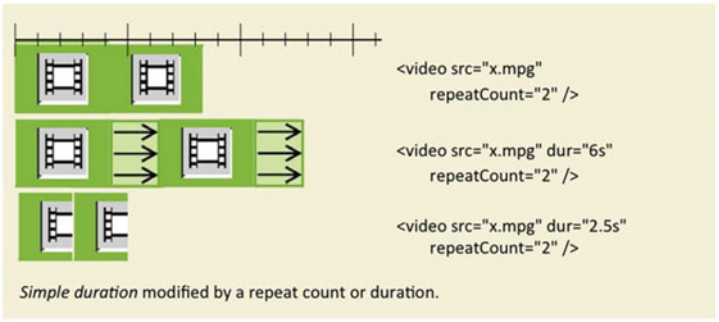
Most media formats require that the duration of all of the component media objects be explicitly defined. SMIL has several attributes that support direct duration definition, but it also provides attributes that allow you to specify or limit several layers of logical object durations. These layered durations, which are illustrated in Fig. 13.4, are as follows:



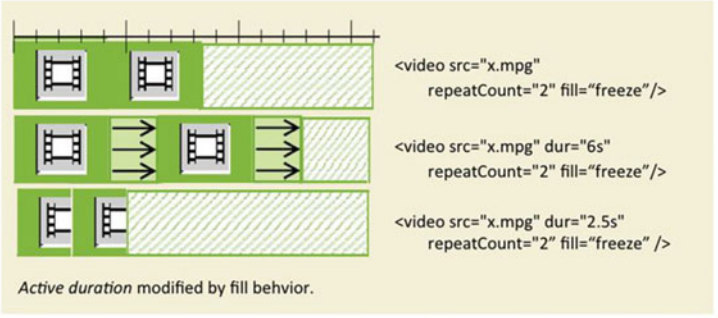
(a) Implicit Duration



(b) Simple Duration



(c) Active Duration



(d) Rendered Duration

Fig. 13.4 SMIL durations

- *Intrinsic duration*: This is the duration of a media object as encoded in the (external to SMIL) media file. Most discrete media items such as images or plain text have an intrinsic duration of 0 s; some quasi-discrete media—such as animated images—may have a longer intrinsic duration. Continuous media objects have duration that is equal to the temporal length of the object. Many media formats (but not all!) define the intrinsic duration explicitly in the media encoding.
- *Implicit duration*: This is the duration that SMIL uses as the basis for scheduling an object. It is usually equal to the intrinsic duration, if available. Discrete objects are modeled as having an implicit duration of 0 s. If the intrinsic duration for continuous objects is not available (such as often this case with MP3 objects), the SMIL agent typically will have to scan the entire object to determine its duration. (This can be a time-consuming process.) The implicit duration forms the starting point for the calculation of other logical timing durations have been defined within SMIL. Note that the implicit duration is a SMIL concept, separate from an object's intrinsic duration. See Fig. 13.4a.
- *Simple duration*: It is possible to modify an object's implicit duration with an explicit duration via SMIL's *dur* attribute. The result of applying an explicit duration (if any) to an object yields its simple duration. (If the implicit duration is not modified by a *dur* attribute, then the implicit and simple durations are the same.) The simple duration of an object may be longer or shorter than the implicit duration. Simple durations can also be defined to have special values that logically limit or stretch the duration of objects; these are the media and indefinite values, as discussed below. See Fig. 13.4b.
- *Active duration*: SMIL defines a number of attributes that allow an element to be repeated. These attributes modify the element's simple duration, and the resulting repeated duration is called the object's active duration. If an element's simple duration is shorter than its implicit duration, only the first part of the element will be repeated. If the element's simple duration is longer than its implicit duration, the entire element plus the temporal difference between the implicit and simple durations will be repeated. (During this "extra" time, either nothing will be rendered or the final frame/sample of the media object will be rendered: The behavior depends on the media *type*.) The *end* attribute can be used to define when the active duration ends. If an element does not repeat and is not shortened by end, its simple and active durations are the same. See Fig. 13.4c.
- *Rendered duration*: The active duration of an element ends after its *dur*/*end* and *repeat* attributes have been applied. This does not mean that an object disappears at the end of its active duration. SMIL provides an attribute to control the persistence of an object after its active duration has ended: the *fill* attribute. If *fill* is set to "freeze," the element will remain rendered until the end of its parent time container. If *fill* is set to "remove," the object is removed from the screen as soon as its active duration ends. For discrete media with a *fill* = "freeze" attribute, the object will simply be rendered as if its active duration was extended; for visual continuous media, the last frame or sample of the object will be rendered.

It is important to understand that each of these durations applies to every temporal element in SMIL. Every element has an implicit, simple, active, and rendered duration. Luckily, most of these durations will be managed by the SMIL agent, but understanding each of these durations, and their impact on presentation timing, can help clarify why a SMIL agent behaves the way it does.

13.5.5.2 Clock Values

Many timing attributes are based on clock values. These values can take several different forms, and they may serve as all or part of an attribute's time value. All clock values represent a relative time and have meaning only within the context of a time container.

Clock values may be given in four general forms:

- *Full clock values*: These are times represented as a colon-separated list of hours, minutes, seconds, and fractions of a second. (If days, months, or years need to be specified, then absolute wall clock timing may be used instead.) This is a relative time and has meaning only within the context of a time container's syncbase. (Syncbases are described below.)
- *Partial clock values*: These are times represented as a shorthand notation for full clock values, containing minutes, seconds, and (optionally) fractions of a second.
- *Timecount values*: These are numbers with an optional type string and an optional fractional component. An integer clock value with no type string (such as "10") implies a timecount in seconds; it is equivalent to "10 s." Allowed type strings are "h," "min," "s," and "ms."
- *Wallclock values*: These are absolute times represented in three parts: a date field, a time field, and an (optional) timezone field. These times are absolute.

13.5.5.3 Syncbases

Except in the special case of wallclock timing, every clock value in SMIL is relative to some other part of the document. The child elements of a <par> container are started relative to the start time of that parent, while the child elements of a <seq> container are started relative to the end of their predecessor (except for the first child, which starts at the beginning of its parent). Every element in a SMIL specification has a specific temporal reference point: the *syncbase*. Most elements never have to specify their syncbase reference explicitly since the common SMIL time containers do this by default. Sometimes, however, an element may want to specify a non-default syncbase as its reference object. SMIL supports this functionality using *explicit syncbase timing*. An explicit syncbase is a named element (within the same host document) that has a temporal context, and which is not a child of the element in which it is being referenced. (This is less complex than it reads.) A

syncbase timing reference contains a temporal event that is used as the scheduling base for the referencing object. This timing reference can be further modified with a clock value. In order to fully appreciate the role of syncbase timing, we first need to consider the elements and attributes defined for less complex operations, but to give a taste of what's ahead, consider the following fragment:

```
0)   <SMIL ...>
1)   ...
2)    <video xml:id="a" src="a.mpg" />
3)   ...
4)    <img xml:id="name" begin="a.begin+10s" end="a.end" src="name.png" />
5)   ...
6)   </SMIL>
```

This element on line 2 starts an associated video object. In some other part of the document, an image containing a text label is started 10 s after the video begins. The label remains visible until the end of the video.

Syncbase timing can be very complex, and its use in simple documents is rare. Still, for certain applications, it can be a powerful construct.

13.5.6 Basic Time Containers

SMIL supports the <par>, <seq>, and <excl> elements.

Element: <par>

The most general of SMIL's timing containers is the <par>. The <par> defines a local time container that can be used to activate one or more child elements. The children of a <par> container are all rendered in "parallel." In terms of SMIL's timing model, this does not mean that they get rendered at the same time, but that they share a common syncbase defined by the <par> container. Any or all of the children may be active at any time that the parent <par> is active.

The basic timing structure of the <par> is illustrated in Fig. 13.5. The default syncbase of the <par> is the beginning of the element. That is, by default, all children of a <par> element start when the <par> itself starts. (This is illustrated by nodes "a" and "b".) As defined above, timing attributes can specify other begin times. By default, the <par> ends when the last child ends, although the exact ending behavior of the <par> is determined by the endsync attribute.

Element: <seq>

A relatively unique timing container is the <seq> element. The children of a <seq> are rendered in such a way that a successor child never can begin before its predecessor child completes. In other words, the syncbase of each child is the *end*

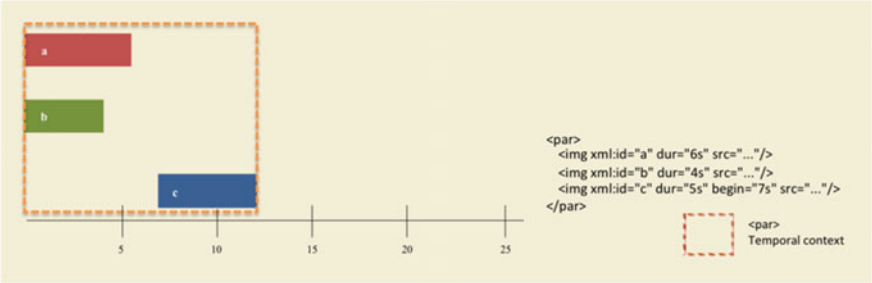


Fig. 13.5 The temporal scope of the <par> element

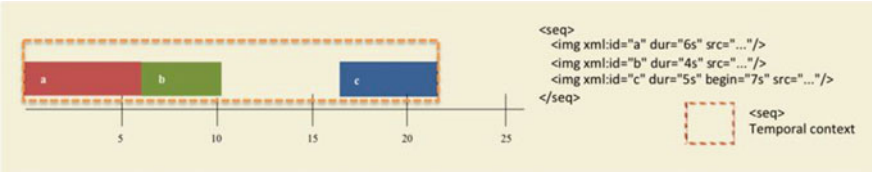


Fig. 13.6 The temporal scope of the <seq> element

of the active duration of its predecessor. A successor element may have an additional start delay, but this delay cannot resolve to be negative in relation to the end time of its predecessor. The <seq> ends when its last child has ended. Unlike the <par>, the <seq> does not support the endsync attribute: Since there is only one child active at a time in a <seq>, there is no need to select among children to determine the container's end.

The basic timing structure of the <seq> is illustrated in Fig. 13.6. The <seq> is especially useful when describing timing in a non-deterministic environment; by specifying that a set of elements logically follow one another, a delay in one element can be easily passed to its successors.

Starting with SMIL 3.0, a seemingly minor change was made to the <seq> container: The restriction that only a nonnegative offset could be used as the value for the *begin* attribute was removed from the specification. This allowed children the <seq>, just as <par> and <excl>, to have event-based starting times rather than only fixed scheduled begins. As with many seemingly simple changes, the impact of this was non-trivial.

Element: <excl>

An <excl> container will start at the temporal moment defined by its parent time container, modified by the value of the begin attribute. The begin time determination for the <excl> is identical to that of the <par>. The critical difference between a <par> and an <excl> is that at most one child of an <excl> may be

active at any one time. Nothing may violate this condition. Under certain circumstances, inactive children of an <excl> may be in a paused state and may respond to certain UI events. For a temporal perspective, they are not active during these periods.

Unlike the children of a <par>, the children have a default begin time of indefinite. This means that, unless the time gets resolved during the active duration of the <excl>, they will never begin. The children of an <excl> may make unrestricted SMIL temporal semantics to define their begin times. As with the <par>, no content may be rendered before the start of the active duration of the <excl>. If a continuous media element is scheduled to begin before the start of the parent <excl>, only that portion that falls within the temporal scope of the <excl>'s active duration will be rendered.

13.5.6.1 Dealing with Cycles and Unknown Begin Times

It is possible to define a cycle by having the begin times of a collection of objects depend entirely on the begin or end of each other. In these cases, it will be impossible to ever define a resolved begin time. Usually, none of the elements in the cycle will be activated; a SMIL player may reject to start a presentation if a cycle is detected.

If begin times are unresolved, they cannot be used to build a timing instance for the object. Elements may become resolved during the active duration of their parent, but if they are not resolved, they are ignored for timing purposes.

13.5.7 Nested Composition of Timing Elements

The basic time containers can be nested in a presentation hierarchy. That is, any child of either a <par> or <seq> (or <excl>) can be a simple media object or an embedded time container. As in SMIL 1.0, the hierarchy can represent relatively static presentation timing. The introduction of event-based activation/termination in SMIL 2.0 also allows a dynamic activation path to be defined. Most of the children of a time container will influence the timing behavior of that container, but some children—such as the <a> element, or the <switch>—are timing transparent. This means that they do not contribute to the determination of the container's duration.

13.5.8 Special Timing Values

SMIL defines two special timing values that can be used to specify the temporal context of an element. These are *indefinite* and *media*.

Value: indefinite

The *indefinite* value can be used to indicate that a timing dependency (either a begin/end time or duration) is to be determined outside of the element in which the indefinite value appears. While many SMIL authors (and some SMIL implementers!) assume that *indefinite* is synonymous with "forever," it is really synonymous with "I have no local idea …". As we will see later in this chapter, an attribute assignment of *dur* = "indefinite" simply means that some other part of the SMIL specification—either the parent time container or some other element—will determine the element's duration. Only if no other part of the document constrains the duration will an indefinite value result in an unlimited duration, but even then, a value of *indefinite* can never result in a presentation that a user (or user agent) cannot end.

Value: media

The attribute value *media* can be applied to obtain the desired timing value directly from the associated media item. Although it is not often required to use the media value explicitly (mostly because this will be the default behavior), it is sometimes useful to set duration or end time to *media* when you want to exert explicit control in complex timing situations.

13.5.9 Interactive Timing and Events

In the normal course of processing, the activation hierarchy of a SMIL document determines the rendering of document elements. The user can influence the elements selected by using SMIL events. The event architecture allows document components that are waiting to be activated or terminated to actually start or stop. There are several uses of events allowed in SMIL 2.0, but perhaps the most important new semantic introduced in SMIL 3.0 was the combination of events and the begin/end attributes. In further combination with the <excl> element, events provide a very powerful mechanism for conditional content activation.

SMIL also supports a rich hyperlinking architecture. Unlike links in XHTML, the fundamental concept of the SMIL link is that it models a temporal seek in a presentation. Rather than simply activating a target element, the target play state that is activated is identical to the state that the presentation would have been in if the target point had been arrived at "naturally." (One exception is that all intervening event-based activation is ignored.) This means that all nodes temporally between the source and destination of the link need to be evaluated to see if they would have contributed to the final target play state. The temporal seeking and activation facility allow very polished presentation construction—but its implementation in the agent is not for the fainthearted.

Figure 13.7 illustrates the temporal composition of the example presentation in Fig. 13.1. Note that the <par> and <seq> elements are nested as green and blue

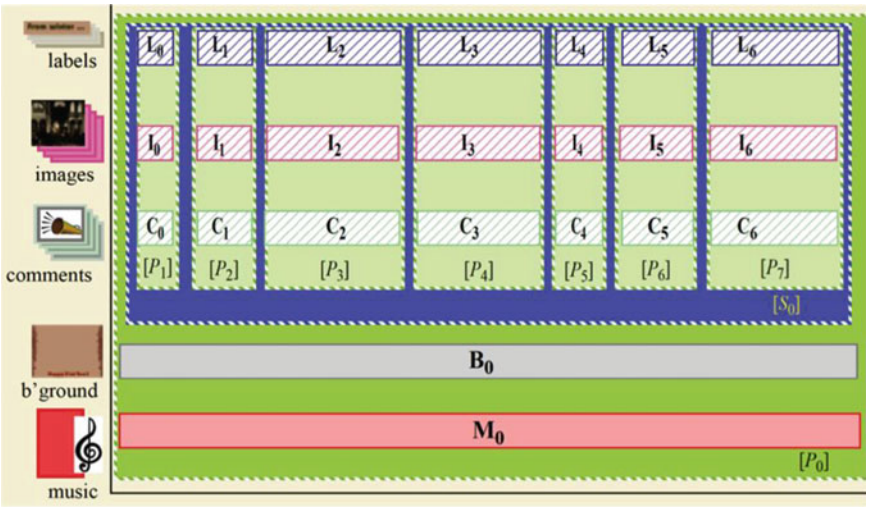


Fig. 13.7 Pure structural representation of the presentation in Fig. 13.1

boxes, respectively. (The node labels of the structure containers are placed in brackets at the bottom of each container.)

13.5.10 Applying SMIL Timing Attributes

SMIL was designed so that obvious structural relationships between elements—and the intrinsic duration of continuous media objects—could be used to specify most timing and synchronization relationships explicitly. There are many occasions, however, when the default synchronization behavior might not be expressive enough to build a particular media message. For this reason, SMIL provides a host of timing control attributes.

13.5.11 General Timing Control Attributes

The general timing control attributes supported by SMIL are *begin*, *dur,* and *end*. The *begin* attribute controls when a particular element starts relative to its syncbase, the *dur* attribute controls the element's simple duration, and the *end* attribute controls the end of the active duration. (In general, *begin* + *dur* = *end*, but this is not always true.)

All three of these attributes retain their behavior from SMIL 1.0. The principal SMIL 2.0 extension to *begin* and *end* is that they attributes support a list of *begin*/

end times instead of a single *begin*/*end* value. This is not particularly useful for simple SMIL applications, but it is very useful if a combination of SMIL's scheduled and event-based timing is used.

Attribute: begin

Each of SMIL's time containers defines a default begin time for its children. The default for <par> is the start of that <par>, while the default for a <seq> depends on the lexical order of its children: The first child starts at the start of the <seq>, and each successive child starts after the preceding child has ended. SMIL allows this default behavior to be overridden by the begin attribute. Usually, this clock value contains a nonzero positive temporal offset, such as:

```
0)  begin="10"
1)  begin="10s"
2)  begin="0:10.0"
3)  begin="0:00:10.0"
```

All of these values represent a begin delay of 10 s. Given a choice, the second version, line 2, is preferred as it gives a reasonable balance between clarity and brevity. Recall that each of these notations defines a 10-s delay from the implied syncbase of the element containing the begin attribute. (This is usually either the start of a parent <par> or the end of a predecessor child in a <seq>.)

The begin attribute and the <seq> element.

There is one restriction on the use of the begin attribute that is important for all SMIL versions: A <seq> time container may only have a single begin attribute value and that value must contain a nonnegative clock value. Multiple begin time lists are not allowed in a <seq>, and neither is explicit syncbase timing. In the following set of statements, line 4 is not valid SMIL, but lines 5 through 7 are valid and provide the same functionality as was intended in line 4:

```
0)   <par>
1)   <audio xml:id="snarf" ... />
2)   <seq>
3)       <img begin="2s" dur="5s" ... />
4)       <img begin="snarf.end+2s" ... />    <!-this is invalid SMIL! -->
5)       <par>
6)           <img begin="snarf.end+2s" ... />    <!-this is valid SMIL! -->
7)       </par>
8)   </seq>
9)  </par>
```

Attribute: dur

The *dur* attribute establishes the simple duration of an element. When used with a media object, it defines how much of that media object will be used. When used with a time container, it defines a limit on the simple duration of that container. If the defined duration is longer than the time container or media object, temporal and

rendering padding will result. The various types of duration qualifiers supported by SMIL were shown in Fig. 13.4.

Attribute: end

A SMIL element ends at the end of its active duration. This is determined by either the element's content or the parent time container. An element can also specify a non-default end time via the *end* attribute. The *end* attribute is very similar to the *begin* attribute.

The *end* attribute provides explicit control over the end of the active duration. As with *begin*, the *end* attribute can specify a single clock value or an explicit syncbase (with optional clock value offset), or a list of values. Unlike the *begin* attribute, there is no restriction on the use of *end* with <seq> elements. Note that even if multiple end times are specified, the associated element will end only once—this is when the first end condition is met.

The following fragments give examples of the use of the *end* attribute:

```
0)  <video src="one-hour-video.mpg" end="25s" .../>
1)  <video src="one-hour-video.mpg" end="1:00:15" .../>
2)  <video src="one-hour-video.mpg" end="snarf.end" .../>
3)  <video src="one-hour-video.mpg"end="35s; snarf.end+5s" .../>
```

The first two of these statements have obvious behavior. Line 2 says that the video will end whenever the temporal element *snarf* ends. (This is an example of syncbase timing.) Line 3 states that the video will end at either 35 s after the start of its rendering or 5 s after the end of the syncbase element *snarf*.

Combined Use of begin, dur, and end

It is possible to combine the use of the *begin*, *end,* and *dur* attributes on an element. Combining *begin* and *dur* usually results in obvious behavior, since they both relate to the simple duration of an object. Since the *end* attribute overrides the default end of the active (not simple) duration, the resulting behavior is not always obvious.

13.5.12 Object Persistence Attributes

Two attributes control the persistence of objects after they have completed their active duration: *fill* and *fillDefault*. In this chapter, we consider simple uses of *fill* behavior.

Attribute: fill

SMIL allows an object to remain visible after the end of its active duration by specifying a *fill* attribute. If *fill* is set to "freeze," the object will remain visible until the end of its parent time container. If it is set to "remove," the object is removed as soon as its active duration ends. The default for visual media is effectively "freeze"; *fill* has no meaning for audio media. There are some special cases for *fill* that allow

the visibility of an object to extend past the active duration of its parent (such as when transitions exist). The value of *fill* also determines how long an object remains "clickable" for linking and interaction.

If the *fill* attribute is applied to a time container, then the rendering state of all of the children (and descendants of the children) is set to their respective fill behavior at the moment the active duration of the container ends.

13.5.13 Extended Timing Control Attributes

SMIL 2.0 introduced two attributes that provide extra duration control: *min* and *max*. These attributes can be used to define a lower or upper bound on the active duration of the containing element, regardless of that element's other timing characteristics. SMIL also supports the ending of a time container via the *endsync* attribute.

Attribute: min

The *min* attribute can be used to specify an explicit minimum active duration on an element. It accepts either a clock value or "media" as values. The default value of *min* is "0," which is the same as saying that the value is unconstrained. The value *media* may only be used on media objects; its use says the minimum duration is the implicit value of the media object.

Attribute: max

The *max* attribute can be used to constrain the maximum active duration of an element. It is similar to *min*, except that an explicit value of indefinite can be specified. (This is also the default.)

The behavior of the *max* attribute is predictable, although finding relevant use cases can be a challenge. For example, consider the following statements:

```
0)  <videoxml:id="a"src="one-hour-video.mpg"max="30s"end="bazinga.end".../>
1)  <video xml:id="b" src="one-hour-video.mpg" end="30s; bazinga.end" ... />
```

Line 0 states that the element *a* will play for a maximum of 30 s or until the element *bazinga* ends (if this is earlier). The same behavior applies to video element *b* (line 1).

Attribute: endsync

Where *min* and *max* provide clock value-based constraints on element timing, SMIL provides the *endsync* attribute to provide logical control on timed objects. As in SMIL 1.0, if a <par> has multiple children, it can specify that the entire <par> ends when the first child ends, when the last child ends, or when a named child ends using the *endsync* attribute. This attribute can be assigned to the <par> and <excl> time containers.

The following fragment illustrates endsync behavior:

```
0)   <par endsync="snarf" >
1)       <video xml:id="bazinga" src="video.mpg" ... />
2)       <audio xml:id="snarf" src="audio.mp3" ... />
3)   </par>
```

The <par> ends when the *snarf* ends; if this is later than the end of *bazinga*, the last frame of *bazinga* will be frozen. The default value of *endsync* is "last." An element with the assignment *endsync* = "all" waits for all of its children to play to completion at least once, regardless of whether they have scheduled or interactive begin times.

The behavior of the *endsync* attribute is intuitive, except when combined with the *dur* and/or *end* attributes. This is because having both an *endsync* and a *dur* or *end* will cause conflicting definitions of element duration. In order to resolve these conflicts, the following rules are applied by a SMIL agent:

- If *dur* and *endsync* are specified on the same element, the *endsync* is ignored.
- If *end* and *endsync* are specified on the same element, the *endsync* is ignored.

13.5.14 Repeating Objects and Substructures

The default behavior of all elements is that they play once, for either an implicit or explicit duration. The SMIL *repeatCount* attribute allows an iteration factor to be defined that acts as a multiplier of the object's simple duration. (The resulting duration is the active duration.) A special value *indefinite* is used to specify that an element repeats continually until its (parent) timing context ends. The *repeatDur* attribute defines duration for all of the repeated iterations.

Attribute: repeatCount

The *repeatCount* attribute makes the element's content sub-presentation or media object to repeat the stated number of times. Its value can be either a number or *indefinite*. A numeric value indicates the media object plays that number of times. Fractional values, specified with decimal points, can be used as well.

The following statement specifies that *snarf* is to be repeated 4.5 times:

```
0)   <audio xml:id="snarf"  src="audio.mp3" repeatCount="4.5" ... />
```

If the associated audio file was 4 s long (its implicit duration), then the active duration of *snarf* is 18 s.

A repeat count may also be applied to time containers. As is shown in the following fragment, the simple duration of the container is repeated for the number of iterations specified by the repeatCount.

```
0)   <par repeatCount="3" >
1)       <video xml:id="bazinga" src="video.mpg" ... />
2)       <audio xml:id="snarf" src="audio.mp3" ... />
3)   </par>
```

In this example, the active duration will depend on the implicit durations of the media items; the simple duration will be the maximum of *bazinga* and *snarf*. The active duration will be the simple duration times three.

Attribute: repeatDur

The *repeatDur* attribute is similar in most respects to *repeatCount*, except that duration is given instead of a multiplier. It states that the element is to repeat its playback until the specified duration has passed. The following statement specifies that *snarf* is to be repeated for 15 s:

```
0)   <audio xml:id="snarf"     src="audio.mp3" repeatDur="15" ... />
```

The actual number of times that the audio object is repeated depends on its duration. If the audio object's implicit duration is 3 s, it will repeat times. If its implicit duration is 15 s, it will be played once. If its implicit duration is 25 s, only the first 15 s will be played.

As with all elements, a timing constraint on a parent container will limit the active duration of the children. As a result, in the following fragment, the audio object will be rendered for 25 s:

```
0)   <par dur="25s">
1)       <audio xml:id="snarf" src="audio.mp3" repeatDur="100s" ... />
2)   </par>
```

13.5.14.1 Combing Repeat Behavior with Other Timing Attributes

It is possible to combine either *repeatCount* or *repeatDur* with other timing attributes. The resulting timing usually results in predictable behavior for simple cases, but sometimes the differences in manipulating the simple, active, and rendered durations within one element can lead to SMIL statements that are not always easy to understand.

The composite behavior is nearly always clear if the differences between the various types of SMIL durations are well-understood. This becomes vital if use is made of interactive behavior in SMIL, or if multiple begin times on objects are used. If you plan to use SMIL Animation—or if the SMIL repeat attribute is used—then a complete understanding of the SMIL timing model is essential. Interested readers are encouraged to consult the standard text on SMIL [6]. As a last resort, the SMIL specification can be consulted [x]: There you will find 156 pages of information on the timing model alone.

13.6 Advanced Timing and Synchronization Attributes

SMIL provides a collection of advanced attributes to control synchronization behavior and to facilitate integration of SMIL into other XML languages. While these attributes are rather esoteric for a complete discussion here, we introduce them briefly in the following sections.

13.6.1 SMIL Synchronization Control Attributes

In a perfect world, all of the defined timing in a specification would be implemented perfectly by a SMIL agent. Unfortunately, the world is not only imperfect—it is also unpredictable. In order to provide some measure of control in the face unpredictably, SMIL provides three high-level synchronization control attributes: *syncBehavior*, which allows a presentation to define whether there can be slippage in implementing the composite timeline of the presentation; *syncTolerance*, which defines how much slip is allowed; and *syncMaster*, which allows a particular element to become the "master" timebase against which all others are measured.

13.6.1.1 XML Integration Support

When used in a native SMIL document (one in which the outer XML tag is <SMIL>), the nature and meaning of various timing elements is clear. When integrating SMIL timing into other XML languages, a mechanism is required to identify timing containers. The SMIL specification does this using the *timeContainer* and *timeAction* attributes.

13.7 Summary and Conclusion

Support for a general model for timing and synchronization SMIL's most important contribution. The basic time container elements <seq> and <par> specify SMIL's simplest temporal relationships: those of playing in sequence and playing in parallel. The basic in-line timing values of the *inline* timing attributes *begin*, *end,* and dur set an element's timing as temporal offsets from the begin time it gets from its time container parent. Syncbase values for the *begin* and *end* attributes synchronize an element with the beginning or ending of other elements. The attributes *repeatCount* and *repeatDur* cause an element's media object to repeat its playback. If, after its start time, duration, and repeated duration, the element ends before the end of its parent, the *fill* attribute defines the persistence of the element in SMIL's timing tree.

All of the various options for timing and synchronization control have given SMIL the reputation—in some circles—to being overly complex. Certainly for developers of HTML5 [10], the impression existed that simpler was better. This reputation is largely undeserved. In SMIL, very simple facilities exist for crafting complex presentation based entirely on implicit timing control. The value of SMIL is that complexity is available when it is needed without having to resort to scripting. SMIL also provides a unified timing control model in which all elements of a document have a common temporal basis. This "complexity" actually simplifies the task of creating interactive presentations.

This book summarizes a number of timing formats that have been applied to controlling content across the World Wide Web. SMIL was the first of these formats to be captured in a set of standards that, for a long time, formed the basis of timing in Web documents. There have been many uses of SMIL in embedded applications, and nearly every smartphone on the planet carries a partial SMIL implementation. Still, in terms of mass adoption at the end-user level, SMIL has never been a great success. This has little to do with SMIL's timing model or its use of structure-based timing. SMIL has suffered from a lack of universally accepted media codecs for video and audio content. This has crippled any hope of interoperability. The irony of this situation is that such interoperability forms one of the core potential successes of SMIL, at least as far as the development of an abstract timing model is concerned.

References

1. W3C, World Wide Web Consortium. http://w3.org/
2. Bugaj, S., Bulterman, D.C.A., Butterfield, B. et al.: Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, June 1998. https://www.w3.org/TR/1998/REC-smil-19980615/
3. Ayers, J, Bulterman, D.C.A., Cohen, A. et al.: Synchronized Multimedia Integration Language (SMIL 2.0), 2nd edn., Jan 2005. https://www.w3.org/TR/2005/REC-SMIL2-20050107/
4. Bulterman, D.C.A., Grassel, G., Jansen, J. et al.: Synchronized Multimedia Integration Language (SMIL 2.1), Dec 2005. https://www.w3.org/TR/2005/REC-SMIL2-20050107/
5. Bulterman, D.C.A., Jansen, J., Cesar, P.S. et al.: Synchronized Multimedia Integration Language (SMIL 3.0), Dec 2008. https://www.w3.org/TR/SMIL/
6. Bulterman, D.C.A., Rutledge, L.: SMIL 3.0: Interactive Multimedia for the Web, Mobile Devices and Daisy Talking Books. Springer (2008)
7. W3C Synchronized Multimedia Working Group. https://www.w3.org/AudioVideo/
8. Bulterman, Dick C.A.: Specification and support of adaptable networked multimedia. Multimed. Syst. **1**, 68 (1993). https://doi.org/10.1007/bf01213485
9. Bos, B. et al.: Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, June 2011. https://www.w3.org/TR/CSS2//
10. Hickson, I. et al.: HTML-5: A vocabulary and associated APIs for HTML and XHTML, October 2014. https://www.w3.org/TR/html5/

Chapter 14
Specifying Intermedia Synchronization with a Domain-Specific Language: The Nested Context Language (NCL)

Marcio Ferreira Moreno, Romualdo M. de R. Costa and Marcelo F. Moreno

Abstract This chapter reports on the intermedia synchronization features of Nested Context Language (NCL), an XML-based domain-specific language (DSL) to support declarative specification of hypermedia applications. NCL takes media synchronization as a core aspect for the specification of hypermedia applications. Interestingly, NCL deals with media synchronization in a broad sense, by allowing for a uniform declaration of spatiotemporal relationships where user interactivity is included as a particular case. Following the W3C trends in modular XML specifications, NCL has been specified in a modular way, aiming at combining its modules into language profiles. Among the main NCL profiles are those targeting the domain of Digital TV (DTV) applications. Indeed, NCL and its standardized player named Ginga are part of ITU-T Recommendations for IPTV, Integrated Broadcast–Broadband (IBB) and DTV services, and Integrated Services Digital Broadcasting—Terrestrial (ISDB-T) International standards. This chapter discusses the main reasons that make NCL a comprehensive solution for the authoring of interactive multimedia applications. It also discusses the aspects of its conceptual model, the Nested Context Model (NCM), which defines an intrinsic support for easily specifying spatiotemporal synchronization among components (e.g., media and input assets).

Keywords NCL · NCM · Ginga · Multimedia applications
Intermedia synchronization

M. Ferreira Moreno (✉)
IBM Research, Rio de Janeiro, RJ, Brazil
e-mail: mmoreno@br.ibm.com

R. M. de R. Costa · M. F. Moreno
Department of Computer Science, UFJF, Juiz de Fora, MG, Brazil
e-mail: romualdo@ice.ufjf.br

M. F. Moreno
e-mail: moreno@ice.ufjf.br

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_14

14.1 Introduction

This chapter reports on Nested Context Language (NCL), an XML-based domain-specific language (DSL) to support declarative specification of hypermedia applications. Due to the NCL features, intermedia synchronization, one of the most important QoS issues in document engineering and multimedia data communication, has been achieved in some important environments including interactive Digital TV (DTV) systems and the Web.

Declarative languages emphasize the declarative description of an application, rather than its decomposition into an algorithmic implementation. Such declarative descriptions generally feature a higher-level specification, and thus, they are easier to be designed than imperative ones, which usually require a programming expert. Moreover, declarative languages can be designed with a focus on the spatial and temporal synchronization among components of the application. This is the case of NCL, which takes media synchronization as a core aspect for the specification of hypermedia applications. Interestingly, NCL deals with media synchronization in a broad sense, by allowing for a uniform declaration of spatiotemporal relationships where user interactivity is included as a particular case.

With a stricter separation between application content and structure, NCL does not restrict which type of content or interactive devices can be used in an application. Instead, it defines the glue that holds components together in multimedia presentations. The definition of which content type of media objects are supported, and which input devices, depends on the components integrated into the NCL formatter (the NCL Player). NCL Players by themselves are unable to display media objects, but they are built to maintain temporal and spatial synchronization of these objects, or to support any other event-oriented relationship specified among media objects, as specified by the NCL application. To display media objects, NCL Players rely on embedded media players. Therefore, NCL allows for the specification of spatiotemporal relationships using multimodal input interfaces and different perceptual contents such as videos, images, audios, and texts, as NCL media objects. Applications specified using other languages can also be described as NCL media objects, including media objects referencing imperative code content (e.g., Lua, ECMAScript) or declarative code content (e.g., HTML-based, X3D, SMIL, NCL). Following the W3C trends in modular XML specifications, NCL has been specified in a modular way, aiming at combining its modules into language profiles. Among the main NCL profiles are those targeting the domain of DTV applications. Indeed, NCL and its standardized player named Ginga are part of ITU-T Recommendations for IPTV, Integrated Broadcast–Broadband (IBB) and DTV services, and Integrated Services Digital Broadcasting—Terrestrial (ISDB-T) International standards. This chapter discusses the main reasons why NCL is a comprehensive solution for the authoring of interactive multimedia applications, which are the aspects of NCL intrinsic support for easily defining spatiotemporal synchronization among components (e.g., media and input assets). In this context, this chapter presents the NCL conceptual model, discussing the NCL features, from its support

to IBB systems, to how it can handle multiple time-bases in supporting intermedia synchronization with or without interleaved media content, and its support for multi-device presentations.

Current DTV systems put together different services, as a consequence of the transport platform convergence. Efforts in this direction are reported in ITU-T H.760 Recommendation series (Multimedia Application Frameworks), where the NCL is highlighted as the integration mechanism with its Ginga-NCL Player.

Regarding the time-bases, NCL offers an approach that provides high-level abstractions that hide or minimize the complexity of dealing with multiple time-bases. In NCL intermedia synchronization, events are related regardless of the moment they occur. Events are associated with specific segments (called anchors) of some content (e.g., segments defined by their beginning and ending samples in the content stream). The start, pause, resume, end, and abort of an anchor presentation define synchronization points (instantaneous events) as well as the selection of an available anchor. Synchronization points can be used in causal or constraint relationships to define intermedia synchronization. For example, one can specify that the start of a media object presentation can cause the start of another media object presentation (causal relationship), or that two media objects must always stop at the same time (constraint relationship). Several hypermedia languages, such as Nested Context Language (NCL) [1] and Synchronized Multimedia Integration Language (SMIL) [2], allow for defining causal and constraint relationships.

The multiple device support provided by some languages to allow distributed multimedia presentations still deserves attention. NCL's approach provides a higher-level multi-device presentation specification by defining classes of devices and allowing applications to distribute their content among these classes. The glue-language characteristic of NCL allows for synchronizing the life cycle of applications specified as NCL content nodes but implemented using distinct languages and presented on specific classes of devices. For example, one can have a distributed NCL application running part of its component on devices with SVG support, part on devices with Java support, part on devices with BML or any other XHTML-based language support, etc.

In order to support the aforementioned NCL features, Ginga architecture and its reference implementation also brought contributions to several issues regarding the support to intermedia synchronization. Hence, Ginga and its Hypermedia Temporal Graph data structures are also subjects of discussion in this chapter.

14.2 Related Work

Several languages, models, and systems are available to support a wide range of services related to hypermedia applications. These services can include media production and analysis, presentation as well as broadcasting and delivering on demand. Although many services are implemented as commercial tools with

proprietary formats, standard groups have worked to propose solutions with valuable services for the authors who want to produce hypermedia presentations with all mentioned services.

Synchronized Multimedia Language (SMIL) [3] and Moving Picture Experts Group (MPEG) [4, 5] are relevant technologies related to the specification of intermedia synchronization in hypermedia applications. While SMIL is a W3C standard that enables authoring multimedia applications in the Web, MPEG has a superset of standard technologies focused on multimedia representation, distribution and presentation.

Like NCL, SMIL offers a declarative form where through its elements it is possible to specify hypermedia applications. By using no more than SMIL elements, it is possible to specify structured hypermedia applications composed by media objects with a spatial layout and a temporal behavior. Besides this, some mechanisms [6] have been proposed to add support in SMIL of many facilities described in this paper. However, this approach often produces nonstandard solutions.

In the MPEG superset, the MPEG-4 System is the standard that allows applications specification using a declarative language called XMT-O [4]. There are also other formats specified in this standard, each one focused on a specific multimedia service. For instance, to control the presentation distribution, from servers to receivers, a format called Binary Format for Scenes (BIFS) [4] is defined. Complementarily, for applications to be presented in multiple devices, Lightweight Application Scene Representation (LASeR) [5] seems to be the best choice, since it defines a format most suitable to mobile devices, where the scene structure can be fragmented in many access units, which can be, at least in theory, streamed to and presented by distinct players. However, this approach is not standardized.

LASeR follows the SVG [7] scene structure, another W3C standard, composed by elements in an XML format. In [8], authors propose a time fragmentation of SVG documents to control the playback memory usage. Indeed, this idea could also be used to develop some experiences over SVG and LASeR in multi-devices. Although these languages are quite promising, they do not include yet, in a standard way, many features presented in this paper.

In the field of Integrated Broadcast–Broadband (IBB) systems [9], there are two other technologies that are compliant with IBB requirements.

The Hybrid Broadcast–Broadband TV (HbbTV) [10] combines TV services delivered via broadcast with services delivered via broadband and also enables access to Internet-only services for consumers using connected TVs and set-top boxes. HbbTV specification 1.5 is based on existing standards and Web technologies (XHTML 1.0, DOM 2, CSS 1, ECMAScript). It is a proposal from ETSI that succeeds its former middleware specification called Multimedia Home Platform (MHP) for interactive digital television. HbbTV is a whole new specification that is not compatible with MHP. IBB and non-IBB services can coexist in a DTV system that adopts HbbTV by properly signaling. HbbTV 2.0 recently added support for some kind of media synchronization, companion devices, and new media types,

a set of features that were not supported in previous versions. HTML5 became the base language included in HbbTV 2.0.

Hybridcast [11] uses HTML5 and is standardized in Japan as a second generation of multimedia coding specification that succeeds the first one, based on BML. The specification is completely new, but hybridcast receivers can present BML applications. The system offers a variety of services through a combination of broadcast and broadband telecommunication resources and features. Features also include basic support for companion devices and synchronization via stream events only.

14.3 NCL

The Nested Context Model (NCM 3.0) is the model underlying NCL. However, in its present version 3.1, NCL does not reflect all NCM definitions. In order to understand NCL synchronization facilities in depth, it is necessary to understand the NCM concepts introduced in the next section.

14.3.1 The Nested Context Model (NCM)

The foundation of NCM is the usual hypermedia concepts of nodes and links. The former represents content fragments, while the latter has the purpose of defining relationships among interfaces (anchors and properties) of nodes. This section discusses only the NCM entities and attributes that are of interest of this chapter, that is, the entities that allow the specification of spatiotemporal relationships among node interfaces. Soares and Rodrigues work [12] brings a detailed discussion about the other NCM entities and attributes.

Figure 14.1 presents the NCM class hierarchy focusing on the *Node* entity. Every NCM *Entity* has a unique identifier (id) and other attributes [12]. There are two basic classes of nodes: content node and composite node.

A *ContentNode* represents the usual media objects. *ContentNode* subclasses define the content type (e.g., video, audio, image, text). To define its content, a *ContentNode* can use a reference (e.g., URL) to the content or have a byte array of the content (raw data).

A *CompositeNode* is an NCM *Node* whose content is a set of nodes (composite or content nodes). The set of nodes constitutes the composite node information units. In a *CompositeNode*, a *Node* cannot contain itself. *CompositeNode* subclasses define semantics for specific collections of nodes. A *ContextNode* is an NCM *CompositeNode* that also contains a set of links and other attributes [12]. Context nodes are useful, for instance, to define a logical structure for hypermedia documents.

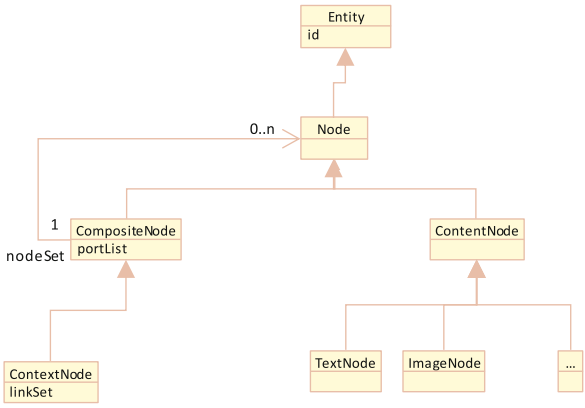


Fig. 14.1 NCM class hierarchy: the node entity

A *Link* is an NCM *Entity* with two additional attributes: a *Connector* and a set of *Binds* to this connector. Figure 14.2 presents the NCM class hierarchy focusing on the *Link* class.

Connector is an NCM *Entity* that defines the semantics of a relation, independently of the components (nodes) that will be included in the relation [12]. In this sense, distinct links representing the same type of relation, but connecting different components, may reuse the same connector. A *Connector* has as attribute a set of access points, called roles. A *Link* refers to a connector and a *bindSet*. Each *Bind* associates each link endpoint (a node component and its interface) to a role at the referred connector.

Figure 14.3 shows an NCM example with three links referring to two different connectors, each connector representing a relation with two different roles (or two different semantics for the included components). Each *Bind* present in the *bindSet* of each *Link* specifies the participant components (nodes). In the example of Fig. 14.3, "link1" defines two binds connecting "movie1" and "movie2" content nodes. "link2" refers to the same connector of "link1", but its binds connect "movie2" and "movie3" content nodes. "link3" will be discussed further in this section. Both, "link1" and "link2", specify different relationships (connecting

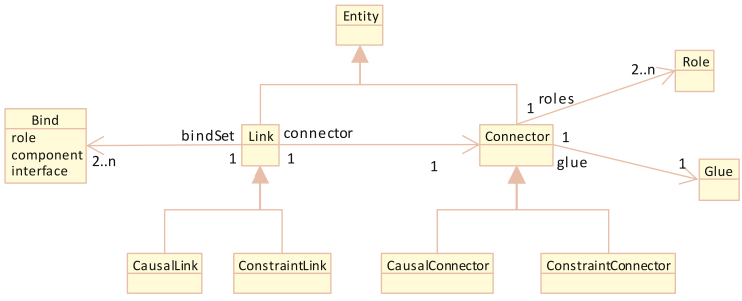


Fig. 14.2 NCM class hierarchy: the link entity

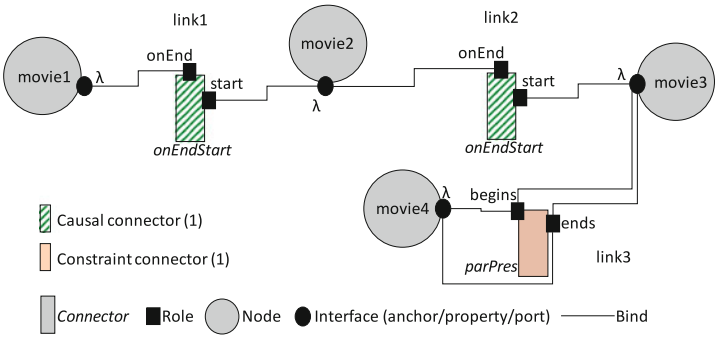


Fig. 14.3 Using causal and constraint relations

different nodes), but represent the same type of relation (referring to the same connector).

As aforementioned, a bind has as endpoint a component (node) and an interface of this component. In NCM, an interface can be an anchor, a property, or a port, which can be defined only in a composite node and specifies its entry points. An anchor can represent an interval in time of the content referred by the node, or an area of the content of a node being presented. Note, in Fig. 14.3, that all relationships endpoints are interfaces with the λ symbol, which means the relationship with an anchor representing the whole node content (an interval in time defined by the node begin and end presentation instants).

In its current version (NCM 3.0 [12]), an NCM *Connector* only allows the specification of relations with causal or constraint semantics. On the one hand, a condition must be satisfied in a causal relation to execute a group composed of one or more actions. Figure 14.3 presents the connector *onEndStart*, which is an example of causal relation that can start (role start) the presentation of one or more nodes when the presentation of one or more nodes finishes (role *onEnd*). On the other hand, on constraint relations there is no causality involved. Figure 14.3 presents an example of a *ConstraintConnector*, identified as *parPres*. In the example, "link3" refers to the *parPres* connector that defines that two or more nodes must begin their presentation at the same time and must end their presentation at the same time. The occurrence of the presentation of one node without the occurrence of the presentation of the others also satisfies the constraint, which specifies that, if and only if the presentation of these nodes occurs, their beginning and ending times have to coincide.

Returning to the class hierarchy in Fig. 14.2, note that connectors are specialized in causal and constraint subclasses. A *CausalLink* defines a causal relationship, which means it refers to a causal relation (*CausalConnector*). Similarly, a *ConstraintLink* refers to a *ConstraintConnector*. It is a set of roles organized in a glue that defines the semantics of the relation (connector) and, therefore, the semantics of the relationship (link). Both, role and glue, are discussed in the next paragraphs.

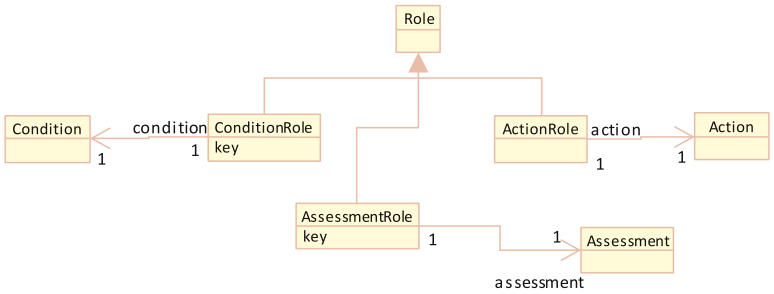


Fig. 14.4 NCM class hierarchy: the role class

The concept of event[1] is the foundation of the *Role* class. Therefore, each role describes an event to be associated with a component of the relation. Figure 14.4 shows the NCM class hierarchy for the *Role* class. There are three subclasses of *Role*: *ConditionRole*, *AssessmentRole*, and *ActionRole*. Each connector type can use a different set of roles. Causal connectors can use the aforementioned three types of role, while constraint connectors only use assessment roles.

A *ConditionRole* represents a condition, defining a logical expression, which evaluates presentation events (e.g., when the presentation of a node begins), attribution events (when a property value of a node is set, such as set the "width" property to "50px"), or node property values (e.g., evaluating if the "height" property has a value greater than "200px"). When evaluated, a condition returns *true* if it was satisfied, executing the related *ActionRoles* (e.g., "stop" the presentation of a node), or *false* otherwise. While a condition always returns a *Boolean* value when evaluated, an *AssessmentRole* returns a value (the current state of an event or a value of a node property). Condition and assessment roles have the attribute *key* to represent input events from interaction devices. For instance, when the user interacts with an interface element, the condition is satisfied and a group of actions is executed.

Figure 14.5 presents the NCM class hierarchy for the *Glue* class. A *Glue* describes how roles must interact and must consider the use of all roles in the connector. A *ConstraintGlue* (in a constraint connector) has a *StatementExpression* relating assessment roles. A *CausalGlue* (in a causal connector) specifies a *TriggerExpression*, relating condition expression (*ConditionExpression*, which has condition roles), assessment statement (*AssessmentStatement* or *AssessmentValueStatement* with assessment roles), and action expression (*ActionExpression* with action roles). Satisfying the conditions and assessments of a trigger expression causes the execution of action roles.

Statement expressions allow for the comparison between two assessment roles using an *AssessmentStatement* (for instance, if the presentation property "width" of a node is equal to a presentation property "height" of another node) or of an

[1]NCM uses the definition of event as stated in the Pérez-Luque and Little work [13]: An event is an occurrence in time that may be instantaneous or may extend over a time interval.

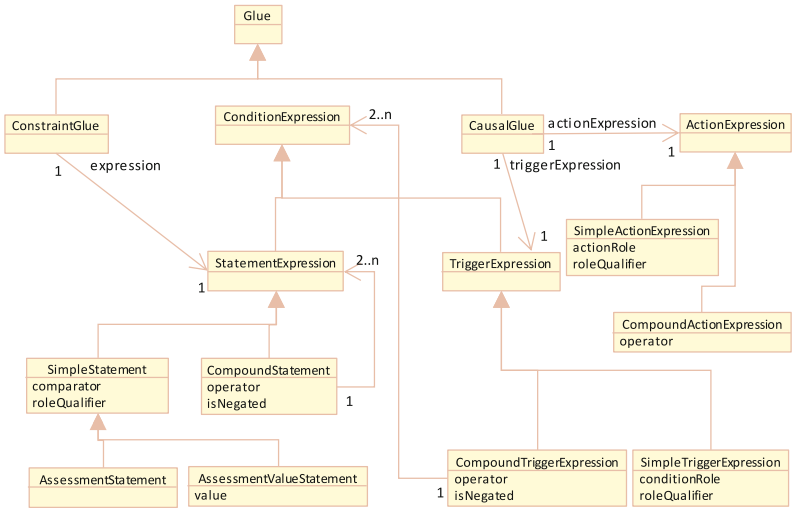


Fig. 14.5 NCM class hierarchy: the glue class

assessment role to a value using an *AssessmentValueStatement* (e.g., comparing "height" with the value "500px").

Trigger expressions can relate any number of conditions and assessment roles using, respectively, condition and statement expressions. In addition, they can relate actions (for instance, the presentation action "start" shown in Fig. 14.3). Statement, trigger, and action expressions can be simple or compound. Simple expressions refer to a role of its type, while compound expressions may relate any number of roles of its type using a logical *operator* ("and" or "or"). Compound trigger expression and compound statements can be negated (attribute *isNegated*) to change the logical result of the expression.

Simple statement, trigger, and action roles have attributes (*max* and *min*) to specify the maximum and minimum cardinality. For instance, a role with "max = unbounded" allows for the definition of an n-ary relation, since it can be bounded to an unlimited number of participants. Finally, these types of roles also have the attribute *roleQualifier* to specify the role behavior when more than one participant uses the same role. For instance, when a role "start" relates two nodes, the role qualifier specifies if the two nodes must be started in parallel (value "par"), following a sequential order ("seq"), or if only one of the nodes shall be started ("excl").

14.3.2 Describing Temporal Relationships with NCL

The basic NCL structure defines the root element, called <ncl>, and its children elements, the <head> element and the <body> element, following the terminology adopted by W3C standards.

The <head> element may have elements that represent repositories of NCM entities as its children. For instance, a connector base can be specified to gather NCM connectors. The <body> element is treated as an NCM context node. Therefore, links (element <link>), content nodes (<media> element), and context nodes (<context> element) can be defined as child of the <body> element. The <media> element defines a media object specifying its content location. Its content type can be specified using an element attribute. Otherwise, the content type is inferred using the content or its URL.

Listing 14.1 presents an example of an NCL document specifying causal and constraint relationships. The example is illustrated in Fig. 14.6. Its idea is to present an episode of a TV series from a broadcast channel. During the presentation of this episode, the user can select an icon to see a recap from an on-demand broadband service. In this case, a second icon can be selected to play a thematic game of the TV series.

In the NCL document of Listing 14.1, port "bp1" (line 23) specifies that the document presentation begins with the presentation of the media object "got" (lines 23–27). In fact, the <port> element of a context (note that the <body> element is the main context of an NCL document) allows externalizing an interface of any of its internal child objects. The object "got" has two interval anchors: "anchor1" and "anchor2" (lines 25 and 26).

The anchors specified with the <area> element allow for the definition of a content anchor representing a spatial, temporal, or spatiotemporal segment of a media object's content. Details about both anchors will be discussed later in this section. Link "bl1" (lines 45–48) refers to the causal connector "onBeginStart" (lines 4–7), which defines the condition role "onBegin" and the action role "start" with unbounded max cardinality. The link specifies that when "anchor1" of "got" begins, the presentation of media object "img1" (line 29) shall be started. Link "bl2" (lines 49–52) refers to the causal connector "onEndStop" (lines 8–11) to specify the following relationship: When "anchor1" of "got" ends, "stop" the presentation of "img1". Links "bl3" and "bl4" (line 53) were omitted in Listing 14.1 and specify similar relationships, but binding to "anchor2" of "got" instead of "anchor1" to start and stop the "img2" media object instead of "img1". Link "bl5"

Fig. 14.6 NCL presentation

```
<ncl>
 <head>
  <connectorBase>
   <causalConnector id="onBeginStart">
    <simpleCondition role="onBegin"/>
    <simpleAction role="start" max="unbounded"/>
   </causalConnector>
   <causalConnector id="onEndStop">
    <simpleCondition role="onEnd"/>
     <simpleAction role="stop" max="unbounded"/>
   </causalConnector>
   <causalConnector id="onSelectionStart">
     <simpleCondition role="onSelection"/>
     <simpleAction role="start" max="unbounded"/>
   </causalConnector>
   <constraintConnector id="parPres">
     <assessmentStatement role="begins" max="unbounded"/>
     <assessmentStatement role="ends" max="unbounded"/>
   </constraintConnector>
  </connectorBase>
 </head>
 <body>
  <port id="bp1" component="got"/>
  <media id="got" src="ts://1/10.11">
   <area id="anchor1" begin="94s" end="135s"/>
   <area id="anchor2" first="294tbv" last="315tbv"/>
  </media>
  <media id="recap" src="rtp://1.2.3.4/gameofthronesrecap"/>
  <media id="img1" src="recapIcon.jpg"/>
  <media id="img2" src="gameIcon.jpg"/>
  <context id="recapGame">
   <port id="rgp1" component="luagame"/>
   <media id="luaGame" src="recap.lua"/>
   <media id="htmlForm" src="quiz.html"/>
   <media id="gameBG" src="recapGameBackground.png"/>
   <link id="rgl1" xconnector="parPres">
     <bind role="begins" component="luaGame"/>
     <bind role="begins" component="htmlForm"/>
     <bind role="begins" component="gameBG"/>
     <bind role="ends" component="luaGame"/>
     <bind role="ends" component="htmlForm"/>
     <bind role="ends" component="gameBG"/>
   </link>
  </context>
  <link id="bl1" xconnector="onBeginStart">
    <bind role="onBegin" component="got" interface="anchor1"/>
    <bind role="start" component="img1"/>
  </link>
  <link id="bl2" xconnector="onEndStop">
    <bind role="onEnd" component="got" interface="anchor1"/>
    <bind role="stop" component="img1"/>
  </link>
  <!-- bl3 and bl4 links are similar to bl1 and bl2,
       but binding anchor2 and img2, instead of anchor1 and img1 -->
  <link id="bl5" xconnector="onSelectionStart">
    <bind role="onSelection" component="img1"/>
    <bind role="start" component="recap"/>
  </link>
  <!-- bl6 link is similar to bl5,
       but binding img2 and recapGame, instead of img1 and recap -->
 </body>
</ncl>
```

Listing 14.1 Specifying intermedia synchronism in an NCL document

(lines 54–57) specifies that if the user selects the "img2" icon, the "recap" media object shall be started. To do this, the link refers to the causal connector "onSelectionStart" (lines 12–15), which defines the condition role "onSelection" and the action role "start" with unbounded max cardinality. Link "bl6" (lines 58–61) was also omitted in Listing 14.1 and specifies a similar relationship, but binding the selection of "img2" (line 30) to the start of "recapGame" (lines 31–44). Note, in Listing 14.1, that "recapGame" is a context with three media objects (lines 33–35) and the "rgl1" constraint link (lines 36–43). The link refers to the "parPres" constraint connector (lines 16–20) to define that all those three media objects must begin their presentation at the same time and must end their presentation at the same time.

Returning to the anchors of "got" media object defined in Listing 14.1 example, note that the anchors in a streaming content are points of synchronization that can be defined by their initial and final time values according to (1) a clock relative to the object start time (see "anchor1"); or (2) in time-base values according to a time-base embedded in the media object content (see "anchor2"). In the example, <area id = "anchor2" first = "294tbv" last = "315tbv"> defines that the anchor "anchor2" begins when the time-base reaches the value "294" and ends when the time-base reaches "315". Multiple interleaved timebases can be present in the content. Different protocols and mechanisms can be used to define the identification and values of these time-bases [14], such as Presentation Time Stamps and Normal Play Time from MPEG [14], DVB Timeline [14], Timeline and External Media Information (TEMI) [14]. Independently of the low-level details, NCL defines [14] how to associate each time-base with its media content and how to manage time-base changes in the transmitted stream without impairing the authoring process.

A code span in embedded-code media objects is another type of content anchor definition specified using the <area> element. Embedded-code media objects are media objects with imperative or declarative code (the application media objects introduced in the first section of this chapter). Indeed, NCL has a strict separation between media content and document structure. NCL does not define any media itself. Instead, it defines the glue that relates media objects. In conformance with its conceptual model NCM, NCL does not restrict or prescribe the supported content types of its media objects. Which media objects are supported depends on the media players that are coupled in the NCL Player. The NCL Player specifies mechanisms and an API [15] so other players can be plugged-in while respecting all expected behaviors. In this sense, NCL can embed other NCL documents as media objects. In Listing 14.1 example, the "recapGame" context could be replaced by an NCL media object with the same specifications. Indeed, NCL provides a high level of reuse in the design of hypermedia applications. Soares Neto's work [16] discusses how the NCM and NCL design have succeeded in supporting reuse in a declarative approach.

14.3.3 Multi-Device Presentations

An NCL document can be presented in multiple devices at the same time. These devices can be registered in classes of two different types: those whose members are able to run media players (including imperative and declarative object players) and those whose members are not required to run media players. Members of the first class type, called active, shall be able to present media objects they receive from another device, called parent device. Therefore, devices in an active class run the same initial content but with individual and independent control by each of its members. On the other hand, devices in the second class type, called passive, must present the same content under a unique control.

There is no limit for the number of device classes in an NCL application. Furthermore, a device may be required to pertain to more than one class and to more than one class type. The NCL Player should offer a procedure to discover and easily register devices in classes. In a home network, for example, a media center (or a set-top box) can be responsible for this task. For simplicity, the NCL Player can provide some default classes. Specifically, Ginga middleware defines "systemScreen(1)" and "systemAudio(1)" as reserved classes of passive type. Ginga also defines "systemScreen(2)" and "systemAudio(2)" classes as reserved classes of active type. Moreover, when there is just one device in the device domain—the base device—no registering is necessary. Indeed, a base device belongs to an exclusive class, in which no other device can be registered to and is implicitly declared.

Using NCL, an author is able to specify which part of an application should be sent to each class of devices. The <regionBase> element can be associated with a device class where the presentation will take place. In order to identify the class (i) of devices, the device attribute can receive the "systemScreen(i)" or the "systemAudio(i)" value. When this attribute is not specified, the presentation must take place in the same device in which the NCL document is running.

In order to exemplify the behavior of a distributed presentation using multiple devices, suppose secondary devices could be supported in the NCL application provided in Listing 14.1. The <regionBase> elements required for this example are shown in Listing 14.2.

In Listing 14.2, three region bases, and thus three classes of devices, are defined. The first one (line 1) defines a presentation region for the primary device (the base device, implicitly declared). In this device, the video, for instance, an episode of a TV series from a broadcast channel, occupies the whole TV screen. The second class defines a presentation region (line 4) occupying the whole screen for secondary devices in the passive class 1. Finally, the last one (line 8) defines presentation regions for secondary devices in active class 2. The background presentation region occupies the whole secondary screen, while the game and the form are positioned according to the regions properties.

```
1.  <regionBase>
2.    <region id="mainScreen" width="100%" height="100%" zIndex="1"/>
3.  </regionBase>
4.  <regionBase device="systemScreen(1)">
5.   <region id="playIcon" width="100%" height="100%" zIndex="1"/>
6.    <region id="background" width="100%" height="100%" zIndex="1">
7.  </regionBase>
8.  <regionBase device="systemScreen(2)">
9.    <region id="gameIcon" width="100%" height="100%" zIndex="1"/>
10.   <region id="gameBackground" width="100%" height="100%" zIndex="1">
11.   <region id="game" left="5%" top="30%" width="40%" height="40%" zIndex="2"/>
12.   <region id="form" left="50%" top="5%" width="45%" height="90%" zIndex="2"/>
13.   </region>
14. </regionBase>
```

Listing 14.2 Specifying multi-device layouts in an NCL document

When a media object has to be presented on devices included in a passive class, only one media node instance must be created. This instance, created by the parent device, is then streamed to all child devices in the passive class. Therefore, if the icon to see a recap is selected in any of those devices, the recap presentation will appear in every device registered in "systemScreen(1)".

On the other hand, if an individual navigation on a form is desired, and this is normally the expected behavior, the active class solution must be assumed. When dealing with an active class, the required processing for playing media object contents is transferred from the parent device to the registered devices. If a viewer selects the game icon, he will play, individually, a thematic game of the TV series on his own device.

Although the Ginga standardized classes are the active and passive ones [1], it is possible to extend these classes specifying parameters and states that must be present in devices. These parameters include devices features such as the screen size, players available, and supported charsets. The Ginga module for multi-device support, called Ginga-MD [23], specifies "HardwarePlatform", "SoftwarePlatform", and "NetworkCharacteristics" entities to allow authors to specify the features desirable for devices in a class. These entities are based on the UAProf [17] model, an implementation of W3C Composite Capabilities/Preference Profiles (CC/PP) framework [24].

In Ginga-MD platform, classes are specified using Resource Description Framework (RDF) files containing, as aforementioned, the desirable device features. These files can also contain states specifications, including the maximum or minimum number of devices allowed in a class. When the minimum number is specified, the start of a presentation depends on the number of devices available, which is limited, if specified, by the maximum number. It is also possible to specify applications whose media content is produced on-the-fly by devices. In these applications, authors must specify the <media> element with the "src" attribute containing the device path, which is composed by the device class name and the device number in the class [9].

Regardless of class types or features provided by the NCL Player, in the NCL hierarchical control model devices can only present media objects coming from a single parent device (explicitly or rendered in video frame buffers or audio sample sequences). A class may not have more than one parent device in a given moment. Moreover, a base device may not receive media objects from other devices of the same device domain. In other words, it is not possible to have a device as an ascendant or a descendant of itself.

In the beginning of an NCL application presentation, all input events associated with devices of the application domain are under control of the domain's base device.

When a <media> element presented on a device of an active class receives the presentation focus and is selected, the device in charge of the presentation gains control of all its input events and all input events of devices that are in classes that will be its descendants (classes for which it will be the parent device). The media player can then follow its own navigational rules. When the "BACK" key is pressed, the control of all previously mentioned input events is returned to the parent device. As usual in NCL, the focus will go to the element identified by the *service.currentFocus* attribute of the settings node (<media type = "application/x-ncl-settings) [18].

It must be noted that there may be more than one device controlling the key navigation, but each one controlling different input events from the others.

14.4 Intermedia Synchronization Management

The synchronization among media objects in an application can be defined associating specific time moments for each media event, in an approach usually called timeline. Although this approach can make the presentation and playout stages implementation easier, it is only appropriate when applications have only predictable events, that is, when the moments in time for the events occurrence can be calculated before the application runtime. In applications where unpredictable events are common, like viewer interactions, and when content and content-presentation adaptations are carried out during runtime, another approach is necessary, and this is just the case of NCL applications, where an event-driven approach was chosen.

Using specifications like those provided by NCL, the author does not need to know the exact moments in time when the events will occur. However, during the application presentation stage, it is necessary to calculate the moments in time of events specified, maintaining the presentation control of the application. Besides this, in the case of distributed applications, it is also necessary to think about how to manage transmissions from servers to receivers in order to preserve the presentation quality.

Since authoring goals are very different from presentation goals, other intermediate structures can be proposed in order to guide all the synchronization control

process. The Hypermedia Temporal Graph (HTG), detailed in the next subsection, is a labeled digraph able to maintain all relationships among actions, predictable or unpredictable, and, at the same time, improve upon the presentation flow control.

14.4.1 Hypermedia Temporal Graph (HTG)

HTG was designed to be a directed time graph model able to represent relationships among events in an application. Each vertex in a graph represents an event state transition. HTG recognizes the same event types defined in NCL: presentation event, corresponding to playing a content anchor (whole mediaobject content, or part of this content); selection event, corresponding to a viewer interaction (selection of a content anchor); and attribution event, corresponding to setting a value to a mediaobject's property (variable). Each event defines a state machine that should be maintained by the receiver user agent, as shown in Fig. 14.7. An event can be in the sleeping, occurring, or paused state, and change its state upon receiving actions: start, stop, pause, resume, natural end, and abort.

The edges between vertices represent relationships between transitions. Edges are labeled by conditions that must be satisfied in order to trigger the edge traversing. More precisely, in HTG, graphs are defined by (V, A, C) triples, where

$V = (v_0, v_1, v_2, v_3, \ldots, v_{n-1})$ is a finite set of vertices, where each vertex represents a transition on an event state machine. Each vertex is represented by a triple: the action name that triggers the transition; the corresponding event type; and the corresponding anchor unique identifier (if presentation or selection event) or property unique identifier (if attribution event);

$A = (a_0, a_1, a_2, a_3, \ldots, a_{m-1})$ is a finite set of edges that individually represent a relationship among transitions. An edge "a" is a pair $(v, w) \in V \times V$, where v and w are named source and target vertices, respectively;

$C = \{c_{ij}\}$ is a finite set of conditions associated with edges. A c_{ij} condition is associated with the edge $(v_i, v_j) \in A$ and must be satisfied in order to trigger the traverse to the target vertex (trigger the transition) of that edge.

Conditions can be simple or compound. A simple condition is defined by

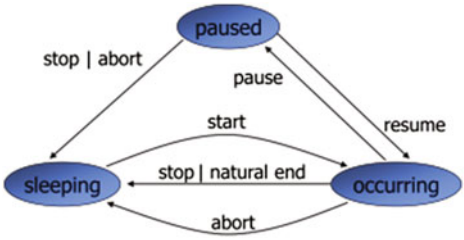


Fig. 14.7 Event state machine

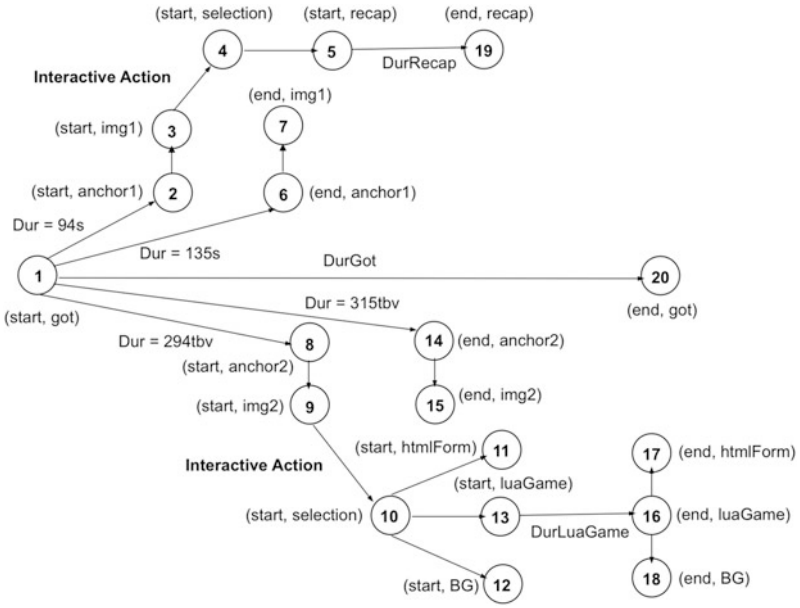


Fig. 14.8 Temporal graph for NCL application of Listing 14.1

- a temporal interval that must be spent in order to fire the edge transition;
- a variable that must be evaluated in relation to a desired value; or
- external actions, such as user interactions.

Note that the two last simple condition types can define unpredictable event. A variable may be evaluated in runtime, and a user interaction is always unpredictable within a certain time interval.

Compound conditions are defined through logical operators (OR, AND, NOT) binding two or more conditions (simple or compound).

The HTG for NCL application of Listing 14.1 is presented in Fig. 14.8. The graph for this application is made up of vertices representing the beginning and the end of the media object "got" presentation, a TV series from a broadcast channel, and its temporal anchors, and also of vertices representing the beginning and the end of the images (icons) presentation.

To represent the viewer interactions, interactive actions conditions are used in the graph. The selection event over "img1" starts the presentation of the video "recap", an on-demand IPTV service. The form, the Lua game, and also the background image are presented when the selection event occurs over "img2". Since the selection event is unpredictable, the events triggered due to the selection event will have their moments in time calculated in a relative way.

The Lua game has an unexpected duration; therefore, when this game ends, the presentation of the form and the image is stopped. This specification was chosen to represent the constraint relationship among them. In Fig. 14.8, the vertex

representing the end of the game triggers vertices representing the form and the image presentation end.

14.4.2 Scheduling Plans

After defining an application starting point, HTG can be used to derive other structures related to the synchronization management, including the player-load plan that specifies moments in time for player instantiations, prefetching plan that specifies moments for requests object retrieval from content servers, and the presentation plans for starting each media presentation [19].

When applications contain only predictable events, graph edges are labeled only by temporal intervals. From the document starting time, the graph traverse identifies every action that must be applied to media players. These actions can have their occurrence in time computed taking into account the time intervals required to satisfy conditions from the HTG entry point to the corresponding action vertices. This set of actions and their corresponding occurrences in time compose the presentation plan.

For each start presentation and resume presentation action in the presentation plan, new occurrences in time must be computed for the equivalent player-load plan, taking into account the delay for each player instantiation. Other types of actions present in the presentation plan must be disregarded when building the player-load plan. At the same way, the prefetching plan is built based on the presentation plan, as usual, taking into account the estimate network transfer delay and transfer jitter for each presentation of object in the presentation plan.

While the presentation plan for applications without unpredictable events can be entirely computed a priori (at compile time), for applications where unpredictable events may occur this is not true. It is the case of the NCL document of Listing 14.1, which has two selection events, corresponding to viewer interactions.

However, the same procedure previously applied can be used to compute actions and their corresponding occurrences in time for all predictable events from an application starting point to an unpredictable event; and from each unpredictable event to the next unpredictable event in the graph traverse. In this last case, the computed occurrences in time will be relative to the starting time of the unpredictable event in the traversed path.

During an application presentation, as soon as an unpredictable event time is known, the presentation plan is updated changing all moments in time relative to this event to be now relative to document starting time, that is, by adding the moment in time that the unpredictable event has occurred to moments in time relative to this unpredictable event.

Sometimes an unpredictable event does not happen within the time period at whatever time its occurrence is allowed. In this case, all actions in the presentation plan whose occurrence depends only on this event must be removed. Suppose, for instance, that in the aforementioned NCL application, the second icon has not been

selected. In this case, the time moments for presentation of the game, form, and background can be discarded.

When unpredictable events may occur, the player-load plan and prefetching plan can be updated in a similar way to presentation plan. However, they can also be built using a conservative algorithm assuming that all unpredictable events happen immediately after they are enabled. This strategy for prefetching plan may include the worst-case transfer delay and jitter.

Sometimes it may be useful to build some scheduling plans also at the servers. In terrestrial DTV systems, for instance, it is usual to have some application data asynchronously transmitted. In this case, data must be delivered before its presentation time, repeatedly, in order to guarantee its reception no matter the time a channel is tuned in.

In the aforementioned scenario, a pushed-data plan, also called carousel plan due to the cyclical data structure usually used to transport these data, may improve the transmission management. The carousel plan is similar to the presentation plan built in receivers, with the exception that, once built, it does not need to be updated, since there is no feedback coming from receivers. Therefore, as for the server knowledge, all unpredictable events must be treated as if they will happen at the moment they will be enabled.

14.5 Ginga

Ginga is the NCL Player standardized internationally for Internet Protocol Television (IPTV) [1], Integrated Broadcast–Broadband (IBB) systems [9], and Terrestrial DTV services [20].

Ginga architecture is depicted in Fig. 14.9. To be Ginga compliant, the Ginga Common Core (Ginga-CC, see (1) in Fig. 14.9), Ginga-NCL (2), and the Private Base Manager (3) subsystems are required. Additionally, Ginga allows for optional extensions, which include execution engines based on other programming languages.

The core of Ginga-NCL (2) subsystem is the NCL Player (12). This component is tasked with receiving and controlling multimedia applications authored in NCL. Applications are delivered to the NCL Player (12) by the Ginga Common Core (1).

The NCL Player (12) deals with NCL applications collected inside a data structure known as private base. The Private Base Manager (3) component is tasked with receiving NCL editing commands and control commands and maintaining the life cycle of NCL applications being presented. These commands may be issued by the broadcaster, by the user (via the AppCatUI component (4)), or by the applications themselves (via the Live Editing API [1]).

Ginga-NCL Presentation Engine (2) supports multi-device presentations through its Layout Manager module. This component is responsible for mapping all presentation regions defined in an NCL application to canvas on receiver device's displays.

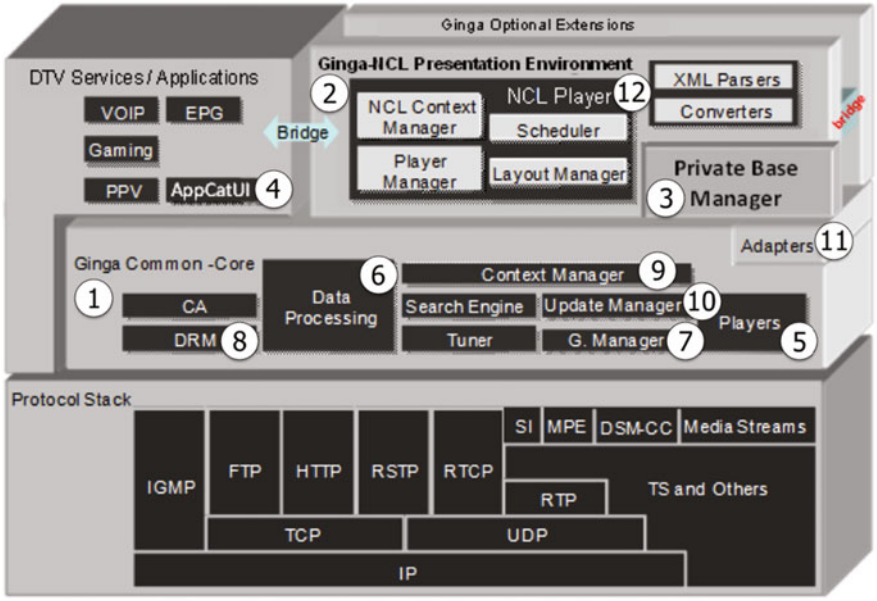


Fig. 14.9 Ginga architecture overview

The Ginga Common Core (Ginga-CC) (1) is composed of media players (5), procedures to obtain contents that can be transported in diverse (broadcast and broadband) networks accessed by a receiver device (6), and the conceptual display graphical model defined by the receiver device platform (7). The Ginga Common Core (1) is also tasked with gathering metadata information (6) and providing this information to NCL applications; for providing an API to communicate with Digital Rights Management (DRM) system (8); for managing context information (like user profiles and receiver device profiles) (9); and for supporting software version management (update) of Ginga's components (10).

Media player components (5) serve application needs for decoding and presenting content types, including perceptual media content and content that contains declarative or imperative code, like HTML code, Lua code. A generic media player API establishes the necessary communication between media player components and the Ginga-NCL subsystem (2). Thanks to this API, Ginga-NCL (2) and the Ginga-CC (1) are strongly coupled but independent subsystems. Ginga-CC (1) may be substituted by other third-party implementations, allowing Ginga-NCL to be integrated into other middleware specifications, extending their functionalities with NCL facilities. Players (5) that do not follow the generic API are required to use the services provided by Adapters (11).

14.5.1 Ginga IBB

Integrated Broadcast–Broadband (IBB) [21] service is a kind of converged service where a receiver device (TV, set-top box, mobile) is able to connect to different delivery networks. But only recently the TV receiver devices became able to do this. Interactive Digital TV services already existed before IBB initiatives. Some established, standardized technologies like Ginga have supported the creation of multi-sourced multimedia content since their conception. For this reason, Ginga promptly preformed a key role to contribute to the definition of IBB requirements, architecture, systems, and services.

Since its first standardized version in 2007, Ginga-NCL provides support to converged services by making use of broadcast and broadband sourced media objects (like those media objects in Listing 14.1, lines 24–35). NCL's characteristics make it a comprehensive, declarative solution for IBB services: the language flexibility; its reuse facility; multi-device support; presentation and application content adaptability; API for building and modifying applications on-the-fly; and, mainly, its intrinsic ability for easily defining spatiotemporal synchronization among media assets (including viewer interactions).

Ginga specifications have been updated in order to promote better support for IBB requirements. As defined in [21], an IBB system must incorporate an application control framework that coordinates the coexistence of DTV services, IBB services and its various types and sources of applications. In Ginga, this framework is now implemented in its core components, namely the NCL Player and the Private Base Manager modules.

The NCL Player is tasked with receiving an NCL application and controlling its presentation, trying to guarantee the specified relationships among media objects. The NCL Player deals with applications that are collected inside a data structure known as private base. Applications in a private base may be edited, started, paused, resumed, aborted, stopped, saved and may refer to each other.

Ginga associates at least one private base with each TV channel (set of services)—the TV channel's default private base. When a certain TV channel is tuned, its corresponding default private base is opened and activated by the Private Base Manager. Resident applications are also supported and managed in a specific private base, as well as preinstalled applications.

Ginga allows IBB service providers to control the execution, availability, and visibility of their service-associated applications, using signaling mechanisms available through the Application Information Table (AIT) [9] and through stream events, e.g., NCL editing commands. IBB service providers cannot control applications which are not signaled in their service and manually started by the user.

A Ginga implementation must grant the isolation between the running applications within its scope, but the IBB receiver device must provide isolation from other platform applications. This is important for stand-alone applications that may not be aware of other native applications executed in the system.

Ginga IBB defines a formal structure called Private Base Data Structure (PBDS), in which applications can traverse and reflectively get information about themselves and about other available applications. Broadcasters, IBB applications, and Ginga extensions can manipulate the PBDS to manage applications' life cycle, change their behavior on-the-fly as well as set up their persistency needs.

A well-defined control API is used to pass commands to the Private Base Manager, from the broadcaster via AIT signaling and stream events and from IBB applications, via the Live Editing API.

Application Catalogue User Interface (AppCatUI) is an extension of the Ginga middleware that must be provided by the IBB receiver device, intended for listing the applications found in the available private bases. The available applications may be launched or managed by the end user, by adding, moving, or removing them. The list identifies if the application is persistent or nonpersistent. It also identifies the applications for which the minimum resources required for presentation are already preloaded.

14.6 Conclusions

This chapter presented the Nested Context Language (NCL), its relying model Nested Context Model (NCM), their main features, and the Ginga Presentation Engine.

NCL is a domain-specific language that defines the glue that holds components together in multimedia presentations. By focusing on the specification of hypermedia application structure, NCL brings together outstanding features. NCL is a flexible, highly reusable language, that supports multi-device presentations, content adaptation, multi-sourced content, on-the-fly application editing, and, mainly, it provides an intrinsic ability for homogeneously defining spatiotemporal synchronization among media assets, including viewer interactions.

Its support for multi-device presentation, where devices are organized hierarchically and easily identified by device classes instead of network addresses, takes advantage of the inherent spatiotemporal synchronization of the NCL to allow developers to easily create rich, multi-screen, distributed multimedia applications.

The internationally standardized NCL Player for DTV/IPTV/IBB services, called Ginga, allows the broadcasters, applications, and Ginga extensions to manage applications' life cycle, change their behavior on-the-fly as well as set up their persistency needs. Ginga specifies a well-defined control API used to pass commands to the receiver, from the broadcaster via stream signaling and from applications, via a Live Editing API.

NCL is indeed under constant evolution by addressing new requirements from emerging scenarios, where spatiotemporal, multi-device synchronization is needed. These scenarios under study not only promote the integration of different delivery networks, but also go beyond TV and Web, and even include unconventional media types that only flexible languages like NCL will be capable to support.

As with most of multimedia documents available today, NCL documents do not explore data semantics. Its specification offers little to ease authoring of meaningful content. To tackle this issue, there is an effort [22] at integrating support for enriched concept description to NCL. These extensions enable the specification of relationships between concept descriptions and multimedia content in the hypermedia way, composing what Moreno et al. [22] call hyperknowledge.

References

1. ITU-T. Recommendation ITU-T H.761: Nested Context Language (NCL) and Ginga-NCL (2014)
2. Bulterman, D.C.A., Rutledge, L.W.: SMIL 3.0: Flexible Multimedia for Web, Mobile Devices and Daisy Talking Books, 2nd edn. Springer Publishing Company, Incorporated (2008). ISBN: 3540785469 9783540785460
3. W3C World-Wide Web Consortium: Synchronized Multimedia Integration Language—SMIL 3.0 Specification. W3C Recommendation (2008)
4. ISO/IEC International Organization for Standardization 14496-1: Coding of Audio-Visual Objects—Part 1: Systems, 3rd edn (2004)
5. ISO/IEC International Organization for Standardization 14496-20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF) (2006)
6. Cesar, P., Bulterman, D.C.A., Obrenovic, Z., Ducret, J., Cruz-Lara, S.: An architecture for non-intrusive user interfaces for interactive digital television experiences. In: Proceedings of European Interactive TV Conference (2007)
7. W3C World-Wide Web Consortium: Scalable vector graphics—SVG 1.1 Specification. W3C Recommendation (2011)
8. Concolato, C., Le Feuvre, J., Moissinac, J.C.: Timed-fragmentation of SVG documents to control the playback memory usage. In: Proceedings of ACM Symposium on Document Engineering, DocEng, New York, USA, (2007)
9. ITU-R. Recommendation ITU-R BT.2075-1: Integrated Broadcast-Broadband System (2017)
10. ETSI. ETSI TS 102 796 V1.2.1 (2012-11): Hybrid Broadcast Broadband TV
11. ARIB. ARIB STD-B62: Multimedia Coding Specification for Digital Broadcasting (Second Generation) (2014)
12. Soares, L.F.G., Rodrigues, R.: Nested context model 3.0. Part 1—NCM Core. Tech. Rep. DI. ISSN 0103-9741 (2005)
13. Pérez-Luque, M.J., Little, T.D.C.: A temporal reference framework for multimedia synchronization. IEEE J. Sel. Areas Commun. 36–51 (1996)
14. Moreno, M.F., Costa, R.M.R., Soares, L.F.G.: Interleaved time bases in hypermedia synchronization. IEEE Multimedia Mag. **22**(4), 68–78 (2015). https://doi.org/10.1109/MMUL.2015.74
15. Soares, L.F.G., Moreno, M.F., Marinho, R.S.: Ginga-NCL architecture for plug-ins. Softw. Pract. Exp. **43**, 449–463 (2013). https://doi.org/10.1002/spe.2144
16. Soares Neto, C.S., Soares, L.F.G., Souza, C.S.: The nested context language reuse features. J. Braz. Comput. Soc. **16**(4), 229–245 (2010)
17. WAP Forum—Open Mobile Alliance—WAG UAProf. In: Technical Report WAP-248-UAPROF-20011020-a
18. Soares, L.F.G., Costa, R.M.R., Moreno, M.F., Moreno, M.F.: Multiple exhibition devices in DTV systems. In: Proceedings of the Seventeen ACM International Conference on Multimedia (2009). https://doi.org/10.1145/1631272.1631312

19. Costa, R.M.R., Soares, L.F.G., Moreno, M.F.: Intermedia synchronization management in DTV systems. In: Proceedings of the Eighth ACM Symposium on Document Engineering (2008). https://doi.org/10.1145/1410140.1410203
20. ITU-R. Recommendation ITU-R BT.1699-2: Harmonization of Declarative Application Formats for Interactive TV (2017)
21. ITU-T. Recommendation ITU-T J.205: Requirements for an application control framework using integrated broadcast and broadband digital television (2014)
22. Moreno, M.F., Brandao, R., Cerqueira, R.: Extending hypermedia conceptual models to support hyperknowledge specifications. Int. J. Semant. Comput. **11**, 43 (2017). https://doi.org/10.1142/S1793351X17400037
23. Batista, C.E.C.F.: Ginga-MD.: A NCL based platform for supporting the execution of multi-device hypermedia applications. Ph.D. thesis, Pontifical University of Rio de Janeiro (2013)
24. Klyne, G., et al.: Composite capability/preference profiles (CC/PP): structure and vocabularies. W3C Work. Draft (2004)

Chapter 15
Time and Timing Within MPEG Standards

Lourdes Beloqui Yuste

Abstract This chapter focuses on the time system used by the decoder at the end user side to replicate the encoder's clock system to accomplish a synchronized media play-out at end user side. The time system usually uses tools such as clock references and timestamps coded within the media stream. This chapter does not go into detail of the protocols used in IP networks to perform the media delivery, but it explains in detail the time-related fields coded within the media streams which are used at the user side decoder to provide a synchronized media play-out. The principal Moving Picture Experts Group (MPEG) media standards and their time system are described in this chapter. That includes MPEG-2 Transport Streams (MP2T), MPEG-4, MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH), and MPEG Media Transport (MMT). This chapter also details the time information used by Digital Video Broadcasting (DVB) which is broadly used in multiple media delivery systems and used hand in hand with MPEG standards. First of all, this chapter describes the synchronization between video–audio media streams (lip sync) within a program in MP2T, and secondly, the synchronization between multiple programs, delivered within a multiplexed single MP2T stream. Secondly, this chapter describes timing issues in MPEG-4, which utilizes a different timeline system from MP2T to implement clock references and timestamps, as it is an object-oriented multimedia standard. Thirdly, this chapter describes timing issues in MPEG-DASH, which is an Adaptive Streaming over HTTP protocol widely used over the Internet. Additionally, this chapter also describes the time transmission in DVB systems, which use MP2T as a media container. The tools used are the DVB Service Information (DVB SI) and MPEG-2 Program-Specific Information (MPEG-2 PSI) tables. Finally, this chapter introduces the latest MPEG standard for media delivery, MMT, which aims to be a unique media delivery standard for heterogeneous networks, broadband technologies used in Internet TV and IPTV (Internet TV refers in this chapter to media delivery over a public non-managed IP network, such as Internet, and IPTV refers to media delivery over

L. B. Yuste (✉)
Ericsson, Athlone, Ireland
e-mail: lbeloqui@gmail.com

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_15

a private, managed IP network media delivery). It has also been proposed for broadcast (DVB) media delivery.

Keywords Media delivery systems · Media delivery technologies
Time system · Media systems

15.1 Introduction: Relevance of Time and Timing in Media Delivery Technologies

Presently, MPEG-2 Transport Streams (MP2T) are used as a contained format for media delivery in broadcast systems, such as Digital Video Broadcasting (DVB) via Terrestrial, Satellite, or Cable technology. This chapter is concerned with the tools used by DVB, MP2T, MPEG-4, and MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) to provide a good Quality of Experience (QoE) for end users with the synchronized play-out of media. It also introduces the latest MPEG Media Transport (MMT) standard, which intends to be the media delivery standard for heterogeneous networks.

Timelines are one of the key tools used for media synchronization over broadband and broadcast media delivery as it is the main synchronization tool for MPEG-2 and MPEG-4 standards.

For the correct media play-out at receiver side, the receiver needs to replicate the source time system, and time and timing information are the tools to facilitate this task. Time indicates the moment in time when units of the media stream should be displayed and timing adds a timeline at the media stream.

The decoder at receiver side gathers the time information, and decodes and displays the media stream in a synchronized manner. In the following sections, the tools to provide time and timing in a media stream are fully described.

The sections of this chapter are organized as follows: The first section presents an introduction of the importance of time and timing in media delivery technologies, and the second section deals with the relevance of MPEG-2 Transport Streams in media delivery. Section 15.3, Sect. 15.4, and Sect. 15.5 present time and timing in MPEG-2, MPEG-4, and MPEG-4, respectively. Finally, in Sect. 15.6, time transmission in DVB systems is explained and in Sect. 15.7, time transmission in MMT is described. The conclusion is found in Sect. 15.8.

15.2 Importance of MPEG-2 Transport Streams in Media Delivery Systems

At this moment in time, there are two groups of media delivery systems: broadband and broadcast media delivery. However, Moving Picture Experts Group (MPEG) standards are used in both. The latest broadcast media delivery uses Digital Video

Broadcasting (DVB) standards, whereas broadband delivery uses multiple media standards and protocols when media is transmitted via IP networks (Internet). The relevance of MPEG-2 Transport Streams (MP2T) relies on the fact that it is the standard used by MPEG-DASH and DVB systems for media delivery and it is also used in broadband media delivery, for example, when DVB Internet Protocol TV (DVB-IPTV) is deployed.

Broadcast media delivery uses DVB Terrestrial (DVB-T/T2), Satellite (DVB-S/S2), Cable (DVB-C/C2), or the latest DVB Handheld/Mobile (DVB-H) standards. The DVB broadcast system delivers media (multiple programs) to all users, allowing them to select the TV channel to watch. The DVB broadcast system uses MP2T as a media container to deliver the media content to end users.

Broadband media delivery is classified into two main systems, IPTV and Internet TV, although Internet TV usage is gaining popularity. IPTV systems use private, managed IP networks, providing media delivery mainly via multicast protocols. Media is replicated in the network, whereas Web TV uses public, unmanaged IP networks via unicast delivery (every end user is streamed one program). The main advantage of Web TV over IPTV relies on the fact that the former is available worldwide while the latter is locally restricted to the private network. Moreover, Internet TV provides multiple media choices, e.g., Internet Radios from the country of your choice. Thus, the user is not being limited to the IPTV media provider selection.

As explained in Sect. 15.1, MP2T synchronization tools and/or techniques apply to single programs, even though an MP2T stream may convey multiple multiplexed programs. As such, there is a need to consider different synchronization tools and techniques to synchronize multiple programs. Programs can be conveyed in one or multiple MP2T streams and can be delivered via different media delivery platforms, such as broadcast TV (DVB) and broadband (Web TV and IPTV) platforms.

In summary, although multiple media standards can be used, MP2T is the principal media container used by DVB standards, including Cable, Terrestrial, Satellite, and IPTV. A synchronized media play-out is a key requirement for a satisfactory Quality of Experience (QoE) demanded by end users. This chapter explains how MP2T streams accomplish intra- and inter-media synchronization via MP2T timelines inserted by the encoder and used by the decoder for a synchronized media play-out.

In the next section, it described the time and timing within MP2T streams via timestamps and clock references are described. Moreover, it explains how the decoder, at client side, uses the information to perform a synchronized media play-out.

15.3 Time and Timing Within MP2T

The synchronized media play-out of multiple media streams (video, audio, subtitles, etc.) is a key part of the appreciation of good QoE from the user side. MP2T streams accomplish this via inserted timelines and timestamps (time and timing) within the MP2T packets.

Timelines refer to the relative time within the stream and the timestamps signal when the different media units should be decoded/displayed in relation to the time in the timelines.

Timing is the frequency at which the media clock works, and *time* is a concrete moment in time to which multiple media streams can refer. In other words, timing is the clock reference and time is the timestamp; both are needed to accomplish media synchronization of the MP2T media content play-out.

To synchronize multiple media streams, two types of synchronization are needed: intra- and inter-media synchronization. Intra-media sync refers to the timing, the synchronization of the clock frequency within a media stream. On the other hand, inter-media synchronization refers to the synchronized play-out of multiple media streams using timestamps, e.g., the audio, video, and subtitles.

To sum up, the clock references are the tools to replicate the media encoder clock at the end user decoder, implementing intra-media sync, while the timestamps are the tools used to synchronize the combined play-out of multiple media streams, implementing inter-media sync. In the next two subsections, the MP2T clock references and timestamps are described in detail. Timelines and timestamps interrelate to accomplish media synchronization, and as a result, both intra- and inter-media sync should be achieved. This relationship is explained in this section.

In this section, the timelines within the MP2T streams, clock references, and timestamps are described. It is explained how the Phase-Locked Loop (PLL) performs synchronization at MP2T stream decoder side. And lastly, as an added feature, System Clock Descriptor (SCD) is described.

15.3.1 MP2T Streams

The main data streams carried by MP2T streams contain media. These streams are made up of the smallest media data units, called access units (AUs) or media data units (MDUs), which are the smallest timed media data unit. Their size depends on the type of media; e.g., a video MDU (i.e., a video frame) is generally bigger than an audio MDU (i.e., an audio frame/sample). While the AUs/MDUs are encoded media data, the presentation unit (PU) contains decoded media units ready to be presented by the media decoder.

A combination of MDUs from the same media type creates an Elementary Stream (ES) which is then packetized (adding information via a packet header) to become a Packetized Elementary Stream (PES), which is the media content of the

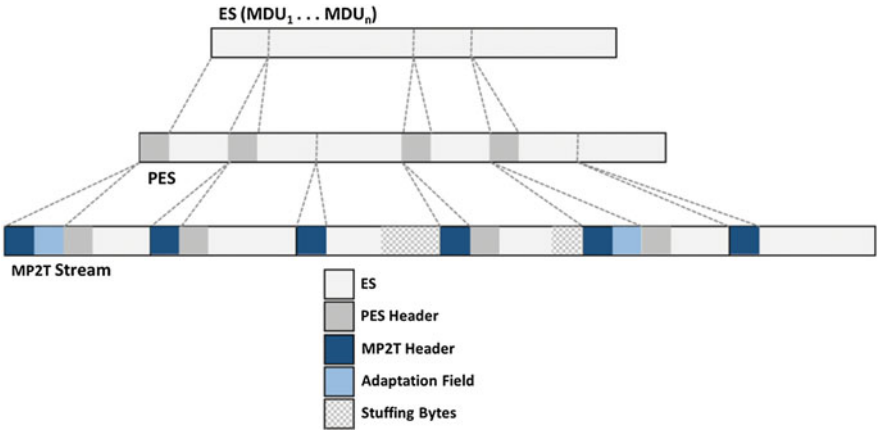


Fig. 15.1 ES, PES, and MP2T streams (including examples of Adaptation Fields and stuffing bytes within the MP2T payload)

MP2T payload. MP2T packets have a fixed size of 188 bytes (stuffing bytes are used when it is required to accomplish this condition). In Fig. 15.1, the packetizing structure of an MP2T is described, although it does not include another type of MP2T payload which contains information tables (see Sect. 15.6) or descriptors (see Sect. 15.3.5).

An MP2T packet has a 4-byte header and an optional variable size Adaptation Field. The Adaptation Field is where the timing information will be conveyed. Therefore, the MDUs are linked to the MP2T stream timelines. In Fig. 15.2, the MP2T header and Adaptation Field, with the clock references, are shown within an MP2T packet.

The clock references are found in the Adaptation Field, whereas the timestamps are part of the PES Header. The Adaptation Field is optional; e.g., it is normally present when the beginning of a PES is found at the MP2T payload. The clock references in the MP2T stream are always found in the same program. In other

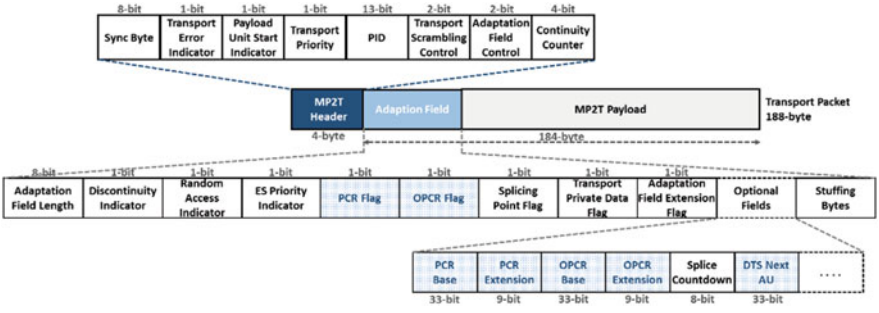


Fig. 15.2 Clock References in MPEG-2 Transport Stream (MP2T)

words, MP2T packets with the same PID. This PID value is included in the information tables (see Sect. 15.5).

Moreover, the MP2T packets convey descriptors and information tables that also carry timing information (excluded from Fig. 15.1 for simplicity). Information tables are further described in Sect. 15.6, and descriptors are used for the solution described in Sect. 15.3.5.

15.3.2 Clock References

To accomplish intra-media sync, it is important that both the encoder's and the decoder's clocks are synchronized to the same clock frequency. MP2T uses clock references for this purpose.

There are two types of clocks: global and relative clocks. Global clock uses global time references such as Coordinated Universal Time (UTC), whereas a relative clock, used in MP2T streams, uses the MP2T timelines as its own clock system and frequency.

The main clock references are encoded into two fields within the Adaptation Field: the 33-bit field *Program_clock_reference_base* (PCR_{base}) and the 9-bit field *Program_clock_reference_extension* (PCR_{ext}) (See Fig. 15.2). Those two combined fields make the PCR value, which conveys the 27 MHz value of the encoder's system clock. The terms related to MPEG-2 timelines are listed and defined in Table 15.1.

The system clock frequency in MP2T streams (SCF_{MP2T}) in encoder's and decoders' clock is 27 MHz and falls within the following constraints [1]:

$$27\,\text{MHz} - 810\,\text{Hz} \leq \text{SCF}_{\text{MP2T}} \leq 27\,\text{MHz} + 810\,\text{Hz}$$

While trying to synchronize to the same clock frequency, the $SCF_{MPEG\text{-}2}$ change rate should not be greater than [1]:

$$\text{SCF}_{\text{MP2T}}\,\text{Change Rate} \leq 75 \times 10^{-3}\,\text{Hz/s}$$

Another important value related to clock references is the Transport Rate (TR), whose value depends on the $SCF_{MPEG\text{-}2}$ and the PCR values [1]:

$$TR(i) = \frac{\left(\left(i^{'} - i^{''}\right) \cdot SCF_{MP2T}\right)}{PCR(i^{'}) - PCR(i^{''})}$$

The following equation is used to obtain the PCR value from the two PCR fields, PCR_{base} and PCR_{ext}, [1]:

Table 15.1 Terms for MPEG-2 timelines related to MP2T. Table from [1] and definitions from [2]

Term	Meaning
CA_{freq}	Clock accuracy frequency
CA_{ext}	'*Together with the CA_int, it gives the fractional frequency accuracy of the system clock in parts per million*'
CA_{int}	'*Together with the CA_exp, it gives the fractional frequency accuracy of the system clock in parts per million*'
DTS(j)	The decoding time of an AU, $td_n(j)$, *being* j the AU index within the ES
i	'*Index of any byte in the Transport Stream for* $i'' < i < i'$'
i'	'*Index of the byte containing the last bit of the immediate following* PCR_{base} *field applicable to the program being decoded*'
i''	'*Index of the byte containing the last bit of the most recent* PCR_{base} *field applicable to the program being decoded*'
j	AU or MDU index within the ES
k	PU index within the ES
n	'*Index to the ESs*'
PCR(i)	'*It indicates the time t(i), where i is the index of the byte containing the last bit of the* PCR_{base} *field*'
PCR_{base}	PCR base field in units of the system clock frequency
PCR_{ext}	PCR extension field in units of the system clock frequency
PTS(k)	'*It indicates the time of presentation,* $tp_n(k)$*, in the STD of a PU k of ES n*'
SCF_{MP2T}	'*System Clock Frequency of a MPEG-2 program*'
$td_n(j)$	'*Decoding time of an AU* $A_n(j)$'
$tp_n(k)$	'*Presentation time of PU* $P_n(k)$'
TR(i)	'*Number of bytes in the Transport Stream between the bytes containing the last bit of two successive* PCR_{base} *fields of the same program divided by the difference between the time values encoded in these name two PCR fields*'

$$\mathrm{PCR}(i) = \mathrm{PCR}_{\mathrm{base}}(i) \cdot 300 + \mathrm{PCR}_{\mathrm{ext}}(i)$$

PCR is a numeric value which with the TR and the SCF_{MP2T} gives '*the time t(i) at which the* i^{th} *byte enters the T-STD*' [1], being T-STD the Transport System Target Decoder.

$$\mathrm{t}(i) = \frac{\mathrm{PCR}\left(i''\right)}{\mathrm{SCF}_{\mathrm{MP2T}}} + \frac{i - i''}{\mathrm{TR}(i)}$$

The values of PCR_{base} and PCR_{ext} can be obtained from the $SCF_{MPEG\text{-}2}$ using the following equations (being integer divisions):

$$PCR_{base}(i) = \frac{SCF_{MP2T} \cdot t(i)}{300} \%2^{33}$$

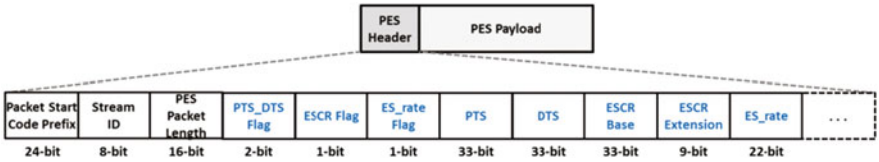


Fig. 15.3 PES Header and timestamps fields

$$PCR_{ext}(i) = \frac{SCF_{MPEG-2} \cdot t(i)}{1} \%300$$

The coding frequency of PCR values for each program should never be greater than 0.1 s; therefore [1]:

$$|t(i) - t(i')| \leq 0.1 \text{ s}$$

'for all i and i' where i and i' are the indexes of the bytes containing the last bit of consecutive PCR_{base} fields in the MP2T packets' [1].

There are another two types of clock references that are similar to, but not exactly the same as, PCRs. These are the Elementary Stream Clock Reference (ESCR) and the Original Program Clock Reference (OPCR).

The ESCR values, conveyed within the PES Header, follow the same pattern as the PCR clock references. ESCR is conveyed via two fields the $ESCR_{base}$ (33-bit) and $ESCR_{ext}$ (9-bit). ESCR frequency is equal to SCF_{MPEG-2}, 27 MHz (See Fig. 15.3). ESCR values are used to convey clock references within the PES packets without being packetized into the MP2T streams. The OPCR values, conveyed within the Adaptation Field, also follow the same pattern as the PCR and ESCR clock references. OPCR is conveyed via two fields: the $OPCR_{base}$ (33-bit) and $OPCR_{ext}$ (9-bit) (See Fig. 15.2). OPCR frequency is equal to SCF_{MPEG-2}, 27 MHz. It is only conveyed in the Adaptation Fields where PCR values are found, and it is used to map the MP2T PCR values to the original single program clock references, which are conveyed in the MP2T stream.

Clock reference is the tool used to replicate the encoder's clock at the end users' decoder. The Phased-Locked Loop (PLL), within the decoder, is used to keep the encoder's and decoder's frequencies synchronized, in this case to 27 MHz, which is the value of SCF_{MPEG-2}.

15.3.2.1 Phase-Locked Loop (PLL)

The Phase-Locked Loop (PLL) is responsible for keeping the decoder's clock, at end user side, at the same frequency as the encoder's at the media source. In the particular case of MP2T streams, the SCF_{MPEG-2} is set at 27 MHz and therefore, the PLL should keep the decoder's clock running at this same frequency.

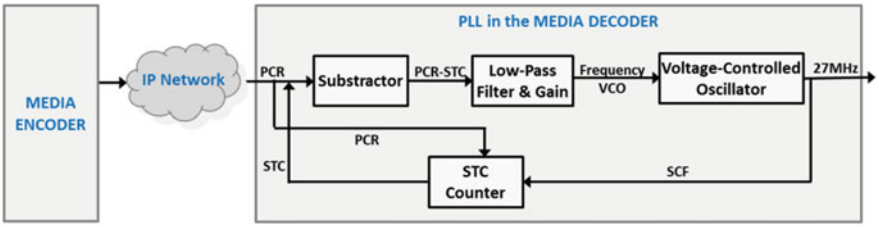


Fig. 15.4 MPEG-2 PLL (from encoder to decoder via an IP network) based on [1]

Figure 15.4 shows the PLL diagram. The PLL input is the PCR, and the *Subtractor* then calculates the difference between the PCR and the system time clock (STC). This difference is the input to the *Low-Pass Filter and Gain* which calculates the Voltage-Controlled Oscillator (VCO) Frequency. The VCO sets the SCF to the new VCO frequency. Finally, the *STC Counter*, using the new SCF value, estimates the new STC sent to the STC Counter value to repeat the process until the SCF equals the 27 MHz, which is the PCR frequency for MP2T streams.

As mentioned in the previous subsection, the $SCF_{MPEG\text{-}2}$ change rate should not be greater than 75×10^{-3} Hz/s.

15.3.2.2 System Clock Descriptor (SCD)

The System Clock Descriptor (SCD) has two purposes: to inform the media decoder about the use of an external clock to generate the timestamps and also to inform on the clock accuracy. The information is sent via the SCD within the MP2T packets. All the fields of the SCD are listed in Table 15.2.

When an external clock reference is used, the flag *external_clock_reference_indicator_flag* is set to one.

When clock accuracy frequency (CA_{freq}) needs to be greater than 30 ppm, part per million, then the descriptor's field clock accuracy integer (CA_{int}) is not equal to zero and the CA_{freq} is calculated from the CA_{int} and CA_{exp} following the equation [1]:

$$CA_{freq} = \begin{cases} 30\,\text{ppm} & if\ CA_{int} = 0 \\ CA_{int} \cdot 10^{-CA_{exp}}\,\text{ppm} & if\ CA_{int} \neq 0 \end{cases}$$

15.3.3 Timestamps

MP2T timestamps, used in combination with the relative clock, unrelated to global time allow the synchronization of the play-out of the multiple media streams of a program inside an MP2T stream (e.g., video, audio, and subtitles).

Table 15.2 System Clock Descriptor (SCD) fields [1]

Term	Bits	Meaning
Descriptor_tag	8	It signals the format of the System Clock Descriptor (SCD)
Descriptor_length	8	Descriptor bytes' size after the *descriptor_length* field. It is useful to know the end of the descriptor
External_clock_reference_indicator_flag	1	Flag that indicates the use of a reference external clock. It indicates that an external clock was used to generate the timestamps
Reserved	1	–
Clock_accuracy_integer	6	Integer of frequency accuracy of system clock, in parts per million (ppm) units. It is used to calculate clock accuracy if it is higher than 30 ppm
Clock_accuracy_exponent	3	Exponent of frequency accuracy of system clock (ppm). It is used to calculate clock accuracy if it is higher than 30 ppm
Reserved	3	–

In MP2T streams, there are two main timestamps: Decoding Timestamp (DTS) and Presentation Timestamp (PTS). Both fields run at a frequency of 90 kHz and are represented in two 33-bit fields located at the PES Header (See Fig. 15.3).

Their presence is signalled by the PTS_DTS_flag (2-bit), also located at the PES Header. The meaning of the values of this flag is described in Table 15.3.

To understand the reason behind the need for two different timestamps, the fundamentals of video coding need to be explained. Audio MDUs use instant decoding, and therefore, the PTS and DTS values are equal. Then, only the PTS is present in the PES Header. On the other hand, video requires of two different PTS and DTS values, because in MPEG-2 video coding includes three types of frames: Intra-frame (I-frame), predictive frame (P-frame), and Bidirectional Predictive (B-frame).

MPEG-2 uses the similarity between video frames to improve video compression. For example, when the difference between two video frames is minimal, there

Table 15.3 Values of PTS_DTS_Flag

Value	Meaning	Usability
00	Neither PTS nor DTS is present in PES	No timestamps are found in the PES Header
01	Forbidden	No case will have DTS without PTS
10	PTS present in PES	Used by audio PES or video B-frames when decoding time equals the presentation time
11	PTS and DTS present in PES	Used by video I- and P-frames when decoding time differs from presentation time

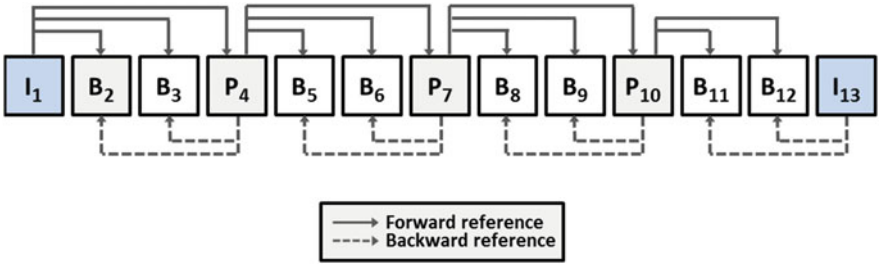


Fig. 15.5 GOP containing I-, P-, and B-frames [2]

is no need to encode the entire video frame again. MPEG-2 notes only the differences between frames. A Group of Picture (GOP) is a group of I-, P-, and B-frames.

An I-frame does not rely on any other video frame for decoding. However, a P-frame needs a previously arrived/decoded I- or P-frame to be decoded before being decoded itself. Finally, a B-frame requires I- and P-frames which are presented before and after the B-frame itself. These I- and P-frames need to be decoded previously to the decoding of the B-frame.

Following the graphic example in Fig. 15.5, for example, the I-frames, I_1 and I_{13} have no reference from other frames. A P-frame, for example P_4, has references from I_1 to be decoded. Also, a B-frame, such as B_3, has references from I_1 and P_4.

Figure 15.6 shows a hypothetical example of timestamps related to frames and the PCR values among video frames. The PCR values run at 27 MHz and the timestamps at 90 kHz. DTS and PTS have different values when frames need to be decoded previously to the presentation.

In the case of a video stream which has no B-frames, the decoding order matches the presentation order. Also, when talking about audio frames, DTS always equals PTS because instant decoding time is assumed and the presentation order is also the decoding order for audio MDU. Finally, in B-frames only PTS is present in the MP2T packet because DTS and PTS are equal.

The timestamps always refer to the first AU within the PES payload. Every PES contains an AU.

PTS and DTS can be obtained based on the $SCF_{MPEG\text{-}2}$ and the presentation and decoding time, as shown in the following equations (being the integer division) [1]:

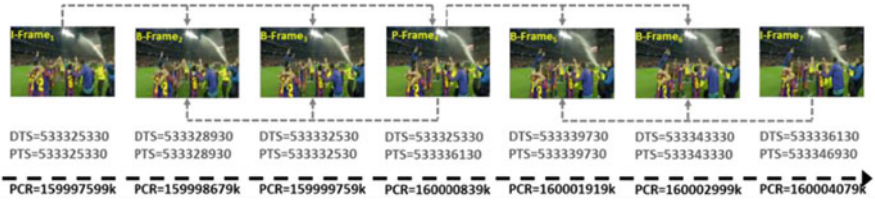


Fig. 15.6 Example of frame sequence with timestamps and PCR timeline values [2]

$$PTS(k) = \frac{(SCF_{MPEG-2} \cdot (tp_n(k)))}{300} \%2^{33}$$

$$DTS(j) = \frac{(SCF_{MPEG-2} \cdot (td_n(j)))}{300} \%2^{33}$$

One last type of timestamp is located in the Adaptation Field: the *DTS_next_AU* (33-bit) field, in which frequency runs at 90 kHz. It is used to support splicing taking place within the MP2T stream. Splicing is the action of uniting two media streams, concatenating the end of one media stream to the beginning of another one. The field *DTS_next_AU* indicates the DTS value of the attached media stream. The field *seamless_slice_flag* (1-bit) indicates the presence of *DTS_next_AU* field in the Adaptation Field. Figure 15.2 shows the position of both fields in the MP2T packet within the Adaptation Field, with both fields being optional.

15.3.4 T-STD (Transport System Target Decoder)

The tools to accomplish the synchronized media play-out of MP2T streams are used at the T-STD, by means of the D-PLL and clock references and timestamps values. An MP2T program enters the T-STD, following the high-level example shown in Fig. 15.7, and the different media types are sent to their respective decoding buffers (e.g., one video and one audio stream). The T-STD extracts all the timing information, i.e., PCR values, sent directly to the D-PLL, and the timestamps, sent to the respective comparators for either audio or video media types.

The D-PLL, with the PCR values as input, calculates the STC, the decoder's clock. The STC values are also sent to the timestamp's comparators to time/signal when the media is decoded or presented (e.g.,, the system differs in the case of video or audio).

The extraction of the PCR and timestamps could be performed by a single extraction process, like in [2], but due to the location of the clock references, in the Adaptation Field, and timestamps in PES, it has been decided to separate the functions in the diagram.

In the case of the audio stream, DTS equals PTS due to the instant decoding of the audio MDUs. Therefore, only the Audio PTS is sent to the *Comparator STC/DTS = PTS* where the value is checked against the STC values. When the values match a signal, the audio MDU is sent to the *Audio Decoder* to decode and instantly present it. Thus, there is no Audio Presentation Buffer in the T-STD.

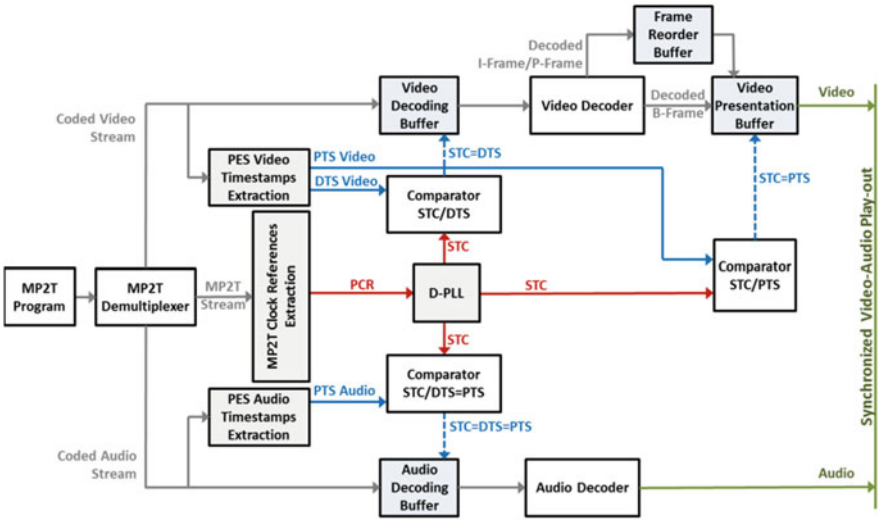


Fig. 15.7 T-STD of a program stream with one video and audio media stream. Modified from/based on [2]

In the case of the video stream, the decoding time could differ from the presentation time. The video MDUs leave the *Video Decoding Buffer* to be decoded by the *Video Decoder*, when STC equals DTS. In the case of I- and P-frames, once they are decoded they are sent to the *Frame Reorder Buffer*[1] to be re-arranged before being sent to the *Video Presentation Buffer*. B-frames are sent directly to the *Video Presentation Buffer*. All video frames wait in the *Video Presentation Buffer* until the STC equals PTS, which indicates the presentation time, the play-out time, which matches the presentation time of the audio MDUs sent by the *Audio Decoder*.

15.3.5 Delivery of Timeline for External Data

In 2015, Amendment 1 to the ISO-IEC 13816-1 [3] was published, providing the tools to deliver a timeline within MP2T streams for external data. MP2T streams convey multiple programs, and therefore, the tool provides a way to link the MP2T stream programs to an external timeline conveyed within the MP2T stream. This timeline is not impacted by PCR discontinuities, and as such, it provides a

[1]The presence of the *Video Presentation Buffer* may be not needed, as happens in [2], as the *Frame Reorder Buffer* could take its role, although the presence of the *Video Presentation Buffer* shows the concept of the presentation timestamp more clearly.

consistent timeline within the MP2T stream [3]. The timeline used is called Timeline and External Media Information (TEMI).

This timeline uses descriptors as the main tool to create the timeline. Descriptors are '*structures used to carry various features of the timeline or other information*' [3]. In particular, the timeline uses the *temi_timeline*, *temi_location,* and *temi_base_URL* descriptors.

The *temi_location_descriptor* '*is used to signal the location of external data that can be synchronized with the program. It conveys several locations and their type (optionally including MIME types), along with the ability to signal upcoming external data association though a countdown until activation of the external data*' [3]. This descriptor has a field *timeline_id* (7-bit) that conveys '*a unique identifier for this content location*' [3].

The *temi_base_url_descriptor* '*is used to assign a default base URL to all location descriptors.*' It contains a list of *base_url_path* (8-bit), which provides '*Base URL common to all following location descriptors*' [3].

Finally, the *temi_timeline_descriptor* '*is used to carry timing information that can be used to synchronize external data*' [3]. This descriptor, as the *temi_location_descriptor*, contains the *timeline_id* field (8-bit), which '*indicates the active timeline*' [3] and is acknowledged in the *temi_location_descriptor*.

The type of TEMI Descriptor is acknowledged by the *af_descr_tag* field (8-bit) common to the three TEMI Descriptors. The *af_descr_tag* value for the *temi_location_descriptor is 0x05*, for *temi_base_url_descriptor* it is 0x06, and finally for *temi_timeline_descriptor it is 0x04*.

There are two mechanisms to convey the TEMI Descriptors: The first one includes those in the Adaptation Field, when the descriptors do not increase in excess the size of the Adaptation Field; the second one, which is based on including them in a PES packet within the MP2T packet, allows for the transport of considerably more descriptors data. In Fig. 15.8, the two options are shown.

To provide the Adaptation Field with the tools to convey TEMI Descriptors, extra information is added. The *af_descriptor_not_present_flag* field (1-bit) informs about the TEMI Descriptors' presence within the Adaptation Field.

To provide the tools to convey TEMI Descriptors within a PES packet, and thus within an MP2T packet, a new payload is defined: TEMI Access Units (TEMI_AU). Descriptors will be conveyed within the TEMI_AU. This new type of stream is identified as a private_stream_1[2] within the PMT table (*stream_type* = 0x26). The structure of a TEMI_AU can be found in Table 15.4.

The time-related fields of the *temi_timeline_descriptor* are defined in Table 15.5. *Timeline_id* field is the field that links the *temi_timeline_descriptor* with the *temi_location_descriptor*. The descriptor may include a media timestamp, a Network Time Protocol (NTP) timestamp, a Precision Time Protocol (PTP) timestamp, or a timecode.

[2] '*Private data is any user data which is not coded according to a standard specified by ITU-T*| *ISO/IEC and referred to in this Specification*' [1].

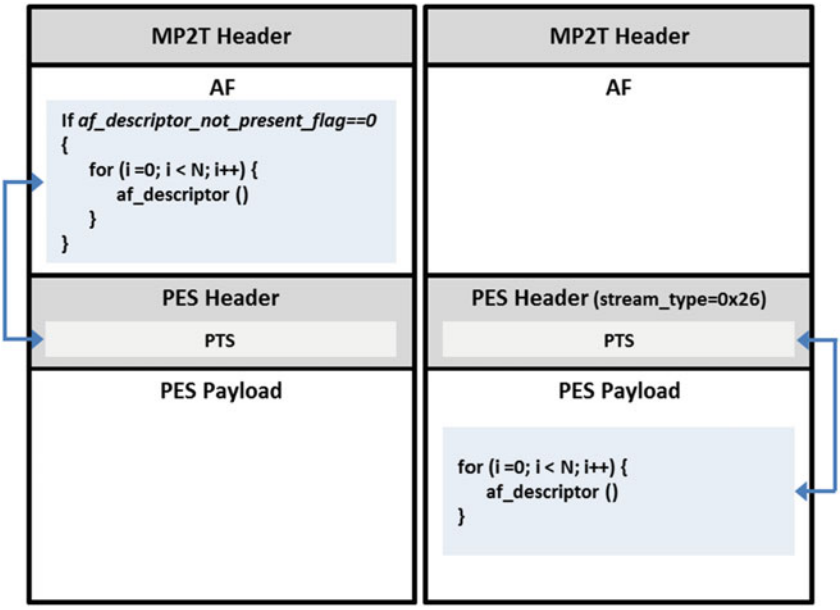


Fig. 15.8 Location of TEMI Descriptors

Table 15.4 TEMI_AU [3]

TEMI_AU {	
CRC_flag	(1-bit)
reserved	(7-bit)
for (i=0; i<N; i++) {	
af_descriptor();	
}	
if (CRC_flag) {	
CRC_32	(32-bit)
}	
}	

If *has_timestamp* is set, then the fields *timescale* and *media_timestamp* are present. If *has_ntp* is set, then *ntp_timestamp* is present. If *has_ptp* is set, then *ptp_timestamp* is present. Finally, if *has_timecode* value is 0x01, then the fields drop, *frames_per_tc_seconds*, duration, and *short_time_code* are present, and if *has_timecode* value is 0x02, then the fields *drop*, *frames_per_tc_seconds*, *duration,* and *long_time_code* are present.

It is clear that if an *ntp_timestamp* or a *ptp_timestamp* is present in the TEMI_AU, then the values of NTP or PTP (as established in clause 6 of IETF RFC 5905) are linked to the PTS value conveyed in the PES Header of the same MP2T packet. The NTP_i and PTP_i values of the *i*th PES packet can be calculated by the following equations [3]:

Table 15.5 Time-related fields of temi_timeline_descriptor [3]

Field	Bits	Definition in [3]
has_timestamp	2-bit	'*Indicates a media timestamp will be carried in this descriptor, and indicates its type. Value 0 means no media timestamp is present, value 1 means a 32 bit media timestamp is present, value 2 means a 64 bit media timestamp is present, value 3 is reserved*'
has_ntp	1-bit	'*When set to 1, indicates that a NTP timestamp will be carried in this descriptor*'
has_ptp	1-bit	'*When set to 1, indicates that a PTP timestamp will be carried in this descriptor*'
has_timecode	2-bit	'*When set to '00', indicates that no frame timecode is present, when set to '01' indicates a short frame timecode is present, when set to '10' indicates a long frame timecode is present, value '11' is reserved*'
timeline_id	8-bit	'*Indicates the active timeline. timeline_id values in the range [0, 0x7F] are identified in a temi_location_descriptor*'
timescale	32-bit	'*Indicates the timescale used to express the media_timestamp field in this message*'
media_timestamp	32-bit or 64-bit	'*Indicates the media time in timescale units corresponding to the PES PTS value of this packet for the timeline identified by the last temi_location_descriptor received*'
ntp_timestamp	64-bit	'*A full 64 NTP timestamp as defined in clause 6 of IETF RFC 5905*'
ptp_timestamp	80-bit	'*A full 80 bits PTP timestamp as defined in IEEE 1588v2*'
drop	1-bit	'*Drop-frame indication, as defined in clause 5 of IETF RFC 5484*'
frames_per_tc_seconds	15-bit	'*The number of those frames that make a time-code second, as defined in clause 5 of IETF RFC 5484*'
duration	16-bit	'*The duration in ticks of a frame expressed in the timescale of 90000 ticks per seconds, as defined in clause 5 of IETF RFC 5484*'
short_time_code	24-bit	'*A short 32 time code as defined clause 6.2 of IETF RFC 5484*'
long_time_code	64-bit	'*A full 64 time code as defined clause 6.2 of IETF RFC 5484*'

$$NTP_i = \frac{PTS_i - PTS_0}{90000} + NTP_0$$

$$PTP_i = \frac{PTS_i - PTS_0}{90000} + PTP_0$$

If the media timestamp is present in TEMI_AU, then the PTS is linked to a media time in timescale units. Therefore, the media timestamp (MTP_i) of the *i*th PES packet, also called media time, can be calculated by the following equation [3]:

$$MTP_i = \frac{PTS_i - PTS_0}{90000} + \frac{MTP_0}{timescale}$$

The last timeline association is done using the relation between *short_time_code*/*long_time_code* and *drop*, *frames_per_tc_seconds* and the *duration*. The definition and use of the fields are described in Table 15.5.

15.4 Time and Timing Within MPEG-4 Systems

15.4.1 MPEG-4 Systems

Before examining time and timing in MPEG-4 systems, the principal concepts of MPEG-4 need to be introduced. MPEG-4 is a layered model which maintains independence between layers. The closest layer to the encoder is the Compression Layer (CL), responsible for encoding the media (e.g., audio and video). The Sync Layer (SL) adds the time and timing tools to perform the play-out synchronization. Clock references and timestamps are encoded within this layer. Finally, the Delivery Layer (DL) is responsible for the transport/delivery of the MPEG-4 stream to end users.

In Fig. 15.9, the MPEG-4 layers are shown. The SL conveys the smallest timed unit. It conveys MDUs and other types of information needed for the decoder to decode the MPEG-4 stream, such as elements of the Object Description Framework.

The Object Description Framework establishes the media streams in the MPEG-4 stream and the relation between them. The Object Description Framework also sets a structure of descriptors to define the MPEG-4 stream. The Initial Object Descriptor contains links to the Scene Description Stream (containing the Interactive Scene Description) and the link to the Object Descriptor Stream (containing the audio–video objects). The scene description sets the spatiotemporal relationships, via object descriptors, between audio–visual objects.

The scene structure to define a scene has two levels of description information: the structural and the media object descriptions. The structural level includes the Binary Format for Scenes (BIFS), which defines the time and space organization of the media objects within a scene object. Additionally, the media object description level defines the location, configuration, and synchronization of the media streams involved in the MPEG-4 stream [5].

The MPEG-4 CL is based on media objects. A video (e.g., audio and visual streams) is made of scenes, and each scene is made of multiple objects. This media object-oriented coding approach is one of the main differences between MPEG-2 and

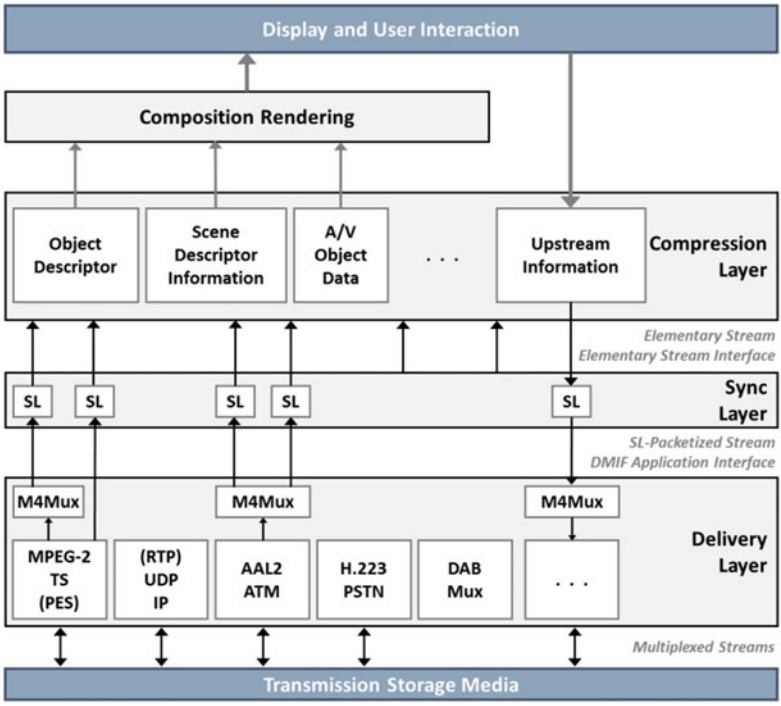


Fig. 15.9 MPEG-4 high-level layers diagram

MPEG-4. Taking an example from Fig. 15.10, the football scene described there is made of five visual objects and two audio streams. The scene objects are the three football players, the ball, and the background (the background is treated in Fig. 15.10 as one, although it could also be treated as two different objects, the crowd and the grass). The main objective of dividing a scene into multiple objects is the possibility of encoding each visual object in different video qualities. For example, the crowd does not need to have as much detail/quality as the players because viewers focus on the ball and the players, not on the background of the football match.

The Object Description Framework links, using the BIFS, all descriptors with the media streams within an MPEG-4 stream.

The Scene Description Stream conveys the Interactive Scene Descriptions, each of them providing the object structure of the scene. In the case of Fig. 15.10, the scene descriptor provides two possible video qualities, Base and Enhancement Layer Scene. The end user decoder selects to display one or the other video qualities based, for example, on the user's receiver characteristics or bandwidth availability.

BIFS follows a tree structure which provides the framework to organize the media scene objects via nodes. In Fig. 15.10, there are two intermediate nodes: *Movie Texture A*, providing object structure for the Base Layer Scene, and *Movie Texture B*, providing object structure for Base and Enhancement Layer Scene.

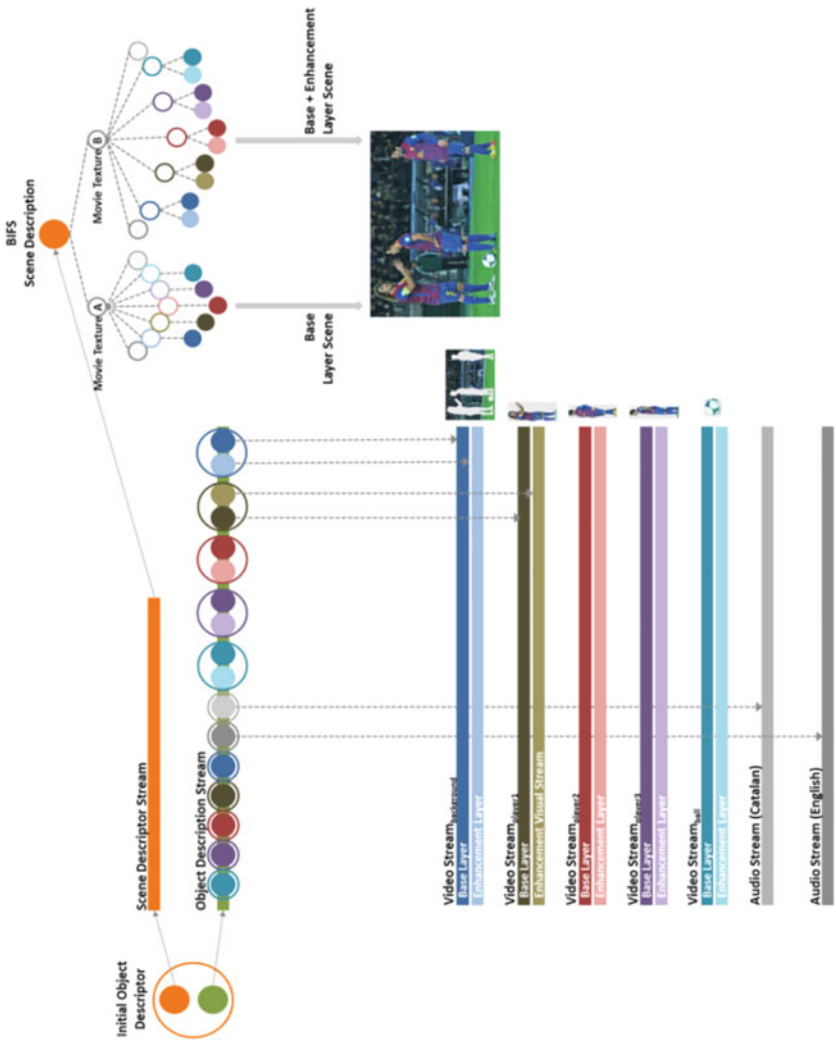


Fig. 15.10 Example of an MPEG-4 stream and its object descriptor framework. Following example from Fig. 15.2 in [4]

Regarding video, the *Movie Texture A* provides links to the object descriptors to the Base Layer visual streams, whereas the *Movie Texture B* provides links to the object descriptors of the Base and Enhancement Visual Streams. Finally, both Movie Textures provide links to the provided audio streams (Catalan and English). In this example, there is only one audio quality for each audio stream.

The SL layer provides the media synchronization tools for the content of the CL which is made up of different content types, such as media content, Audio and Video Object Data, Object and Scene Descriptors.

15.4.2 Clock References

The same principle for MP2T streams and clock references applies to MPEG-4 timelines. Thus, the goal is to replicate the encoder's clock at the decoder's end user side. Figure 15.11 shows the location of the time references in the encoder's and decoder's time system. The system time base (STB), decoder's clock system, replicates the encoder's clock system, object time base (OTB), using the Object Clock Reference (OCR) information conveyed in the SL stream.

In Table 15.6, the main characteristics for both OTB and STB time systems are listed, and in Fig. 15.12, we see how values are linked within an MPEG-4 SL and the SLConfig Descriptor.

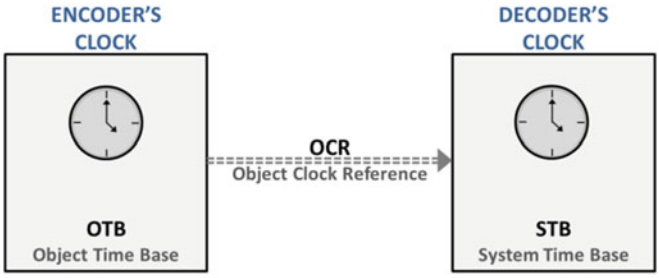


Fig. 15.11 MPEG-4 clock references location [6]

Table 15.6 Main OTB and STB characteristics

OTB	STB
Data stream notion of time	Terminal notion of time
Resolution is defined by the application or the profile	Resolution is implementation dependent
Timestamps in the data stream relate to the OTB	Terminal actions relate to the STB
OTB is sent to the terminal through the OCR values	

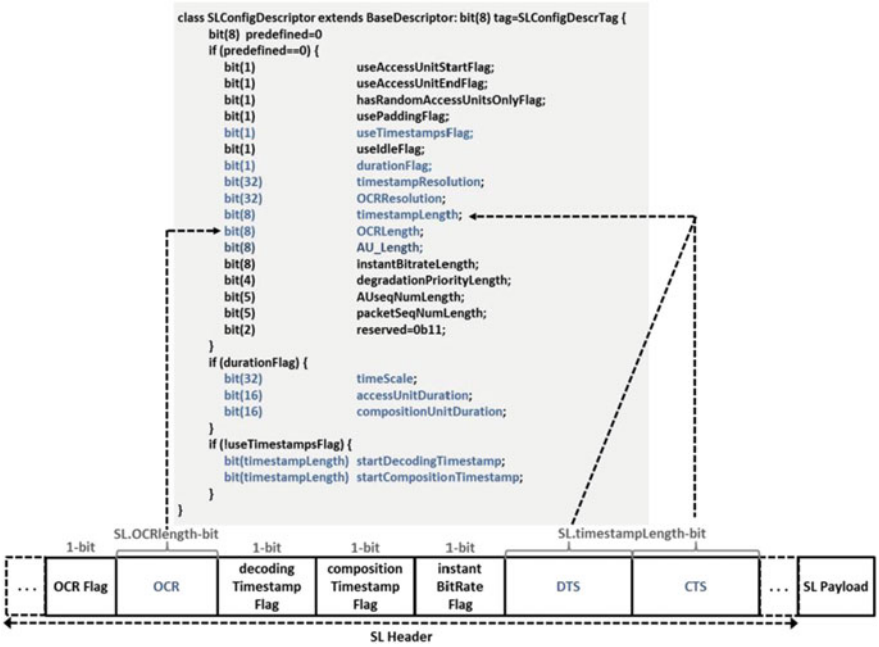


Fig. 15.12 MPEG-4 Part 1 bitstream and its time-related fields and descriptors [6]

The OCR values are the clock references which have variable length and resolution. The length and resolution are specified at the *SLConfig Descriptor*. Within the descriptor, the fields *OCRresolution* (OCR_{res}) (32-bit) and *OCRlength* (OCR_{len}) (8-bit) provide this information about the SL OCR fields. The presence of the OCR value in the SL stream is signalled by the *OCRflag* (1-bit). In Table 15.7 are found all the terms used in MPEG-4 time-related formulae.

The following is a key difference between MP2T and MPEG-4 clock references. The former has a fixed resolution and fixed bit length, while the latter can have different values set by the encoders.

To reconstruct the time related to the OTB value, named OTB reconstructed time (t_{OTBrec}) at the decoder's end, the following formula is applied [7]:

$$t_{OTBrec} = \frac{OCR}{SL.OCR_{res}} + k \cdot \frac{2^{OCR_{len}}}{SL.OCR_{res}}$$

where '*k is the number of times the OCR counter has wrapped around (number of times the value reaches the maximum and starts over)*' [7]. The value of k is set when the minimum value is found for the following expression.

$$|t_{OTBestimated} - t_{OTBrec}(k)|$$

Table 15.7 Terms for MPEG-4 timelines related to Sect. 15.4. Table from [7] and definitions from [2]

Term	Meaning
$AU_{duration}$	'*the duration of an access unit*' in timescale units
AU_{time}	AU duration in time units (duration)
CTS	'*Each Composition Unit (CU) has an associated nominal composition time, the time at which it must be available in the composition memory for composition*'
CU_{time}	CU duration in time units (seconds)
$CU_{duration}$	'*the duration of a composition unit*' in timescale units
DTS	'*Each AU has an associated nominal decoding time, the time at which it must be available in the decoding buffer for decoding*'
FCR(I″)	'*the time encoded in the fmxClockReference (FCR) in units of the FCR resolution*'
FCR_{res}	The resolution of FCR in cycles per second
fmxRate(i)	'*It indicates the rate specified by the fmxRate field for byte i*'
i	'*the index of any byte on the M4Mux stream for i″ < i < i′*'
i″	'*the index of the byte containing the last bit of the most recent fmxClockReference field in the M3Mux stream*'
k	'*k is the number of times that the objectClockReference counter has wrapped around*'
m	'*an integer value denoting the number of wrap-arounds*' for timestamps values
$SL.OCR_{len}$	'*the length of the objectClockReference field in SL packet headers*'
$SL.OCR_{res}$	'*the resolution of the object time base in cycles per second*'
SL.timescale	'*used to express the duration of access units and composition units. One second is evenly divided in timescale parts*'
$SL.TS_{len}$	'*the length of the time stamp fields in SL packet headers*'
$SL.TS_{res}$	'*the resolution of the time stamps in clock ticks per second*'
$T_{estimated}$	'*current estimated value of the OTB*'
$T_{OTBrec}(k)$	OTB reconstructed time for value *k*
$T_{ts}(m)$	Timestamp for value *m*

The time system should have a resolution to be able to distinguish two OCR values. Therefore, two OTB time instants should have a difference greater than the following expression [7]:

$$\frac{1}{SL.OCR_{res}}$$

OCR resolution is not fully used by the decoder if the timestamp resolution is higher. As explained in Sect. 15.3.4, the timestamps resolution is also declared at the SLConfig Descriptor.

15.4.2.1 Clock Reference Stream

An SL stream can also be used to share a timeline between multiple streams. As such, this special type of SL stream called Clock Reference Stream does not contain a payload. It provides OCR values based on the length and resolution of its related SLConfig Descriptor, OCR_{res}, and OCR_{len}.

The media streams, linked to the time provided by the OCRs within the Clock Reference Stream, use these values (clock references) at the decoder to feed the decoder's STC, so their play-out processes can be synchronized via linked media timestamps.

15.4.3 Timestamps

As with MP2T, there are timestamps in MPEG-4. These are based on AUs (the equivalent to MDU in MPEG-2) and how they are encoded. It has been described in the MP2T timestamps section (Sect. 15.3) how the encoding of the audio and video influences the timing (timestamp system) of audio, as it is based on instant decoding, and video, because of the use of different video frames.

The MPEG-2 video frames concept is not present in MPEG-4. Instead, it is replaced with video slices, basically because of the object-oriented video encoding approach. The principle is the same, although the idea is that video slices are not presented, as frames are in MPEG-2, but composed, because a single image frame is formed by multiple video objects which, composed at the same time, present the entire video image/frame. Thus, in MPEG-4 video is structured as with video slices.

We have Decoding Timestamps (DTS) and Composition Timestamp (CTS) within the SL stream which follows the value characteristics advertised in the SLConfig Descriptor, *timestampLength* (TS_{len}) (8-bit), and *timestampResolution* (TS_{res}) (32-bit). The presence of timestamps is signalled by the fields *decodingTimestampFlag* (1-bit) and the *compositionTimestampFlag* (1-bit) in the SLConfig Descriptor fields.

The decoding time (t_D) and the composition time (t_C) are calculated by using the following expressions [7]:

$$t_D = \left(\frac{DTS}{SL.TS_{res}} + m \cdot \frac{2^{SL \cdot TS_{len}}}{SL.TS_{res}} \right)$$

$$t_C = \left(\frac{CTS}{SL.TS_{res}} + m \cdot \frac{2^{SL \cdot TS_{len}}}{SL.TS_{res}} \right)$$

Taking into account that both DTS and CTS timestamps have the same resolution, TS_{res}, and field length, TS_{len}, the general formula for timestamps time, decoding, or presentation time can be used [7]:

$$t_{ts}(m) = \left(\frac{timestamp}{SL.TS_{res}} + m \cdot \frac{2^{SL.TS_{len}}}{SL.TS_{res}} \right)$$

In the last three equations, m is the number of wraps-around of the timestamp. The way to prevent ambiguity is to take the value of m that minimizes the following expression [7]:

$$|t_{OTBestimated} - t_{ts}(m)|$$

There is another way to calculate the decoding and composing timestamps, when the AU's time duration is constant in time units. This is achieved via the fields in the SLConfig Descriptor. If *durationFlag* in the SLConfig Descriptor is set, then the descriptor contains the fields *AccessUnitDuration* ($\text{AU}_{\text{duration}}$) (16-bit) and *CompositionUnitDuration* ($\text{CU}_{\text{duration}}$) (32-bit) and *timescale* (32-bit). Moreover, if the *useTimestampFlag* is not set, then the two fields *startDecodingTimestamp* and *startCompositionTimestamp* are present to indicate the initial timestamps used in combination with the $AU_{duration}$ and $CU_{duration}$ to gather timestamps of each AU.

The equations to obtain the information about the time, in seconds, of an AU and CU are based on the timescale [7]:

$$AU_{time} = SL.AUduration \cdot \frac{1}{SL.timescale}$$

$$CU_{time} = SL.CUduration \cdot \frac{1}{SL.timescale}$$

15.5 Time and Timing Within MPEG-DASH

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) is the MPEG standard media delivery technique over Internet, Adaptive Streaming over HTTP [8].

It is a pull-based client-driven media delivery protocol which relies on the most used protocols in Internet, TCP, and HTTP. MPEG-DASH accommodates the media delivery depending on the end user's receiver hardware, processing, and decoding capabilities. It also accommodates the network conditions such as connectivity and bandwidth.

The information in MPEG-DASH is divided into multiple media representations so the client receiver will select the media quality required for the play-out. Media is available in different formats and different format qualities, thus giving receivers the choice to select the best one depending on network, load conditions. The media is stored in HTTP servers and delivered in small media chunks.

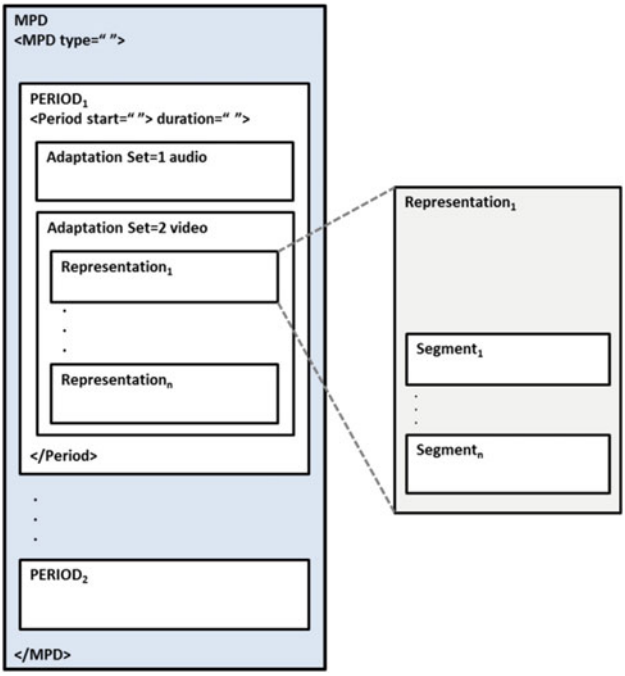


Fig. 15.13 MPD file high-level structure

15.5.1 MPEG-DASH

To achieve the adaptive streaming session, the Media Presentation Description (MPD) file is used. The MPD, using XML format, contains all the necessary information about the available media that can be requested by clients. The client utilizes the MPD file to select the proper Media Segments. The MPD allows the end user (client) to select, for example, the video or audio quality of the media best suited for the capabilities of the receiver, also taking into account the bandwidth conditions.

As defined in [8], media content is '*one media content Period or a contiguous sequence of media content Periods.*' Therefore, the MPD file provides all the structured information needed to achieve the streaming of the media, selected via HTTP Get requests and delivered via *segments containing short* periods of media content through HTTP Response messages.

There are two types of MPD, static and dynamic. Static MPD is used for stored media and dynamic MPD for live media. The MPD file, following the XML format, is composed of tags. The entire file is contained in an MPD tag. Inside this tag, the file is subdivided into *Periods* of time. Each *Period* contains a list of *AdaptationSet* tags, each of which relates to each media stream such as audio and video. Each *AdaptationSet* contains the multiple representations of the same media content type

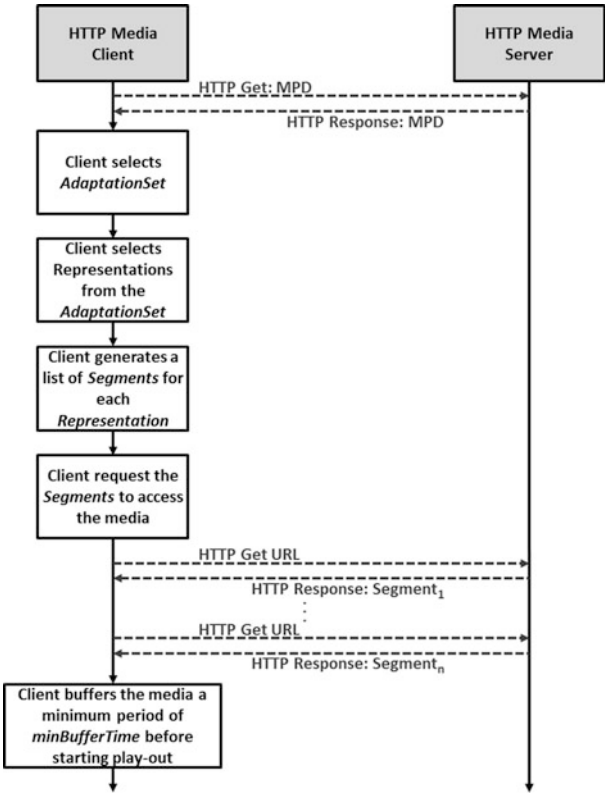


Fig. 15.14 High-level MPEG-DASH client behavior example from [6]

but only one representation in the *AdaptationSet* is selected for the media play-out. Each *Representation* lists the segments required to play the duration of the related period of time and the audio and video coding formats. The high-level structure of an MPD file is shown in Fig. 15.13.

The media is stored in small segments on a conventional HTTP server, and the URL location of each segment and each representation is indicated in the MPD file, so they can be fetched by end users requiring the media. These segments are typically from 2 to 10 s long. They can be stored in multiple media encoding methods, such as MP4 and MP2T streams. Therefore, those segments will include their related clock references and timestamps based on the media standard used.

One example of client behavior is shown in Fig. 15.14. The client selects the media content and sends a HTTP Get message to the HTTP server to fetch the MPD file. The HTTP media server sends the media-related MPD file to the client. Then, the client has all the required information, via the MPD file, to find/retrieve the media that best suits the end user receiver and the network connectivity. The client selects an *AdaptationSet*, from this particular *AdaptationSet,* it selects the

best-suited media *Representation*, and from that, it selects the list of the Media Segments to be requested. Once the client has the list of the media-related segments, it sends to the HTTP media server a number of HTTP Get URL messages to fetch the Media Segments which will be delivered via HTTP Response messages by the HTTP media server. Only when a minimum time is received by the client, *minBufferTime*, does the media play-out does begin at the end user receiver.

Unlike MP2T and MPEG-4 files, MPEG-DASH does not have traditional clock references and timestamps, but it contains time information at different levels of the MPD file related to the entire media stream and the Media Segments, as explained in the next section.

15.5.2 Time-Related Information in MPEG-DASH

Time information within the MPD file is located at different levels. Some fields are located at MPD level, some at Period level and finally at segment level. The Period time-related fields are *start* and *duration*. Finally, at Media Segment level, using three formats, *segmentBase*, *segmentTemplate,* and *segmentList,* the time-related fields can be found *presentationTimeOffset, timescale, duration,* and *segmentTimeline*. The meaning of all time-related fields at all levels is described below.

Information that applies to all the HTTP media delivery can be found in the MPD, including multiple attributes which provide time-related information. As previously mentioned, time-related information can be found at different levels of the MPD file. In Table 15.8, the time-related MPD attributes are listed along with their definitions. Only the attribute *minBufferTime* is obligatory, and *mediaPresentationDuration* is obligatory for static MPD type, whereas *availabilityStartTime* is only obligatory for dynamic MPD type.

The *Period* is a '*set of media content components that have a common timeline as well as relationships on how they can be presented*' [8]. In the *Period* tag, the attribute *duration* provides the duration of the media related to the information within the tag. The *Period* contains two optional time fields; *start* '*specifies the PeriodStart time of the Period. The PeriodStart time is used as an anchor to determine the MPD start time of each media segment as well as to determine the presentation time of each access unit in the Media Presentation Timeline*' [8], and *duration* '*if present specifies the duration of the Period to determine the PeriodStart time of the next Period*' [8].

Inside a time *Period*, every *AdaptationSet* has multiple media *Representations* (each media type in different encoding schemes) and each contains the Media Segments.

There are three segment formats: *segmentBase*, *segmentList,* and *segmentTemplate*. *The segmentBase* is used for representations with one Media Segment but for

Table 15.8 MPD time-related attributes. Definitions from [8]

MPD Attribute	Meaning
Availability Start Time	'*For @type = 'dynamic' this attribute shall be present. In this case, it specifies the anchor for the computation of the earliest availability time (in UTC) for any Segment in the Media Presentation*'
	'*For @type = 'static', if present, it specifies the Segment availability start time for all Segments referred to in this MPD. If not present, all Segments described in the MPD shall become available at the time the MPD becomes available*'
Availability End Time	'*It specifies the latest Segment availability end time for any Segment in the Media Presentation. When not present, the value is unknown*'
Media Presentation Duration	'*If this attribute is not present, the duration of the Media Presentation is unknown. In this case the attribute MPD@minimumUpdatePeriod shall be present*' '*This attribute shall be present when the attribute MPD@minimumUpdatePeriod is not present*'
Minimum Update Period	'*If this attribute is present, it specifies the smallest period between potential changes to the MPD. This can be useful to control the frequency at which a client checks for updates* *If this attribute is not present, it indicates that the MPD does not change*'. '*If MPD@type is 'static', @minimumUpdatePeriod shall not be present*'
minBuffer Time	'*It specifies a common duration used in the definition of the Representation data rate*'
timeShift Buffer Depth	'*specifies the duration of the time shifting buffer that is guaranteed to be available for a Media Presentation with type 'dynamic'. When not present, the value is infinite*' *If MPD is static this attribute is undefined*
Suggest Presentation Delay	'*when @type is 'dynamic', it specifies a fixed delay offset in time from the presentation time of each access unit that is suggested to be used for presentation of each access unit*'
	'*when @type is 'static' the value of the attribute is undefined and may be ignored*'
Max Segment Duration	'*It specifies the maximum duration of any Segment in any Representation in the Media Presentation, i.e. documented in this MPD and any future update of the MPD. If not present, then the maximum Segment duration shall be the maximum duration of any Segment documented in the MPD*'
Max Subsegment Duration	'*It specifies the maximum duration of any Media Subsegment in any Representation in the Media Presentation. If not present, the same value as for the maximum Segment duration is implied*'

multiple segments within a representation the *segmentList* or *segmentTemplate* is used, sharing the same *segmentBase* information. The *segmentList* contains a list of consecutive segments with the URLs where segments are located; see Fig. 15.15.

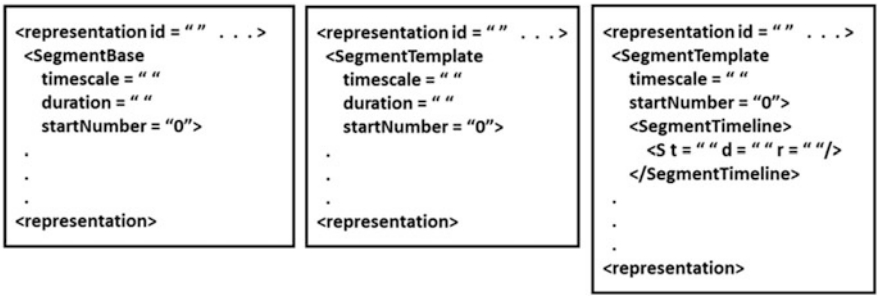


Fig. 15.15 Segment timelines in MPEG-DASH

The following time-related optional fields are used in segments: *duration*, *timescale,* and *presentationTimeOffset*. The *timescale* '*specifies the timescale in units per seconds to be used for the derivation of different real-time duration values in the Segment Information*' [8], the *presentationTimeOffset* '*specifies the presentation time offset of the Representation relative to the start of the Period,*' and *duration*[3] '*specifies the constant approximate Segment duration*' [8].

When the segments do not have the *duration* attribute, the *segmentTimeline* can be used, which '*specifies Segment start time and duration for a contiguous sequence of segments of identical duration*' [8] and provides three fields [8]. First, the optional field *t* which '*specifies the MPD start time, in @timescale units, the first Segment in the series starts relative to the beginning of the Period*' [8]. Second, the mandatory attribute *d* which '*specifies the Segment duration, in units of the value of @timescale*' [8]. Third, the optional attribute *r*, with default value zero, which '*specifies the repeat count of the number of following contiguous Segments with the same duration*' [8].

15.5.3 Timelines in MPEG-DASH Linked to MP2T Streams

Media to be streamed using MPEG-DASH can be stored in different container formats, such as MP2T. This section focuses on how the MPEG-DASH segments relate to MP2T streams and how the timelines of clock references are included.

There are just a few requirements for segments to be stored in MP2T format. First, a segment must contain complete PES packets and complete MP2T packets. Second, a segment can only contain one program. Third, the segment initialization information should not need information from previous segments. Finally,

[3]For segmentBase, the MultipleSegmentBaseInformation is used, which 'specifies multiple Segment base information' [8].

Table 15.9 MPEG2TSPCRInfoBox Syntax

```
aligned(8) class MPEG2TSPCRInfoBox extends Box('pcrb', 0) {
  unsigned int(32) subsegment_count;
  for( i=1; i <= subsegment_count; i++){
    unsigned int(42) pcr;
    unsigned int(6) pad = 0;
  }
}
```

time-varying information must be contained in the Index Segment and/or between I_{SAP}[4] and I_{SAU}[5] [8]. Different types of segments are used in media representation based on MP2T: Index Segment, Initialization Segment, Bitstream Switching Segment and Media Segments.

- Index Segment is a '*segment that primarily contains indexing information for Media Segments.*'
- The Initialization Segment '*contains initialization information for accessing the Representation. The Initialization Segment shall not contain any media data with an assigned presentation time.*'
- Bitstream Switching Segment is a '*segment that if present contains essential data to switch to the Representation it is assigned to.*'
- Media Segment is a '*Segment that complies with media format in use and enables playback when combined with zero or more preceding segments and an Initialization Segment (if any)*' [8].

The Initialization Segment for MP2T streams contains mandatory information, such as MPEG-2 Program-Specific Information (MPEG-2 PSI), Program Association Table (PAT), Program Map Table (PMT), and PCR. '*If PCRs are carried on a media PID, the first packet of this PID is the first packet following the initialization data and carries a PCR*' [8].

A relevant box contained in the Index Segments for MP2T is the *MPEG-2 TS PCR Information Box* (*MPEG2TSPCRInfoBox*), also called *pcrb*, which '*signals the PCR information for MPEG-2 TS*' [8]. The syntax of the MPEG2TSPCRInfoBox can be found in Table 15.9.

PCR field in the MPEG2TSPCRInfoBox '*indicates the MPEG-2 TS PCR corresponding to the first (sync) byte of the first MPEG-2 TS packet in the media Subsegment corresponding to the current iteration*' [8]. It is relevant to indicate that this field is a 42-bit unsigned integer, while the PCR value is obtained from two fields PCR_{base} (33-bit) and the PCR_{ext} (9-bit) to obtain PCR equations used in Sect. 15.3.

[4] I_{SAP} is the index of the Stream Access Point (SAP). SAP is the 'position in a Representation enabling playback of a media stream to be started using only the information contained in Representation data starting from that position onward'.

[5] 'I_{SAU} is the index of the first access unit that follows I_{SAP}' [8]. Stream Access Unit (SAU).

When the Index Segment is present, then the *MPEG2TSPCRInfoBox* is present and the PCR information can be reached.

The internal presentation time (T_P) should be the PTS value used in the MP2T stream, and it is related to timestamps of the MPEG-DASH Media Segments in the Media Presentation Timeline.

Being i the AU index in the media stream, MP2T stream, and PTS the presentation timestamps of the related AU. T_P can be calculated by [8]:

$$T_P(i) = (PTS_A(i) - PTS_0) \cdot \frac{S}{90000}$$

being S the timescale value provided by the Index Segment. When the segment is present. S is the timescale value from the *sidx* box [8].

Conditions to convey PTS values in PES packets related to MPEG-DASH segments are the following [8]:

- '*PES packet starting at I_{SAU} shall contain only an integral number of access units and shall contain a PTS.*'
- '*T_{SAP} is defined to be the earliest presentation time of any access unit of the media stream such that all access units of the media stream with presentation time greater than or equal to T_{SAP} can be correctly decoded using data in the Representation starting at byte position I_{SAP} and no data before I_{SAP}.*'
- '*If a media stream contains a discontinuity, the $PTS_A(i)$ calculation assumes relative timing is maintained. Therefore, $PTS_A(i)$ will be adjusted by the difference between the value of PCR of the first PCR-bearing packet after the discontinuity and its interpolated PCR value (calculated using the pre-discontinuity PCR rate).*'
- '*In case of discontinuities, it is recommended to add a new Period to reset the value of @presentationTimeOffset.*'
- '*It is recommended that the @timescale attribute in the MPD matches the clock frequency S of the Elementary Streams.*'

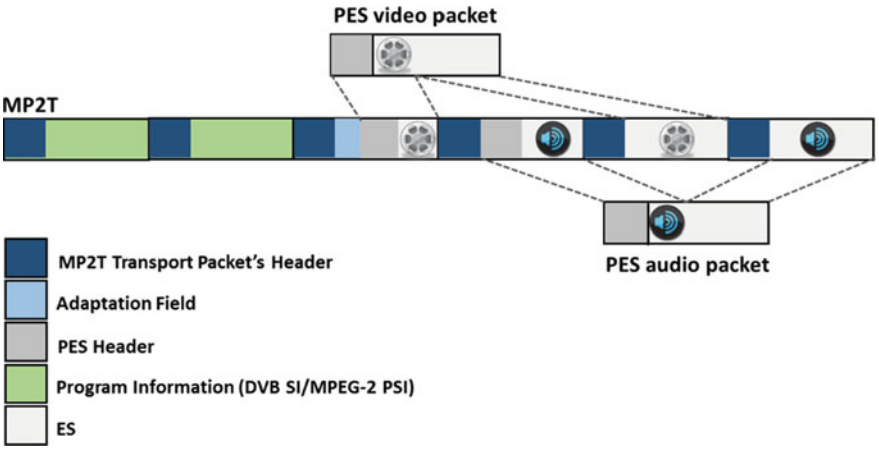


Fig. 15.16 DVB/MPEG-2 stream packets distribution [6]

Table 15.10 Timing in DVB SI and MPEG-2 PSI tables. Information gathered from [1, 9, 10]

		TABLE	Maximum interval (s)	Minimum interval (ms)
MPEG-2 PSI	Program Association Table	PAT	0.5	25
	Program Map Table	PMT	0.5	25
	Conditional Access Table	CAT	0.5	25
DVB SI	Network Information Table	NIT	10	
	Bouquet Association Table	BAT	10	
	Service Description Table	SDT	2	
	Event Information Table	EIT	2	
	Running Status Table	RST	–	
	Time and Date Table	TDT	30	
	Time Offset Table	TOT	30	
	Stuffing Table	ST	–	

- '*If the Segment Index ('sidx') box is present, then it is further recommended that the media stream for which the Segment Index ('sidx') box that appears first in the Index Segment is the Elementary Stream defining the value of the @time-scale attribute.*'

15.6 Time Transmission in DVB Systems

15.6.1 MPEG-2 PSI and DVB SI Tables

The MPEG-2 PSI and the DVB SI tables are two groups of tables that contain information about the programs and media related to a DVB/MP2T stream, providing support to the decoder's tasks.

They are conveyed in the MP2T stream, with each table within an MP2T packet. In Fig. 15.16, an example of the tables' distribution within an MP2T stream can be appreciated. Tables are resent over time within the MP2T stream. The time restrictions, gathered from different sources [1, 7, 8], are listed in Table 15.10.

Although MPEG-2 PSI and DVB SI are two different groups of information tables, their program information is linked via the *program_number* (16-bit) and the *transport_stream_ID* (12-bit). This relationship can be seen in Fig. 15.17.

The Program Association Table (PAT) is linked to the Service Description Table (SDT) via the *transport_stream_ID*. '*For each service in the multiplex, the PAT indicates the location (the Packet Identifier (PID) values of the Transport Stream (TS) packets for the corresponding Program Map Table (PMT)*' [9]; moreover, '*the PMT identifies and indicates the locations of the streams that make up each service and the location of the Program Clock References for a service*' [9].

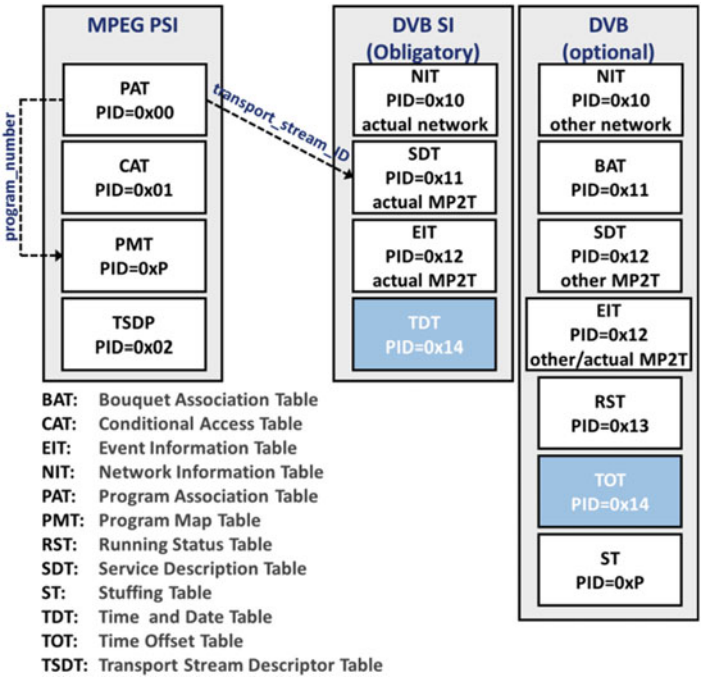


Fig. 15.17 High-level DVB SI and MPEG-2 PSI tables [9]. In blue the time-related tables [6]

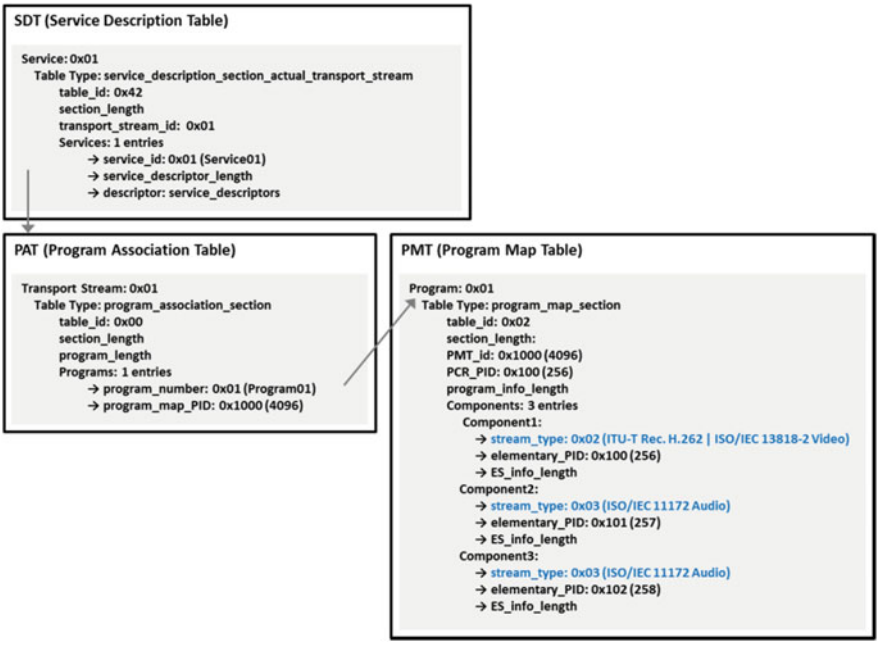


Fig. 15.18 High-level example of DVB SI and MPEG-2 PSI structure

The *PCR_id* (13-bit) field in the PMT gives the MP2T stream PID of the MP2T transport packets which contain the clock references, the PCR values for each program.

The PAT contains the program information. A Program contains multiple streams of different types. Following the example described in Fig. 15.18, the service is linked to a Transport Stream via the *transport_stream_id* field which connects the service to the related MP2T stream. The PAT table of the MP2T then provides the link of this Program to the related PMT table which lists the media streams of the Program. In Fig. 15.18, a Program is shown with one video stream (PID: 256) and two audio streams (PID: 257/258). The example also shows how the PMT contains the value of the PID stream containing the PCR values within the MP2T stream which, in this case, is the video stream (PCR_PID = 256).

15.6.2 Time-Related DVB SI Tables

Time is distributed to end users in an MP2T stream via two different tables: the Time and Date Table (TDT) and the Time Offset Table (TOT). The function is to distribute the UTC time from the media server to all end users.

The TDT conveys the UTC time in the *UTC_time* (40-bit) field using Modified Julian Date (MJD) format [11]. The TOT also contains the same *UTC_time* field, but with the possibility of adding a time offset based on the end user's location. The country information is encoded in the *country_code* (24-bit) and *country_region_id* (6-bit). The offset is signalled via the *local_time_offset_polarity* (1-bit) and the *local_time_offset* (16-bit) fields. The time to apply any time update is carried by *time_of_change* (40-bit), *next_time_offset* (16-bit) fields.

As an example, an end user located on Pamplona (Spain) at May 25, 2011, 19:39:25 will have the following values: *UTC_time* (D9 9A 19 39 25), *local_time_offset_polarity (*1), *local_time_offset* (1), *country_code* (ESP).

15.7 MMT: The Latest Media Delivery System

The MPEG Media Transport (MMT) protocol was approved in 2014 [12] and is designed to provide a tool for media delivery over heterogeneous networks [13] as well as could be used in broadcast media delivery platforms [13, 14]. MMT presents a new alternative to media delivery via MP2T media streams delivered using Real-Time Protocol (RTP) and User Datagram Protocol (UDP) [15].

As such, MMT aims to be the protocol used in multiple media delivery systems, via broadcast and broadband networks (heterogeneous networks).

The MMT protocol is an application layer protocol, which aims to provide streaming services via heterogeneous IP networks. Its main functionalities are [16]:

- A single MMT flow delivering multiple MMT Assets (protocol-level multiplexing).
- Adaptive to network jitter.
- To provide QoS support.

According to [14], MMT is not only intended for media delivery over heterogeneous broadband networks, but also media delivery broadcast systems.

MMT is specially designed for media delivery in heterogeneous IP networks, but also in broadcast systems. The common functionalities between broadband and broadcast media delivery are: '*synchronized delivery, inter-media synchronization, multiplexing assets into a single flow and buffer management*' [16].

One of the drawbacks of MP2T is that there is no efficient way to insert personalized advertisements or a personalized audio stream into the MP2T, because it requires multiplexing, re-multiplexing, and/or splicing [15].

Another drawback from MP2T is the use of MPEG-2 Program Streams, also defined in [1], as media storage, but the multiple payloads used in the RTP protocols make the storage of RTP payload complex. As a result, it requires dedicated process for every payload data. One of the challenges faced by MMT is to provide an 'easy *conversion between the format for storage and the format for the packetized delivery*' [15].

15.7.1 MMT Architecture

As can be seen in Fig. 15.19, MMT architecture and functionalities are divided into multiple layers with different functionalities [16, 17]:

- Coding media area (Media Coding Layer).
- Encapsulation function area (Encapsulation Layer).
- Delivery function area (Delivery Layer).
- Signalling or control function area (Signalling or Control Layer).

First, the following operations take place in the Encapsulation Layer: media packetization, media fragmentation, media synchronization (insertion of timestamps), media multiplexing, content protection, insertion of Composition Information,[6] and formatted content available for storage and delivery. The result of the Encapsulation Layer is an MMT package [17].

Second, the operations in the Delivery Layer include: network packetization, network flow multiplexing, insertion of delivery timestamps, QoS operations, and

[6] '*This includes spatial and temporal location of media objects in a given scene*' [17].

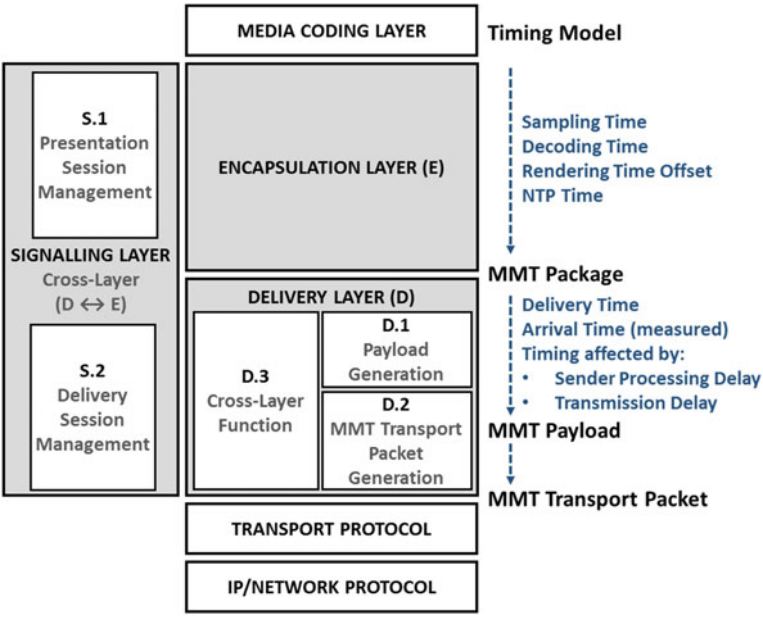


Fig. 15.19 MMT architecture, layers, and functionalities [17] with timing model proposed in [13]

error handling. It is segmented into three main areas: D.1 generating the MMT payload, D.2 generating the MMT transport packet, while D.3 provides information, such as flow control (covering also the Transport Protocol Layer), session management, session monitoring, and error control for cross-layer optimization (related to QoS) [17]. Finally, in the Signalling Layer the presentation session, S.1, and the delivery session, S.2, are managed.

15.7.2 Content Model for MMT

MMT shares with the previous MPEG standards a focus on '*representing structural relationships of Elementary Streams*' and on '*carrying information for synchronized playback of multimedia*' [16]. Furthermore, MMT's logical content model concentrates on providing all information for the Delivery Layer processing [16].

The elements of the MMT content model are: MMT Assets, Media Processing Units (MPU), Media Fragment Units (MFU), Composition Information (CI), and, finally, Asset Delivery Characteristics (ADC). An MMT package is formed by MMT Assets, CI, and ADC.

'*An MMT Asset defines the logical structure carrying coded media data*' [16]. It may contain both timed and non-timed data. An example of MMT Asset is an MP2T stream, an MP4 file, or a JPEG file [16]. An asset is made of a group of

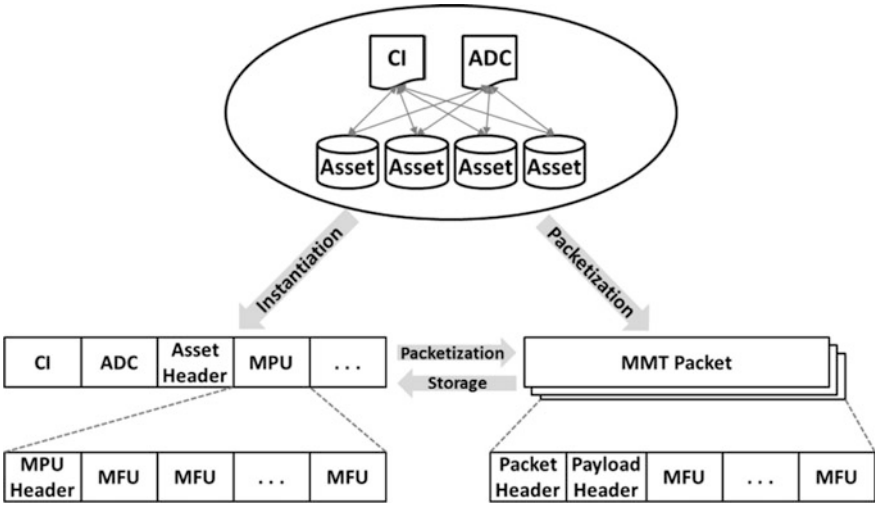


Fig. 15.20 Relationship of an MMT package's storage and packetized delivery formats [15, 16]

MPUs (one or multiple AUs), and each MPU is formed by one or multiple MFUs. '*MFUs carry small fragments of coded media data for which such fragments can be independently decoded or discarded*' [16]. The structure of an MMT packet can be seen in Fig. 15.20.

The structure of MPUs and MFUs is the common data unit for MMT storage and delivery. An MMT package is serialized/instantiated into an MMT file, whereas an MMT package is packetized into an MMT packet [16]. The structure of an MMT file is found in the left part of Fig. 15.20 and an MMT packet at the right part of Fig. 15.20.

Within an MMT packet, the MMT payload has the advantage of being delivered via RTP or MMT protocol, which can also deliver MMT signalling messages [16].

Information for media delivery and media play-out is provided by MMT signalling messages. Related to time delivery via MMT, there is an MMT signalling message called Clock Relation Information (CRI) which '*provides the relationship between the MPEG-2 TS and UTC system time clock for synchronized presentation in a hybrid delivery*' [16].

15.7.3 Timing Model for MMT

A new timing model for MMT is proposed in [13]. This MMT timing model is based on the timestamping in the MMT Encapsulation Layer. '*The proposed time stamping service provides for sender/receiver timing matching for synchronization of several media sources*' [13].

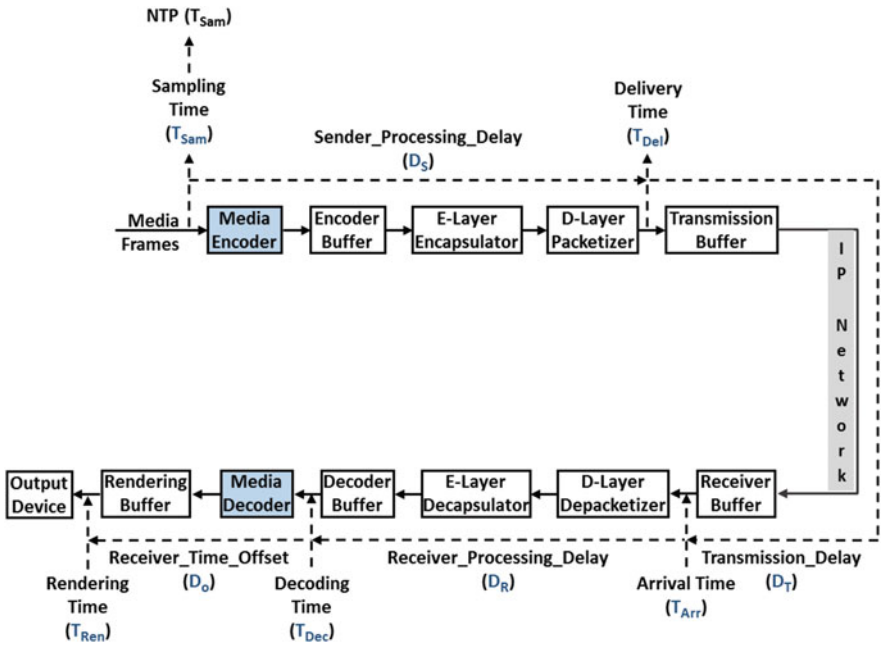


Fig. 15.21 MMT model diagram at MMT sender and receiver sides. Figures 15.3 and 15.4 from [13]

The timing model proposes to define *sampling_time*, *decoding_time,* and *rendering_time* at the Encapsulation Layer. Meanwhile, both the *delivery_time* and *arrival_time* are outlined at the Delivery Layer because they are impacted by the *sender_processing_delay* and the *transmission_delay* [13]. The location of the time information within the MTT architecture can be found in Fig. 15.19.

Sampling_time is the sampling time of the first AU within an MMT packet. The *delivery_time* is the time at which the package is ready to be transmitted. The *arrival_time* is the MMT packet arrival time at the receiver (affected by network transmission delay). *Decoding_time* is the decoding time of the compressed media stream conveyed in the MMT packets. *Rendering_time* is the rendering time of the decoded media to the output devices. In Fig. 15.21, the complete MMT timing model can be seen [13].

The MMT timing model contemplates support for MP2T streams. Therefore, it considers that *sampling_time* should have the same clock resolution as MP2T DTS and PTS timestamps values, 90 kHz, and also the same bit size (33-bit) [13]. See Fig. 15.22.

The MMT timing model also contemplates linking to RTP timestamp protocol, by mapping the *sampling_time* to the RTP timestamp. Considering that the RTP timestamp in the RTP header is a 32-bit field, MMT timing model proposes to have

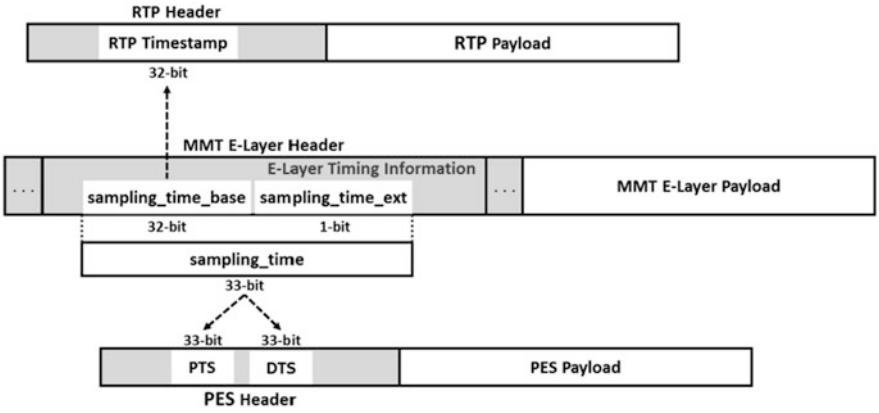


Fig. 15.22 Links between *sampling_time* and RTP and MP2T timestamps based on MMT timing model proposed in [13]

two fields to convey the MMT *sampling_time* (33-bit), *sampling_time_base* (32-bit), and *sampling_time_ext* (1-bit). Therefore, the *sampling_time_base* can be mapped directly into the RTP timestamp [13]. See Fig. 15.22.

15.8 Conclusion

To accomplish the synchronized play-out of multimedia contents, different factors need to be taken into account, including the delivery platform, the media delivery protocols, and the media standards in use. The MPEG and DVB are two groups of the main media standards, although many other can be used.

In this chapter, the timelines, clock references, and timestamps, in the most common MPEG standards (MPEG-2 Transport Streams, MP2T, and MPEG-4) used to provide synchronized media play-out, have been described. The time-related fields and issues in MPEG-DASH and MMT have also been explained. Additionally, the DVB standard (specially used in broadcast media delivery) tools to deliver time-related information to receivers have also been presented.

Definitions

MPEG Timeline Time information, clock references and timestamps, to reproduce, time-align encoder/source media clock to the decoder/receiver media clock.

Timestamps A timestamp is used to agree on a specific moment in time, such as decoding, composing or playing time of an Access Unit (AU) or Media Access Unit (MDU).

Clock References They are the mechanism to recreate encoder's clock frequency at the decoder to guarantee the correct media stream play-out, the means used by MPEG standards to reproduce encoder's clock rate at the decoder.

References

1. Information Technology. Generic Coding of Moving Pictures and Associated Audio: Systems Recommendation H.222 (2000E), ISO/IEC 13818-1, Dec 2010
2. Chen, X.: Transporting Compressed Digital Video, 1st edn. Kluwer Academic Publishers (2002)
3. Information Technology Generic Coding of Moving Pictures and Associated Audio Information. Amendment 1: Delivery of timeline for external data. Recommendation ITU-T H.222 (2014) Amendment 1, Apr 2015
4. The Moving Picture Experts Group. http://mpeg.chiariglione.org. Accessed 26 Mar 2017
5. Herpel, C.: Elementary stream management in MPEG-4. IEEE Trans. Circuits Syst. Video Technol. **9**(2), 315–324 (1999)
6. Beloqui, L., Boronat, F., Montagut, M., Melvin, H.: Understanding timelines within MPEG standards. IEEE Commun. Surv. Tutorials (2016)
7. Information Technology. Generic Coding of Audio-Visual Objects. Part 1: Systems (2010E), ISO/IEC 14496-1, June 2010
8. Information Technology. Dynamic Adaptive Streaming over HTTP (DASH). Part 1: Media Presentation Description and Segment Formats. ISO/IEC 23009-1:2012, Apr 2012
9. ETSI EN 300 468 v1.14.1 (2014-01). Specification for Service Information (SI) in DVB Systems. Digital Video Broadcasting (DVB), Jan 2014
10. ETSI TR 101 211 v1.11.2. Guidelines on Implementation and Usage of Service Information (SI). Technical Report. European Telecommunications. Digital Video Broadcasting (DVB), May 2012
11. Julian Date Calendar/Calculator Day-of-Year (JDay): National Aeronautics and Space Administration, 05 May 2016. https://www-air.larc.nasa.gov/tools/jday.htm. Accessed 19 Mar 2017
12. Information Technology. High Efficiency Coding and Media Delivery in Heterogeneous Environments. Part 1: MPEG Media Transport (MMT), ISO/IEC 23008-1:2014, June 2014
13. Kwang-deok, S., Tae-jun, J., Jeonglu, Y., Ki, K.C., Jinwoo, H.: A new timing model design for MPEG media transport (MMT). In: Proceedings of IEEE International Symposium on BMSB, pp. 1–5, June 2012
14. Aoki, S., Otsuki, K., Hamada, H.: Effective usage of MMT in broadcasting systems. In: Proceedings of IEEE International Symposium on BMSB, pp. 1–6, June 2013
15. Lim, Y.: MMT, new alternative to MPEG-2 TS and RTP. In: Proceedings of IEEE International Symposium on BMSB, pp. 1–5, June 2013
16. Youngkwon, L., Kyungmo, P., Jin Young, L., Aoki, S., Fernando, G.: MMT: an emerging MPEG standard for multimedia delivery over the internet. IEEE MultiMedia **20**(1), 80–85 (2013)
17. Fernando, G.: MMT: the next-generation media transport standard. STE Commun. **10**(2), 45–48 (2012)

Chapter 16
Synchronization in MPEG-4 Systems

Jean Le Feuvre and Cyril Concolato

Abstract The MPEG-4 standard was defined in the early days of broadband Internet, after successful deployments of digital television networks, with the goal of unifying both broadcast and broadband media architectures and protocols in a single standard, tackling natural media (audio, video, images) as well as synthetic 2D or 3D graphics and audio. As such MPEG-4 can be seen as one of the first attempt at building the so-called convergence of Web and TV. Some parts of the standard have changed the media world forever (AAC audio and AVC|H264 video compression, MP4 file format), and while other parts have not always met their markets in successful way, they paved the way for more recent works, including HTML5 media. In this chapter, we explain how the MPEG-4 standard manages playback and synchronization of audio-visual streams and graphics animations and how multiple timelines can be used to provide rich interactive presentation over broadband and broadcast.

Keywords Interactivity · Synchronization · Graphics · Media Broadcast

16.1 Introduction

The MPEG-4 standard (ISO/IEC 14496), released by the Moving Picture Experts Group (MPEG), defines a large standard covering interactive applications, audio and video compression, advanced 3D graphics, content protection, file formats, fonts and text delivery [1]. Its application scope ranges from simple linear playback to full interactive audio-visual applications. As a consequence, MPEG-4 content can exercise various synchronization features. For instance, multiple media content

J. Le Feuvre (✉) · C. Concolato
Telecom ParisTech – LTCI, 46 rue Barrault, 75013 Paris, France
e-mail: jean.lefeuvre@telecom-paristech.fr

C. Concolato
e-mail: cyril.concolato@telecom-paristech.fr

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_16

may be presented at the same time and hard synchronization constraints may be applied. Alternatively, media content may be presented as a result of an interaction. In this case, the synchronization of this media content with already started media content may be loose. This differs substantially from its predecessor, MPEG-2 (ISO/IEC 13818) [2], which focused on storage and broadcast delivery of linear audio-visual content such as TV channels. Due to this ability to describe various synchronization options, MPEG-4 introduced new synchronization-related tools that are described in this chapter. These tools are specified in the MPEG-4 Systems part (14496-1).

In MPEG-4 terminology, a multimedia presentation is organized by a scene. An MPEG-4 scene consists of any number of media primitives, called objects, classified by type: visual (image or video), audio (natural or synthesized on the player side), text (subtitle, ticker), font and animations (2D or 3D graphics). In order to organize in space and in time these objects and to enable the user to interact with them, MPEG-4 defines languages to describe a scene: BIFS for "Binary Format for Scene," specified in ISO/IEC 14496-11, and LASeR for "Lightweight Application Scene Representation," specified in ISO/IEC 14496-20. The required information for configuring, synchronizing, and decoding the media content associated with scene objects is delivered by the Object Descriptor Framework, specified in ISO/IEC 14496-1. The standard also defines an authoring format for the BIFS language called the eXtensible MPEG-4 textual representation (XMT), using two possible XML syntaxes: XMT-A, very close to the binary scene structure and compatible with the X3D format of the Web3D consortium, and XMT-O, a higher-level representation of the scene based on the SMIL language [4] from W3C.

In this chapter, we present how the Object Descriptor Framework can be used to describe various synchronization aspects of interactive media content, in particular its so-called Sync Layer. We will also present how the MPEG-4 scene descriptions leverage these aspects when proposing the synchronized playback of multiple media objects. We illustrate the different tools using an example multimedia presentation featuring:

- Audio and video media.
- Graphical shapes (rectangles, circles …) overlaid on the video media to highlight region of interests; clicking on a region opens a display area with more information such as name of the selected player for a sport event or Web site and reference of the selected item for a commercial.
- Interaction panel allowing navigation on the media (pause/resume, seeking, and speed adjustments).
- Image used as background for the control panel.

16.2 Describing Synchronization of Media Objects

The part of the MPEG-4 Systems standard which covers synchronization is called the Sync Layer. It is a part of a larger aspect of the standard called the Object Descriptor Framework. This latter is introduced in the next section, while the former will be detailed afterward.

16.2.1 MPEG-4 Object Descriptor Framework

The MPEG-4 Object Descriptor Framework provides the tools needed to describe the media objects used in a scene. Each media object in a presentation is declared by an Object Descriptor (OD). This descriptor carries various descriptions associated with an identifier (OD_ID), unique within a presentation. The descriptions provide the information required by the MPEG-4 player to determine if it can process the media content (e.g., it has a compatible decoder), to configure its network and demultiplexing stacks (e.g., source address, protocol options such as packet loss detection, number and identification of multiplexed streams), to instantiate an associated decoding pipeline, possibly including a decryption part, and to synchronize the media content with other media. The Object Descriptor does not indicate how the media content will be integrated in the presentation. This information is provided by the scene description. In particular, the Object Descriptor does not indicate when the media content will be used. For example, a media may be used only during a given time interval within the presentation, or only after a user interaction. The Object Descriptor does not indicate either how the media content will be rendered, i.e., positioned on the screen for visual media or mixed in final audio output for aural media. The scene describes these aspects and uses the OD_ID to indicate to the player which media object is used. This separation between how a media object is obtained/decoded/synchronized and how it is used in the presentation is a core concept of MPEG-4 presentations, inherited from MPEG-2. The intent of the standard is to allow the design of scenes independent from the media characteristics (network access, compression format…). As an example, the compression format of the media object used in a scene is not known to that scene. This allows either providing the same media in two different formats to adapt for different player capabilities, or re-encoding the media, without changing the scene. Similarly, a media object may be composed of several layers or may be modified to add new layers (in a scalable media context where each layer adds refinement to the lower layer) without the scene having to know about these layers.

In the Object Descriptor Framework, an object is described as a collection of elementary streams, which carry media data. Elementary streams are described by an elementary stream descriptor (ESD), which contain a unique identifier called ES_ID along with decoder configuration (type, bitrate, maximum buffer size, etc.),

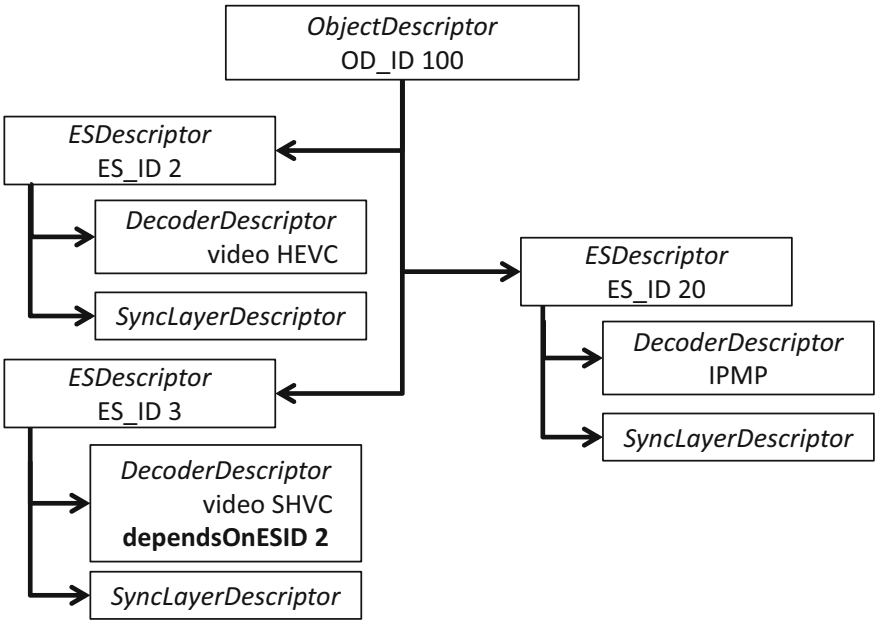


Fig. 16.1 Example object descriptor for a video object composed of two elementary media streams and one associated stream

language information, network settings, and other information. Each elementary stream can only carry a single type of data (visual, audio, font, text, 3D mesh animation, BIFS or LASeR, etc.). An Object Descriptor also cannot mix streams of different types. For example, an object cannot be composed of both video and audio. In this case, two objects have to be defined. Several streams of the same type can be described in a single Object Descriptor, in order to address various use cases: streams with different languages, streams with different compression schemes, or streams with different bandwidth requirements. The player is in charge of determining which stream fits its usage conditions and will select only one of these streams for playback at a given time. A player may also select several streams in one Object Descriptor in case these streams are dependent upon each other, as shown in Fig. 16.1, where the stream with ES_ID 3 is a hierarchical enhancement layer of the stream with ES_ID 2, indicated by means of the *dependsOnESID* field. This typically happens when the media is using hierarchical coding schemes (e.g., using temporal or resolution scalability), in which case the player will select the stream carrying the desired quality and all the required streams carrying lower quality. The description of the hierarchy of streams is provided in the ESD of a stream indicating the ES_ID of the stream depended on.

Other kinds of elementary streams may be present in any quantity in each object descriptor, to provide rights management data for different protection systems (as shown in Fig. 16.1), or content description in various languages.

Finally, in order to allow for dynamic modifications of the set of resources used in a scene, the Object Descriptor Framework defines a new elementary stream type, specific to MPEG-4 presentation: the Object Descriptor Stream. This stream is made of a sequence of Object Descriptor commands used to modify, insert or remove Object Descriptors. The set of commands that should be applied at a given time are packed into one delivery unit. This feature allows delivering updates of the media object description, similarly to the use of Program Map Table updates in MPEG-2.

Our multimedia presentation example will typically have a main object descriptor, called *InitialObjectDescriptor*, referencing one BIFS stream carrying the presentation and one OD stream used to deliver the objects descriptors for that scene. These objects will be at least one for the video, one for the audio and one for the panel background image. For the purpose of illustrating this chapter, we assume that the dynamic position and size of each graphical shape is delivered through a dedicated BIFS stream, hereafter called BIFS ROI stream, and therefore, an object for that BIFS stream is also declared (we will see later in this chapter that there are other possible methods). Since that BIFS streams carries information modifying the main scene, its successful decoding depends on the proper decoding of the main scene: This is indicating by setting the *dependsOnESID* field of the BIFS ROI stream to the value of the main scene BIFS stream ID.

16.2.2 Synchronization Signaling: The Sync Layer

Regarding synchronization, the most important descriptor of the Object Descriptor Framework is the Sync Layer Descriptor, present at the elementary stream descriptor level. It provides information enabling intra-stream and inter-stream synchronization. Its features are described here.

16.2.2.1 Intra-stream Synchronization: Timestamps and Clocks

In MPEG-4, as in most audio-visual delivery system, stream data is divided into blocks called Access Unit (AU). Each AU contains the minimal set of data to which a unique presentation time can be assigned. Typically, an AU corresponds to one coded video frame, one audio coded frame, one set of Object Descriptor commands or one set of BIFS or LASeR at a given time. Each AU is assigned a timestamp delivered by the underlying transport protocol, which expresses the time at which the media has to be presented to the end user. In MPEG-4, this timestamp is called the Composition Time Stamp (CTS). Additionally, an AU may have a Decoding Time Stamp (DTS) assigned, indicating that the AU has to be decoded at a given time but not presented until its CTS is due. This is mainly used for video streams using bidirectional prediction. In the MPEG-4 systems theoretical decoder model,

as in many other multimedia standards, the decoding is considered instantaneous, i.e., the decode duration is 0.[1]

In MPEG-4 Systems, both CTS and DTS are represented as integer numbers, expressed in a resolution called timestamp resolution configured in the Sync Layer Descriptor. For example, a CTS value of 1000 with a timestamp resolution of 1000 would mean 1 s, while the same CTS in a resolution of 1000000 would mean one microsecond. The timestamp resolution is usually chosen based on the frequency of the media data for that stream, e.g., using the sample rate for an audio stream or a multiple of the frame rate for a video stream. This avoids accuracy loss due to the necessary rounding associated with the integer representation. For example, taking a stereo stream sampled at 48000 Hz with 1024 audio samples per AU, the CTS between two AU would be incremented by 1024/48000 = 0.02133333… s. If a timestamp resolution of 1000 is picked, this would mean that the integer increment would correspond to 21 ms for the first and second AU then 22 ms for the third one to adjust the timing, resulting in loss of precision in the timing. Choosing a resolution of 48000 will give 1024 increments between each AU and avoid this precision loss. For visual media (video, animations), the timestamp resolution is typically chosen to be a multiple of the media frame rate (e.g., 25000 for 25 fps, 30000 for 29.97 or 30 fps) while for audio media a multiple of the audio sample rate is usually chosen.

During the processing of a single stream by a media player, in order to properly restitute the delays between the frames (referred to as "intra-stream synchronization"), the CTS of an AU is compared to a clock. In MPEG-4 Systems, this clock is associated with the Object declaring this stream and is called the Object Time Base (OTB), and all incoming AUs are presented when their composition time matches the OTB. The time on this clock corresponding to the instant at which the first AU processed is not necessarily zero. The clock may be initialized with a nonzero value. This case can happen when a user joins a streaming session which started moments ago. The (initial) clock value can be carried within a special stream, called Object Clock Reference (OCR) stream, within the same object or within a different object. To avoid potential drift between the sender's clock and the player's clock, the OCR stream carries updated values on a regular basis (e.g., several times per seconds). This regular delivery of the server clock information is similar to the use of the Program Clock Reference (PCR) in MPEG-2 Systems. Figure 16.2 shows a typical setup for a video stream, using timestamps expressed in 90 kHz timescale but OCR timestamps expressed in 27 MHz. Upon reception of a value on the OCR stream, the OTB is adjusted to the indicated value, and then increases at the pace of the client local clock, until a new value arrives.

The OTB may also be initialized to a value given out-of-band (i.e., outside of the MPEG-4 Systems data, without relying on OCR streams). For example, the initial value of the OTB is typically set to the presentation time of the first access unit,

[1]In practice, it is usually considered bounded by a value less than or equal to the shortest duration of any media frame for that media profile, in order to guarantee real-time decoding.

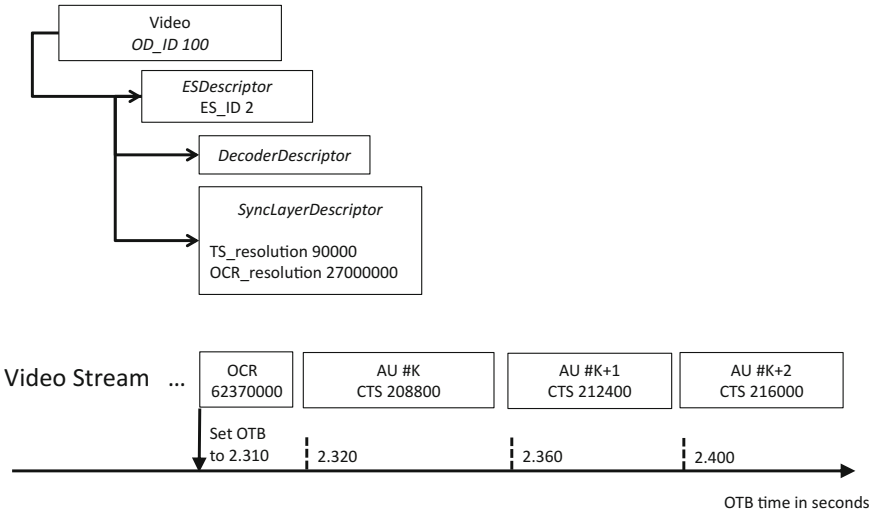


Fig. 16.2 Example sync layer setup

which is usually 0, when playing an ISOBMFF file. When playing a stream of an RTSP [3] streaming session, the initial value of the OTB is defined to be the RTP timestamp indicated by the server during the RTSP setup.

16.2.2.2 Inter-stream Synchronization: Clock References

MPEG-4 introduces the ability to embed in a presentation media streams that are meant to be played together in a synchronous manner, but also streams that are not meant to be played together and therefore do not have synchronization constraints. This is the case for instance, when a media stream is started interactively, while another media stream is already playing: In that case, the content author does not want to pause the already playing stream until the newly started one reaches the same time, as would be the case if all streams were synchronized.

For the purpose of grouping streams that must be played synchronously, the Object Descriptor Framework allows indicating that a stream uses the OTB of another stream. This is achieved by setting the OCR_ES_ID field, in the stream to be synchronized, to the ES_ID of the stream the OTB is inherited from. When an OCR_ES_ID is provided for a stream, playback operations on that stream also apply to all streams that depend on that same OTB. For example, pausing one stream pauses all the other streams. This is quite convenient from a scene description perspective, since only one media has to be controlled. This is illustrated in Fig. 16.3, where the audio stream uses the video stream as its OTB: The audio and video AUs are presented when the OTB time matches their associated CTS, achieving fine synchronization between the two streams; when the CTS of both streams match (as is the case for Video AU K + 1 and Audio AU N + 2),

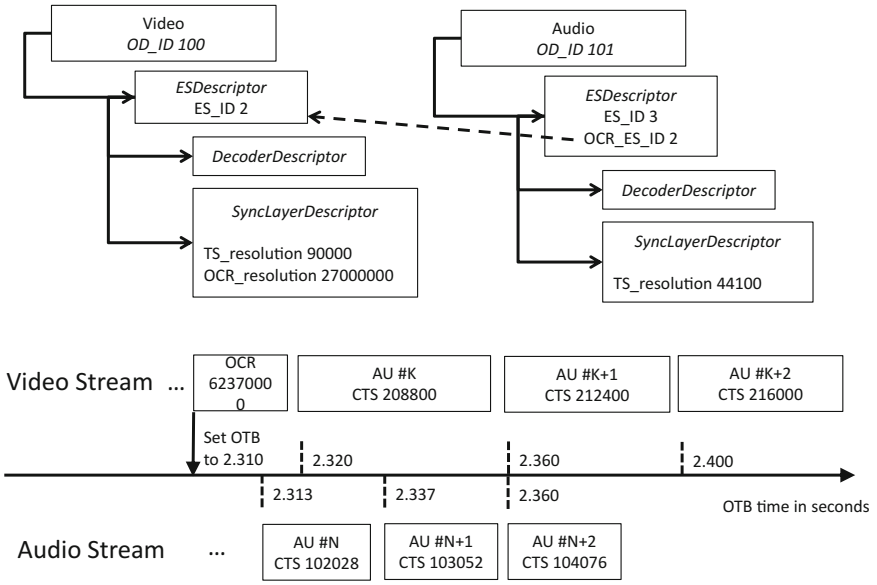


Fig. 16.3 Example of audio-video sync layer setup

frame-accurate synchronization is achieved. In MPEG-4 applications, there are cases where an object describes several streams, but where only one stream is played at a time (e.g., when an object descriptor provides alternate encodings of the same media such as different bitrates or codecs, or alternate languages of the same audio object). This makes it difficult to indicate one of the streams as a clock reference. Only one stream is processed at a time, and the actual stream may change depending on user interaction (e.g., changing language). In such case, it would be suboptimal to set the OCR_ES_ID of those streams to one of them. This would require processing of that stream solely for the purpose of controlling the clock, even when it is not used in the scene. For such cases, the MPEG-4 Systems allow defining objects containing a stream dedicated to the carriage of clock references and no other data. Such streams are called OCR-only streams and are used to carry the object time base for a set of streams regardless of their playback state. In that case, the playback control is performed on this object without the scene author having to understand which actual media stream is selected.

If no OCR_ES_ID is set on a given stream, the stream is assumed to be independent, from a synchronization point of view, from any other stream. In other words, an AU of a stream S1 with a CTS value of X has to be presented with the AU of another stream S2 that has CTS of X, only if they share the same OTB. If this is not the case, even if they have the same value of X, there is no guarantee that they will be processed at the same time. This allows playback operations (pause, resume, seeking, reverse playback, or fast forward) on one stream without impacting the playback of other streams.

The clock reference assigned to a given stream cannot be changed interactively. It may however be modified by the server, by updating the stream descriptor through a command in the OD stream, replacing the value of the OCR_ES_ID for that stream to use the new desired clock reference. For cases where synchronization rule adjustments between streams cannot be predicted at authoring time nor modified by the server but have to be evaluated during playback, the FlexTime model, described later in this chapter, has to be used.

It should be noted that even if streams are signaled as being synchronized, the standard leaves some liberty to implementations as to how hard the synchronization can be, in particular how variable latencies on different synchronized streams are to be dealt with. For instance, an implementation may decide to pause the processing of a complete set of synchronized streams, when one of them is late. Another implementation may decide to keep playing the other streams and seek forward in the late stream when new data arrives, skipping the late AUs.

In our multimedia presentation example, audio, video, and the BIFS ROI stream are meant to be displayed synchronously. A typical setup would indicate that the video stream is self-synchronized (using video OCR_ES_ID equal to video ES_ID) and that the audio and BIFS ROI streams use the video stream as a clock reference. The other streams (main BIFS, OD, and image) are also to be synchronized together but on a different timeline. We will indicate that the main BIFS stream acts as its own clock reference, and the OD and image streams will use that stream as their clock reference.

16.3 Time and Synchronization in Scene Descriptions

As seen in the previous sections, the MPEG-4 Systems standard provides tools within the Object Descriptor Framework to describe how the media streams are processed from a synchronization point of view. The actual use of these media streams, in particular with regard to timing, is provided by the Scene Description streams, such as MPEG-4 BIFS or MPEG-4 LASeR, described in the following sections.

16.3.1 Timing in MPEG-4 BIFS

MPEG-4 BIFS, or Binary Format for Scene, defines a language in textual and binary formats for the representation of spatiotemporal relationships between media objects in 2D and 3D spaces. The standard (ISO/IEC 14496-11) derives from the Virtual Reality Modeling Language (VRML) standard (ISO/IEC 14772-1) and extends it in various aspects: 2D graphics and associated effects, advanced 3D graphics such as face and body animations, various physical effects, data framing tools for animations and complex scene delivery, and more importantly in terms of

synchronization by the integration with the Object Description Framework, to drive the animation and media playback, and by the definition of advanced timeline or media control primitives. This section focuses on these latter aspects.

16.3.1.1 Time in Animations

An MPEG-4 presentation may contain animations. Animations can be as simple as the translation of a graphical element on the screen or as complex as a fade-in/fade-out operation between two videos. These animations can be described in a BIFS scene in a unified way, with a single timing model. The timing model relies on OTB, as described by the Object Descriptor Framework, and times, given in the scene itself. The animation times used in a BIFS scene (i.e., start times, end times) are expressed with respect to a clock that is unique to the scene. This clock starts when the BIFS AU carrying the scene is processed (i.e., when the CTS of the BIFS AU is reached on the OTB associated with the BIFS stream). This clock runs at the same pace as the OTB.

When the scene OTB is not adjusted by OCR information, the scene time is simply the time elapsed on the system clock since the initialization of the scene OTB. However, when the time is adjusted by OCR information, the time in seconds elapsed since the beginning of the scene is defined as the difference between the current OCR timestamp and the OCR timestamp when the BIFS AU was processed divided by the OCR timestamp resolution. Since OCR timestamps are sent at a lower frequency than the animation frame rates, implementations have to be able to estimate the OTB between the last received OCR and the next one to arrive. This is usually done by estimating time elapsed on the system clock since the last received OCR.

The animation times in a BIFS scene are provided by Timer nodes derived from VRML. Timer nodes are active during a time interval given by two fields: *startTime* and *stopTime*, whose values refer to the scene time. A timer is active when the current scene time is in the interval [*startTime, stopTime*], otherwise it is inactive; *stopTime* is ignored if less than *startTime*, in which case the timer is active once the scene time becomes greater than *startTime*. These fields can be updated through interaction within the scene; however, setting the *startTime* field to an active timer is ignored (i.e., the time at which a timer started cannot be modified, the timer has to be deactivated first in order to evaluate the new *startTime* value).

In the MPEG-4 BIFS animation model, active timers generate events, which can be used to compute a corresponding animated value, such as a translation value or a fade transition coefficient. The *time* event carries the current scene time in seconds, as described above. The *fraction_changed* event indicates the time position relative to the timer's duration. For example, if an animation describes a 2 s translation, the *fraction_changed* event will carry the value 0.25 when 0.5 s are elapsed in the animation, 0.5 for 1 s and 1.0 for 2 s. The number of *fraction_changed* events is implementation specific, but usually matches the rendering frame rate. For example, a timer running for 2 s will emit 50 *fraction_changed* events when the scene is

rendered at 25 frames per second (fps) or 200 when the scene is rendered at 100 fps. The number of *fraction_changed* events is independent from the duration of the animation, only the value of the *fraction_changed* is. Timers may be setup to loop, and will generate a *cycleTime* event at the end of each cycle, carrying the scene time at that instant.

To pause the effect of a timer, this timer must be made inactive by setting its *stopTime* field to any value less than or equal to the current scene time, and greater than the *startTime*. Since timers generate *fraction_changed* event values based on their *startTime* value, relative to the start of the scene, the pause behavior (i.e., the generation of *fraction_changed* events carrying always the same value) is achieved by continuously setting the *startTime* to the original *startTime* plus the time elapsed during the pause period.

There are various ways of building animations using timers. Simple animations in BIFS are usually done using timers and interpolation techniques. The *fraction_changed* event of the timer is transmitted to an interpolator, in charge of converting the [0–1] value of the event into 2D or 3D coordinates, or an angle, a color, etc. Interpolators are of two kinds in BIFS:

- Linear interpolators: for N fraction keys (K_i) in [0,1] and N associated values (V_i), the output value for a fraction event f in [K_i, K_{i+1}] is

$$V_f = (V_{i+1} - V_i)*(f - K_i)/(K_i - K_i) + V_i$$

- Animators defined as extension to interpolators and similar to the animation elements of SMIL [4], and animators can specify linear, quadratic, cubic, or Non-Uniform Rational Basis Splines (NURBS) animation path and various ways of defining the time intervals between each value change:
  - Discrete: Value is constant until next value should be output.
  - Linear: As with regular interpolators.
  - Paced: Value changes are equidistant in time, based on the length of the animation path.
  - Spline: Input fraction is remapped to another input fraction through a cubic Bezier spline.

In our multimedia presentation example, this timer-based technique will be used to animate the media control panel (show/hide, animate buttons) or animate the display area for the ROI information. The timers will be triggered upon interactions by the user, and the animations will be executed regardless of the synchronization state of the audio, video, and ROI streams. This technique allows defining most of the temporal behavior of the user interface independently from the media streams used in the presentation.

Another approach for animations with timers is the frame-based approach. It is used most of the time when animations are too complex to be described only by interpolation rules. This is the case for animating subtitles or advanced geometrical shapes (2D cartoons, 3D meshes animations). In this case, the different frames of

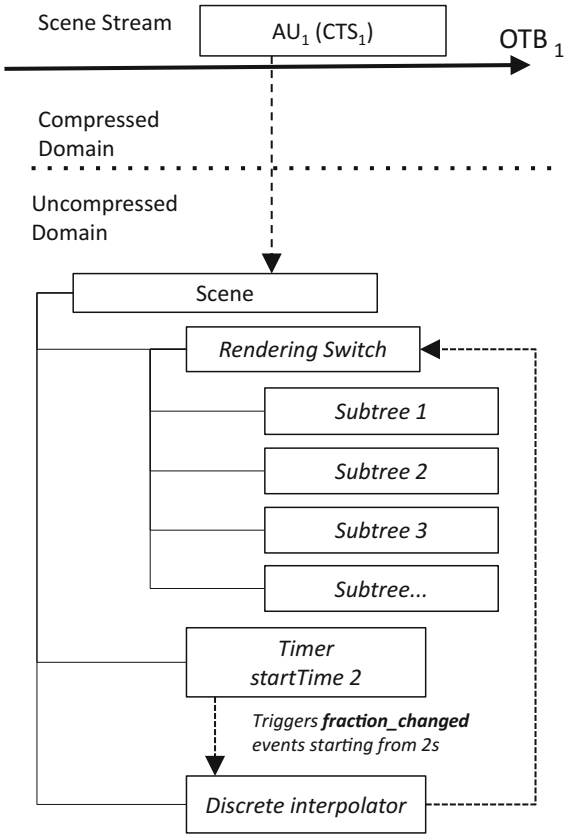


Fig. 16.4 Timer-based BIFS animations

the animation are all described in the main scene but are not visible. The actual frame to be displayed is then selected based on the scene time. This is illustrated in Fig. 16.4: The timer starts *startTime* seconds after the CTS of the scene, and the timer fraction value is interpolated into an integer ranging from first frame to last frame of the animation, used to trigger visibility of each frame. This technique works fine when the frame rate of the animation is constant, but requires more tuning for non-constant frame rate animations.

One of the drawbacks of this technique is that each frame is fully loaded in memory at the player side from the scene start, which can be quite expensive. BIFS provides a more memory efficient alternative to this approach. Each frame is binary encoded as a BIFS *command,* and the command is only executed at a specific instant. This instant can be provided using two approaches. In the first approach, the command is executed when a specific event is triggered. This event can be the result of the user interacting with the scene (e.g., click on a button). The event can also be the start of a timer. This event-based approach has the advantage of storing the compressed animation frames in memory, and having only the active frame fully loaded in memory.

In our multimedia presentation example, animating the ROI using this technique would imply having each of its AUs embedded in the main BIFS scene, and monitoring the playback of the video stream to trigger the right command. Such logic is quite heavy to author (multiple events routing and BIFS command storages have to be described) and will require a new scene whenever some animation data changes, since all animation data is in the BIFS scene. For complex, live, or media-related animations, this technique is usually rejected in favor of the second approach relying on Animation streams.

Animation streams in BIFS are additional elementary streams carrying BIFS commands and depending on a main BIFS stream for decoding. Animation streams typically carry commands modifying parts of the scene that are relatively independent from the rest of the scene, usually in such a way that the modifications do not impact the logic of the scene. Animation streams and the associated main BIFS stream may have different synchronization settings. It is possible to indicate that these streams are not synchronized, by explicitly indicating different OTBs for the main stream and the animation stream(s). As indicated previously, this is achieved by indicating different OCR_ES_ID in the elementary stream descriptor for these streams. In this case, the playback of the animation may be controlled (paused/resumed/stopped) without having any impact on the rest of the scene. However in such case, the animation streams may drift from the main scene. This mode is useful for looping animations, such as waiting animations. Figure 16.5 illustrates how the

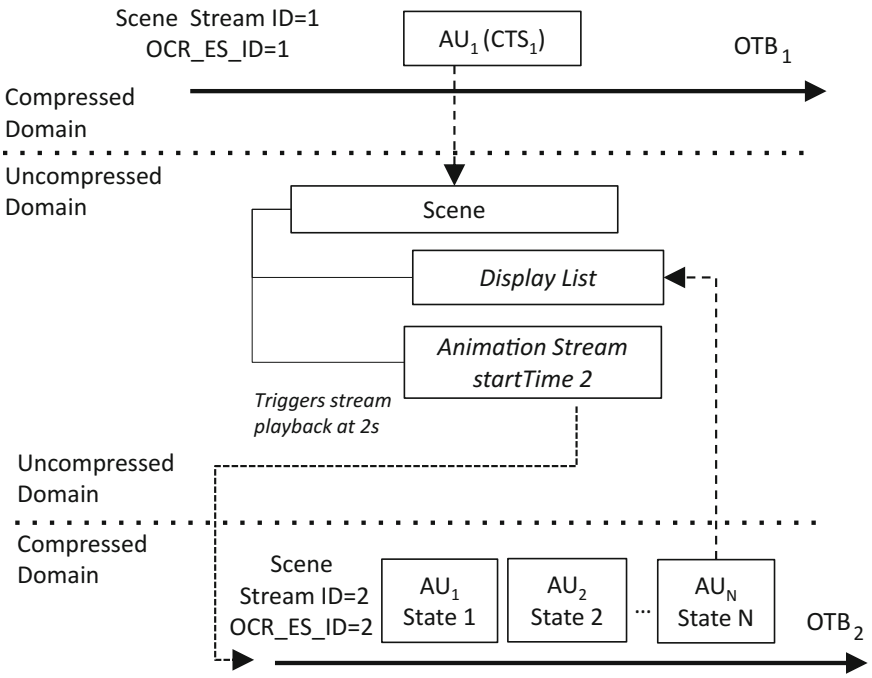


Fig. 16.5 Stream-based BIFS animations

setup of Fig. 16.4 can be achieved with stream-based animation: an *AnimationStream* node triggers, *startTime* seconds after the CTS of the scene, the playback of a secondary self-synchronized BIFS stream whose AUs directly modify a subpart of the scene graph. In this setup, only the current scene state of the animation is decompressed, the other states are compressed AUs of the animation stream. Animation streams playback can be controlled within the scene using a node called *AnimationStream*. This node indicates the OD_ID of the object carrying the animation stream(s) and is capable of receiving events to start and stop the stream and control speed as well as looping versus single playback. This is the technique we chose to deliver the ROI data in our multimedia presentation example.

It is also possible to indicate that these streams are synchronized, by explicitly indicating a single OTB for all streams. This is achieved by indicating the same OCR_ES_ID in the associated elementary stream descriptors. When all streams are synchronized, it is no longer possible to control the playback of the animation streams from inside the scene, since that would pause the main scene timeline as well. That scene can still be paused by means outside of the scene, such as the player user interface. This mode is useful for scenes without local interactivity, such as cartoons or pure animations.

16.3.1.2 Time in Media Handling

Media synchronization in BIFS is very similar to the handling of animations as seen previously. The same principles are found. Media nodes (*AudioSource*, *MovieTexture*, *AudioClip*) used to link a set of streams with a target rendering operation (i.e., audio mixing or video/image texturing) and provide simple playback control (play/pause/stop). OTB indications are used to indicate if the media are synchronized to the main scene or not.

A typical audio-visual scene in BIFS will have one BIFS and one Object Descriptor streams, one Object Descriptor for audio and one Object Descriptor for video. The audio and video streams, if they are synchronized, will share the same OTB, indicated through OCR_ES_ID of the video and/or audio streams. This implies that the composition timestamps of all audio and video AUs will be compared with the same OTB in order to properly synchronize the playback of the streams.

The BIFS scene and the audio/video streams may share the same OTB, in which case no playback control of the video can be done within the scene. This is used for most common use cases where animations have to be synchronized with the audio and video, such as subtitles, annotations, or logos, but without any need for playback control. Alternatively, they can have different OTBs, in which case the video can be started/stopped and paused/resumed from within the scene. This is typically used for video portals, interactive video navigations, etc. The control will be done using one of the media nodes on only one of the objects to control, audio, or video. The standard mandates implementations to automatically apply the same

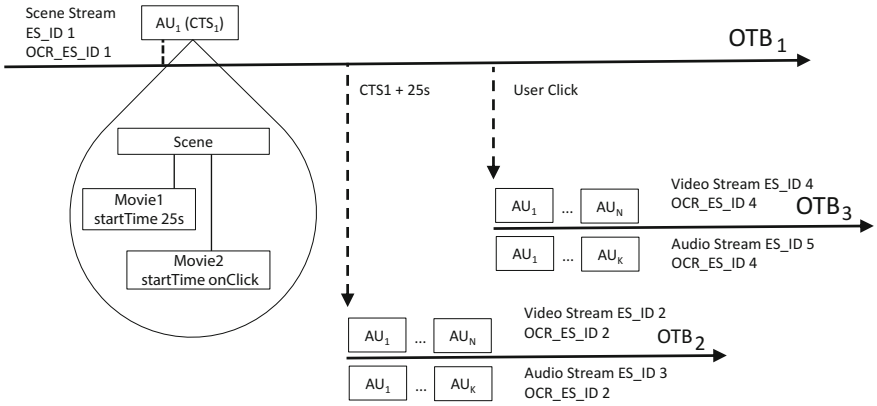


Fig. 16.6 Synchronization groups for a scene with multiple AV content

time control operations on all streams sharing the same OTB, including the one initially receiving the operation request.

Figure 16.6 illustrates a scene with two audio-visual streams from the same content is synchronized through OCR_ES_ID indications. The two couples (Audio, Video) are not synchronized together, allowing independent playback; in that example, one media playback is triggered after 25 s, while the other media playback is triggered upon a user interaction.

BIFS media nodes, inherited from VRML, only had the ability to control when an object (i.e., a (set of) stream(s)) is active, but assumed that the playback of that object would always start from the beginning of the media. To overcome this limitation, media handling in BIFS has been extended with two nodes: *Media Control* and *Media Sensor*. The *MediaControl* node provides the tools needed to start/stop/pause/resume streams at any given media time, to enable looping of full or part of the streams and to mute the stream. A *MediaControl* can control the playback of any media object in the scene, provided that it does not share the same OTB as the main scene. This gives the possibility to control the playback of another sub-scene included in the presentation. The *Media Sensor* node is used to access the media time of the streams it monitors. It can be used to trigger events in the scene based on this media time (trigger events, send fraction events to interpolators, etc.). Both *Media Control* and *MediaSensor* nodes may act on the complete media object identified by its OD_ID, or on one or more parts of that media object. In that case, the associated object descriptor contains a description of the various parts, called media segments, through a *SegmentDescriptor* structure, indicating the name of the segment, its start time and its duration. Segments may overlap in time, only their name shall be unique. This allows designing scenes with simple chaptering, repeating of parts of a media object without having to know the timing of the segment at authoring time. When playing an entire media streams with defined segments, the *MediaSensor* node enables retrieving the name, duration and start

time of the active media segment. The control panel in our multimedia presentation example would feature both a *MediaControl* node, to allow navigation in the video stream, and a *MediaSensor* node, to give feedback on the current media payback position. If chapters were to be added to simplify navigation, the video Object Descriptor would include a list of *SegmentDescriptor* (one per chapter), the panel would display the list of chapters and user selection of one chapter would result in replacing the *MediaControl* target media URL by "od://VIDEO_OD_ID#ChapterName"; the media would then start playback from the media time associated with "Chapter Name" in the video.

16.3.1.3 Multiple Scene Timelines and Timeline Manipulations

A multimedia scene may typically present media streams coming from different locations, with different, variable network behaviors (round-trip time, bandwidth). In most common cases, a media player will pause all the streams declared as synchronized as soon as one of the stream buffer requirements is no longer met. While this is interesting for some content, such as simple audio-visual only playback, an author may sometime tolerate a delay on one of the media stream, as long as that delay is not too important. In order to help authors synchronize such streams without any knowledge of the underlying network conditions, MPEG-4 BIFS defines a model called FlexTime, inspired by SMIL and its *syncTolerance* attribute. The FlexTime model lets an author define how a media player should behave when media streams targeted for synchronous playback (through OCR sharing) suffer different delivery latencies. It defines constraints on how much each media playback rate can be stretched or shrunk. These constraints may be applied even on streams with different duration, for example to ensure an animation normally lasting 60 s will begin and end at the same times than a 70 s video. The model is based on two nodes, *TemporalTransform* and *TemporalGroup*. It allows an author to express, through a *TemporalTransform* node, a set of constraints on children nodes or on a media object: maximum shrink and stretch ratio, how the stretching can be done: freeze of the rendering of the stream, linear decrease of rendering rate or repeat of the frame(s) and how the shrinking can be done (stop of the rendering or linear increase of the rendering rate).

The *TemporalGroup* node controls the rendering of its children *TemporalTransform* nodes by specifying how the different timelines of each *TemporalTransform* should be played together: start at the same time, end at the same time (with or without starting at the same time), or meet each other, i.e., the end of one *TemporalTransform* timeline triggers the beginning of the following *TemporalTransform* listed in the children nodes.

An optional list of priorities is also given for each child. The child with the highest priority is the one selected as the anchor. For example, if the *TemporalTransform* nodes are specified to end at the same time, the one with the highest priority will be used as indicating the reference end time, and the other timelines will be shrunk or stretched accordingly to meet the end time constraint.

In our presentation example, FlexTime could be used to allow some drifting of the ROI compared to audio and video: if the ROI is bigger than the tracked object in the video, it may be displayed with some delay or advance; if the ROI stream comes from a different location and suffers different latency than the audio and video streams, enforcing strict synchronization in the case would trigger unneeded re-buffering. Using FlexTime would let the author specify the sync tolerance for the ROI stream and prevent those re-buffering.

The main complexity of the FlexTime model is that it overrides the original notion of scene time of the former model, which consists of a single timeline, with a collection of multiple loosely synchronized timelines. In the former model, all events triggered at a certain rendering frame have the same associated time (the scene time at that frame), while in FlexTime each event has a time which depends on the *TemporalTransform* associated with the node triggering the event. FlexTime therefore introduces complex behaviors when time values are transmitted between nodes belonging to different *TemporalTransform* timelines.

16.3.2 Timing in MPEG-4 LASeR

MPEG-4 Lightweight Application Scene Representation (LASeR) [5] is a language in textural and binary formats for the representation of spatiotemporal relationships of media objects in 2D space only. The standard (ISO/IEC 14496-20) derives from the Scalable Vector Graphics (SVG) Tiny 1.2 specification [6] and extends it following the BIFS design (binary format, additional graphical primitives, etc.). More importantly for synchronization, it defines of set of commands to modify the attributes of the SVG nodes, insert or remove nodes at given times. Just like for BIFS, commands can be carried in a main LASeR stream, or in secondary LASeR streams for controllable animations.

A LASeR stream may be carried as any other MPEG-4 Systems streams, referring to other media streams, through their OD_ ID and the Object Descriptor Framework, as seen in the previous sections. The standard also defines a format specifically for LASeR use cases, called Simple Aggregation Format (SAF). This format can be described as a simple multiplexing of LASeR and media AUs, which can be delivered over various existing protocols such as HTTP [7], RTP [8], or MPEG-2 Transport Stream. One particularity of the SAF format is its support for player cache pre-fill of any kind of AUs, such as images or small animation streams. This helps pushing data to the player before their consumption, which may reduce lags in interactions with the content. Note however that the LASeR player cannot discard the sending of the cached as would be the case with HTTP/2 PUSH features [9].

The animation model in MPEG-4 LASeR is the same as the SVG Tiny 1.2 model, which can be seen as a restriction of the SMIL timing model covered in [4]. The main restriction is that an SVG document is driven by a single timeline. The timeline can be started when the document loads or when it has finished loading. SVG

Animations are started based on either absolute times on that timeline, or based on times of events occurring on other object, such as click or mouse over events. The underlying buffer and clock management model is the MPEG-4 systems model. The main difference is that indication of clock references is done in the scene for each media element (*audio*, *video,* or *updateSource*) through the *syncReference* attribute indicating the stream acting as the clock reference. The synchronization properties (not synchronized, strict synchronization or synchronization with drift allowed) between the media stream is defined by the SMIL attributes *syncTolerance* and *syncBehaviour*. Another difference with the general MPEG-4 Systems model is that a SAF stream does not carry any OCR; therefore, the timeline of a stream in the multiplex is initialized at the first decoded frame, starting at the decoding timestamp of that frame and running at the same rate as the system clock.

16.3.3 Scene Timing in Broadcast Environment: The MPEG-4 Carousel

In unidirectional point-to-multipoint delivery scenario, such as broadcast or multicast delivery, some streams may not have a high enough frequency of random access points, i.e., of independently decodable data frames, and this can be problematic for bootstrap information. For example, if the streams configuration information (number of streams, types, locations) is sent once every 4 s, a client joining the broadcast or multicast session may have to wait up to 4 s before being able to setup and start decoding. It is therefore often needed to repeat random access point information for such streams. Moreover, such information may change over time, for example when a video stream switches from one codec to another, as can be the case in current broadcast. Without explicit signaling, a client would then need to understand if the newly received information is the same as the previous one used for initialization, which could be quite time consuming. It is therefore common in broadcast protocols to define a mechanism for sending in a cyclic fashion the same information over a period of time, and for signaling when the information changes. This is usually done through version numbering in the headers of the containing data structure (transport packet or higher-level application packet).

This mechanism is called carousel or data carousel and is present in broadcast protocols such as MPEG-2 Transport Stream or file multicast protocols [10, 11].

MPEG-4 Systems define new types of streams that typically have a low frame rate, such as BIFS, OD, or LASeR streams. It is possible to use traditional carousel but MPEG-4 defines a slightly different carousel from traditional ones. This is due to the nature of these new types of streams. The main difference between streaming of scene description data and audio or video data is the handling of random access points. In both cases, random access points allow the decoder to start processing the stream from that point on. In audio and video streams, the following random access

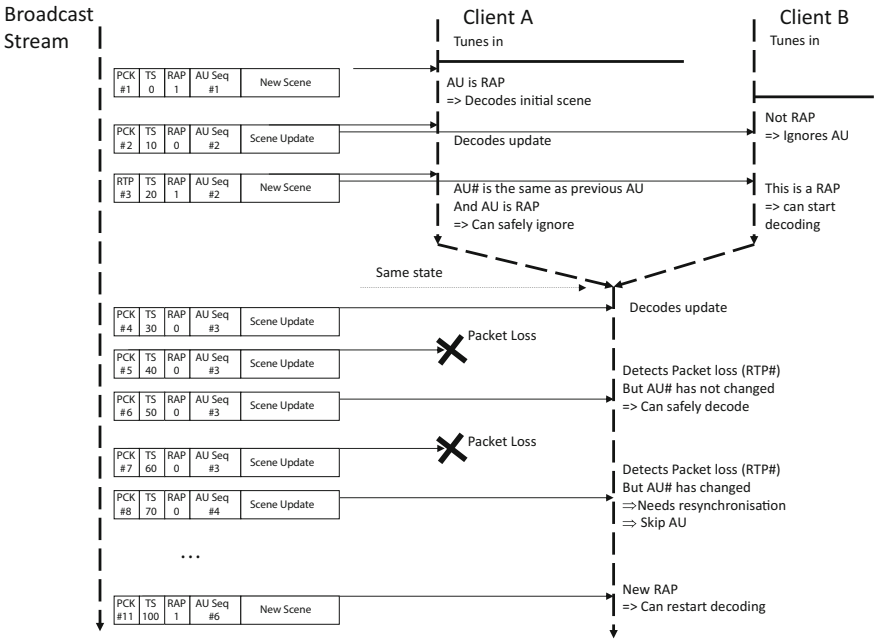


Fig. 16.7 MPEG-4 carousel example

points are decoded like any other AUs. But, in scene description streams, random access points usually carry, and therefore replace, the complete scene. If interactivity features were present in the content, decoding subsequent random access points would reset the interactivity state, hence losing all local interactivity at the client side. In our example, this could mean that when the user has selected an ROI to view its information, this ROI would be deselected and the ROI information area hidden upon loading the following random access point AU of the main BIFS scene, since the initial state of the scene is with no ROI selected. In order to avoid this issue, random access points access units for scene description streams follow carousel techniques: All consecutive random access points corresponding to the same scene state are marked with the same sequence number, allowing a client already connected to discard subsequent random access points while enabling new clients to tune in, as described in the upper part of Fig. 16.7.

Usually, when detecting a packet loss in a stream carrying scene description data, a decoder cannot know if the information loss results in part of the scene being lost, and consequently whether future updates on the scene would be decodable. For example, if a packet carrying a new part of the scene is lost, and the following packet containing an update to this new part is received, the processing of the second packet will fail. However, if both packets were only replacing some properties, such as replacing text content or changing a color, the second packet would be decodable even if the first one were lost. Without explicit signaling of

such packets, a decoder would always reload the first random access point received after the loss, potentially reloading all interactivity information. The MPEG-4 carousel uses the sequence number to distinguish those cases: If consecutive non-random access points of AU have the same sequence number, they correspond to the same scene state and can be safely decoded, regardless of how many AUs were lost in-between them. This is illustrated in the bottom part of Fig. 16.7.

Processing of random access points in MPEG-4 BIFS may have an impact on the synchronization of media streams. As seen previously, *startTime* and *stopTime* fields drive time activation; in MPEG-4 BIFS, the values (i.e., in BIFS commands) for these fields are encoded relative to the composition time of the AU carrying them. For example, a *startTime* field coded value of 2 s carried in a command executed at CTS 30000 s will result in a *startTime* value of 30002. This behavior allows for two use cases in broadcast environments:

- Animation cycles starts from their beginning immediately when the client connects. In this case, each random access point AU in the carousel will have its CTS corresponding to the current scene time. A good example for that is a welcome animation.
- Animation cycles starts from the same point in time regardless of the client connection time, in order to ensure client display is always identical. In this case, all random access points AUs in the carousel will have the same CTS corresponding to the scene time at which the animation was starting. A good example for such use case is a timer displaying the amount of time still available in a live event (game, voting, etc.).

The MPEG-4 carousel can be used on any transport layer supporting the carriage of streams using the MPEG-4 Sync Layer and is mainly used for carriage over MPEG-2 Transport Stream or RTP.

16.4 Deployments

Initially published in 1998, the MPEG-4 Systems standard provides a modern infrastructure for authoring complex multimedia presentations, designed for constrained, low-capacity devices. Some of its features have since then been addressed by other standards, such as HTML5 Media, CSS Animations, or HbbTV 2.0, not without inspiration from the MPEG-4 model.

In current interactive media distribution chains, HTML has been overtaking most of the languages designed for TV and mobile in the past decades, such as MHEG-5 [12], MHP [13], MPEG-4, or 3GPP Dynamic and Interactive Multimedia Scenes (DIMS). With the rapid increase of browser capacities, and powerful JavaScript interfaces enabling control of various part of the audio and video media pipeline, latest interactive media systems deployments now move to HTML-5-based systems. While most of the functionalities of MPEG-4 systems can currently be reproduced in a HTML + JavaScript-based environment, as is also the

case with VRML, browsers still lack support for the fine-grain possibilities offered by MPEG-4 systems as far as synchronization is concerned. Frame-accurate control from JavaScript of the media rendering chain is not guaranteed, which makes the task of frame-accurate synchronization of overlay (HTML or SVG) graphics with media currently impossible. Additionally, frame-accurate (or sample-accurate) synchronization of multiple media streams is still not supported in most browsers; this can be explained by a strong focus on mainstream (single audio–video–subtitle content) media consumption tools in browser developments, covering the needs for most common media applications.

While HTML-5 is seen as a key standard in online multimedia, it still suffers from a high implementation complexity and footprint, as well as potential security risks due to the heavy usage of JavaScript in application design. The MPEG-4 BIFS and LASeR standards were designed to provide lightweight multimedia scene descriptions, without any need for programmatic code to be embedded in the content. The MPEG-4 Systems standard and its BIFS scene format have been deployed in the Terrestrial Digital Multimedia Broadcasting (T-DMB) standard used for mobile television and radio broadcast, providing interactive applications and media synchronization for advanced TV and radio services. The MPEG-4 LASeR standard was also used as the base for 3GPP Dynamic and Interactive Multimedia Scenes (DIMS); however, neither standards have reached mass-market and long-running deployments.

16.5 Conclusion

In this chapter, we have presented the media synchronization tools defined in the MPEG-4 Systems standard, which can be classified in two categories:

- Intra- and inter-stream synchronization descriptions, handled at the Object Descriptor framework level,
- Animation tools and media timeline tools handled at the scene description level.

The MPEG-4 Systems standard defines a powerful architecture for interactive multimedia, allowing mixing streams from different sources such as broadcast and broadband ones, providing frame-accurate media playback control and a unified layout and animation model for 2D and 3D content. Designed for constrained devices before the advent of smartphones and smart TVs, its principles can still be observed in more recent standards such as HTML-5 Media. Successfully deployed as part of T-DMB, the standard yet suffered in its adoption by the industry from being too ambitious in the early years and too far from current media development practices, mostly based on HTML. Reference code for the standard can be freely downloaded from the ISO Website [14], and a more complete, open-source implementation of the standard is provided by the GPAC project [15].

Definitions

Timestamp time label assigned to a block of data in a media delivery system; timestamps are usually integer numbers, converted back to seconds using a *timescale* or *timestamp resolution* value expressing the number of integers in a second; for example, a timestamp of 180000 in a timescale of 90000 indicates a time of 2 s.

Decoding Time Stamp time at which a media frame has to be available for the decoder.

Composition Time Stamp (or Presentation Time Stamp) time at which a media frame must be presented to the user; composition and decoding times usually only differ for video stream using bi-directional motion estimation schemes, hence requiring a coding order different from the display order.

Intra-stream Synchronization process and signaling required to reproduce a time-accurate, jitter-free playback of a given media stream; for example, signaling and maintaining the exact time lapse between successive video frames in a fixed or variable frame rate stream.

Inter-stream Synchronization process and signaling required to reproduce a time-accurate playback of multiple media streams; for example, signaling and maintaining stream timestamps for achieve lip synchronization between an audio and a video stream.

Clock Reference value against which decoding (resp. composition) timestamps are compared to trigger the decoding (resp. display) process; depending on the delivery systems, the clock reference may be delivered together with the content, as is the case in broadcast environments, derived from the wall-clock time another time source such as internet time, or derived from any other local system clock/counter (CPU cycle count, audio output sample rate, etc.).

References

1. Pereira, F.C., Ebrahimi, T.: The MPEG-4 Book. Prentice Hall PTR, Upper Saddle River, NJ, USA (2002)
2. Van der Meer, J.: Fundamentals and Evolution of Mpeg-2 Systems: Paving the MPEG Road, 1st edn. Wiley (2014)
3. Real time streaming protocol. https://www.ietf.org/rfc/rfc2326.txt
4. Synchronized multimedia integration language. https://www.w3.org/TR/SMIL3/
5. Dufourd, J.C., Avaro, O., Concolato, C.: LASeR: the MPEG standard for rich media services. IEEE Multimedia (2005)
6. Scalable vector graphics. https://www.w3.org/TR/SVG12/
7. Hyper-text transfer protocol. https://tools.ietf.org/html/rfc7230

8. RTP payload format for transport of MPEG-4 elementary streams. https://tools.ietf.org/html/rfc3640
9. Hyper text transfer protocol version 2 (HTTP/2). https://tools.ietf.org/html/rfc7540
10. File delivery over unidirectional transport. https://tools.ietf.org/html/rfc6726
11. FCAST: object delivery for the asynchronous layered coding (ALC) and NACK-oriented reliable multicast (NORM) protocols. https://tools.ietf.org/html/rfc6968
12. Meyer-Boudnik, T., Effelsberg, W.: MHEG explained. IEEE Multimedia **2**(1), 26–38 (1995)
13. Piesing, J.: The DVB multimedia home platform (MHP) and related specifications. Proc. IEEE **94**(1), 237–247 (2006)
14. ISO/IEC 14496-5, reference software, ISO/IEC publically available standards. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=36086
15. Le Feuvre, J., Concolato, C., Moissinac, J. C.: GPAC: open source multimedia framework. In: Proceedings of the 15th ACM International Conference on Multimedia, pp. 1009–1012. ACM, Sept 2007. https://gpac.io

Chapter 17
Media Synchronization on the Web

Ingar M. Arntzen, Njål T. Borch and François Daoust

Abstract The Web is a natural platform for multimedia, with universal reach, powerful backend services, and a rich selection of components for capture, interactivity, and presentation. In addition, with a strong commitment to modularity, composition, and interoperability, the Web should allow advanced media experiences to be constructed by harnessing the combined power of simpler components. Unfortunately, with timed media this may be complicated, as media components require synchronization to provide a consistent experience. This is particularly the case for distributed media experiences. In this chapter we focus on temporal interoperability on the Web, how to allow heterogeneous media components to operate consistently together, synchronized to a common timeline and subject to shared media control. A programming model based on external timing is presented, enabling modularity, interoperability, and precise timing among media components, in single-device as well as multi-device media experiences. The model has been proposed within the W3C Multi-device Timing Community Group as a new standard, and this could establish temporal interoperability as one of the foundations of the Web platform.

Keywords Media synchronization · Media orchestration · Motion model
Timing object · Multi-device

17.1 Introduction

The Web is all about modularity, composition, and interoperability, and this applies across the board, from layout and styling defined in *Hypertext Markup Language*

I. M. Arntzen (✉) · N. T. Borch
Norut Northern Research Institute, Tromsø, Norway
e-mail: ingar.arntzen@norut.no

N. T. Borch
e-mail: njaal.borch@norut.no

F. Daoust
World Wide Web Consortium (W3C), Paris, France
e-mail: fd@w3.org

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_17

(HTML) to JavaScript-based tools and frameworks. Unfortunately, there is a notable exception to this rule. Composing presentations from multiple, timed media components is far from easy. For example, consider a Web page covering motor sport, using Web Audio [43] for sound effects and visuals made from HTML5 [18] videos, a map with timed GPS data, WebGL [45] for timed infographics, a timed Twitter [38] widget for social integration, and finally an ad banner for paid advertisements timed to the race.

In this chapter we focus on media synchronization challenges of this kind, making multiple, heterogeneous media components operate consistently with reference to a common timeline, as well as common media control. We call this *temporal interoperability*. Lack of support for temporal interoperability represents a significant deviation from the core principles of the Web. Harnessing the combined powers of timed media components constitutes a tremendous potential for Web-based media experiences, both in single-device and multi-device scenarios.

The key to temporal interoperability is finding the right approach to media synchronization. There are two basic approaches: *internal timing* or *external timing*. Internal timing is the familiar approach, where media components are coordinated by manipulating their control primitives. External timing is the opposite approach, where media components are explicitly designed to be parts of a bigger experience, by accepting direction from an external timing source.

Though internal timing is currently the predominant approach in Web-based media, external timing is the key to temporal interoperability. If multiple media components are connected to the same source of external timing, synchronized behavior across media components follows by implication. This simplifies media synchronization for application developers. Furthermore, by allowing external timing sources to be synchronized and shared across a network, external timing is also a gateway to precise, distributed multimedia playback and orchestration on the Web platform.

This chapter provides an introduction to external timing as well as the flexible media model and programming model that follow from this approach. The programming model is proposed for standardization within the *W3C Multi-device Timing Community Group (MTCG)* [32] to encourage temporal interoperability on the Web platform. The *timing object* is the central concept in this initiative, defining a common interface to external timing and control for the Web. The MTCG has published a draft specification for the timing object [7] and also maintains *Timingsrc* [3], an open source JavaScript implementation of the timing object programming model. The chapter also describes the *motion model*, which provides online synchronization of timing objects.

The chapter is structured as follows: Sect. 17.3 defines media synchronization as a term and briefly presents common challenges for synchronization on the Web. The motion model is presented in Sect. 17.4 followed by an introduction to temporal interoperability and external timing in Sect. 17.5. Section 17.6 surveys the abilities and limitations of Web technologies with respect to media synchronization. Section 17.7 gives a more detailed presentation of the motion model, including online motion synchronization, synchronization of AV media and timed data. Evaluation is presented in Sect. 17.9. Section 17.10 briefly references the standardization initiative before conclusions are given in Sect. 17.11.

17.2 Central Terms

This section lists central terms used in this chapter.

Timeline: Logical axis for media presentation. Values on the timeline are usually associated with a unit, e.g., seconds, milliseconds, frame count, or slide number. Timelines may be infinite or bounded by a range (i.e., minimum and maximum values).

Clock: A point moving predictably along a timeline, at a fixed, positive rate. Hardware clocks ultimately depend on a crystal oscillator. System clocks typically count seconds or milliseconds from *epoch* (i.e., Jan 1, 1970 UTC) and may be corrected by clock synchronization protocols (e.g., NTP [27], PTP [13]). From the perspective of application developers, the value of a clock may be read, but not altered.

Motion: A unifying concept for *media playback* and *media control*. Motion represents a point moving predictably along a timeline, with added support for flexibility in movement and interactive control. Motions support discrete jumps on the timeline, as well as a variety of continuous movements expressed through velocity and acceleration. Not moving (i.e., paused) is considered a special case of movement. Motion is a generalization over classical concepts in multimedia, such as *clocks*, *media clocks*, *timers*, *playback controls*, *progress*. Motions are implemented by an *internal clock* and a *vector* describing current movement (position, velocity, acceleration), timestamped relative to the internal clock. Application developers may update the movement vector of a motion at any time.

Timed data: Data whose temporal validity is defined in reference to a timeline. For instance, the temporal validity of subtitles is typically expressed in terms of points or intervals on a media *timeline*. Similarly, the temporal validity of video frames essentially maps to frame-length intervals. Timed scripts are a special case of timed data where data represents functions, operations, or commands to be executed.

Continuous media: Typically audio or video data. More formally, a subset of *timed data* where media objects cover the timeline without gaps.

Media component: Essentially a player for some kind of timed data. Media components are based on two basic types of resources: timed data and motion. The timeline of timed data must be mapped to the timeline of motion. This way, motion defines the temporal validity of timed data. At all times, the media component works to produce correct media output in the UI, given the current state of timed data and motion. A media component may be anything from a simple text animation in the *Document Object Model (DOM)*, to a highly sophisticated media framework.

User agent: Any software that retrieves, renders, and facilitates end user interaction with Web content, or whose user interface is implemented using Web technologies.

Browsing context: JavaScript runtime associated with Web document. Browser windows, tabs, or *iframes* each have their own browsing context.

Iframe: Web document nested within a Web document, with its own browsing context.

17.3 Media Synchronization

Dictionary definitions of *media synchronization* typically refer to presentation of multiple instances of media at the same moment in time. Related terms are *media orchestration* and *media timing*, possibly emphasizing more the importance of media control and timed scheduling of capture and playback. In this chapter we use the term *media synchronization* in a broad sense, as a synonym to *media orchestration* and *media timing*. We also limit the definition in a few regards:

- Media synchronization on the Web is client-side and clock-based. The latencies and heterogeneity of the Web environment require a clock-based approach for acceptable synchronization.
- Media synchronization involves a media component and a clock. The term *relative synchronization* is reserved for comparisons between two or more synchronized media components.

17.3.1 Challenges

Media synchronization has a wide range of use cases on the Web, as illustrated by Table 17.1. Well known use cases for synchronization within a single Web page include multi-angle video, accessibility features for video, ad insertion, as well as media experiences spanning different media types, media frameworks, or iframe boundaries. Synchronization across Web pages allows Web pages to present alternative views into a single experience, dividing or duplicating media experiences across devices. Popular use cases in the home environment involve collaborative viewing,

Table 17.1 Common challenges for media synchronization on the Web

Synchronization challenges	Use cases
Across media sources	Multi-angle video, ad insertion
Across media types	Video, WebAudio, animated map
Across iframes	Video, timed ad banner
Across tabs, browsers, devices	Split content, interaction
Across platforms	Web, native, broadcast
Across people and groups	Collaboration, social
Across Internet	Global media experiences

multi-speaker audio, or big screen video synchronized with related content on handheld devices. The last use cases on the list target global scenarios, such as distributed capture and synchronized Web visualizations for a global audience.

17.3.2 Approach

The challenges posed by all these use cases may be very different in terms of complexity, requirements for precision, scale, infrastructure, and more. Still, we argue that a single, common solution would be beneficial. Implementing specific solutions for specific use cases is very expensive and time-consuming and lays heavy restrictions on reusability. Even worse, circumstances regarding synchronization may change dynamically during a media session. For instance, a smartphone involved in synchronization over the local network will have to change its approach to media synchronization once the user leaves the house, or switches from WiFi to the mobile network. Crucially though, by solving media synchronization across Internet, all challenges listed above are solved by implication. For instance, if video synchronization is possible across Web pages on the Internet, then synchronizing two videos within the same Web page is just a special case. It follows that the general solution to media synchronization on the Web is distributed and global in nature. Locality may be exploited for synchronization, yet only as optimization.

17.4 The Motion Model

The primary objectives of the motion model are global synchronization, *Web availability*, and simplicity for Web developers. Global synchronization implies media synchronization across the Internet. Web availability means that no additional assumptions can be introduced for media synchronization. If a Web browser is able to load an online Web page, it should also be able to synchronize correctly. The model proposed for this can be outlined in three simple steps:

- Media clock and media controls are encapsulated in one concept and represented as a stateful resource. This chapter uses the term *motion*[1] for this concept.
- A *motion* is an online resource, implying that it is hosted by a server and identifiable by a *Universal Resource Locator (URL)*.
- *Media components*[2] synchronize themselves relative to online motions.

[1] *Motion* as in *motion pictures*. *Moving through media* still remains a good way to conceptualize media experiences, not least as media experiences become virtual and immersive.

[2] *Media component:* anything from a simple DOM element with text, to a highly sophisticated media player or multimedia framework.

According to the model, media synchronization should be a consequence of connecting multiple media components to the same online motion. This way, rich synchronized multi-device presentation may be crafted by connecting relevant media components to the same online motion, as illustrated in Fig. 17.1.

Importantly, the practicality of the motion model depends on Web developers being shielded from the complexities of distributed synchronization. This is achieved by having a *timing object* locally in the Web browser. The timing object acts as an intermediary between media components and online motions, as illustrated in Fig. 17.2. This way, the challenge of media synchronization is divided into two parts.

- *motion synchronization*: timing object precisely synchronized with online motion (Internet problem).
- *component synchronization*: media component precisely synchronized with timing object (local problem).

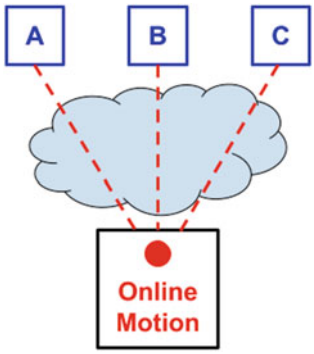


Fig. 17.1 Media components on three different devices (A, B, C), all connected to an online motion (red circle). Media control requests (e.g., pause/resume) target the online motion and are transmitted across the Internet (light blue cloud). The corresponding state change is communicated back to all connected media components. Each media component adjusts its behavior independently

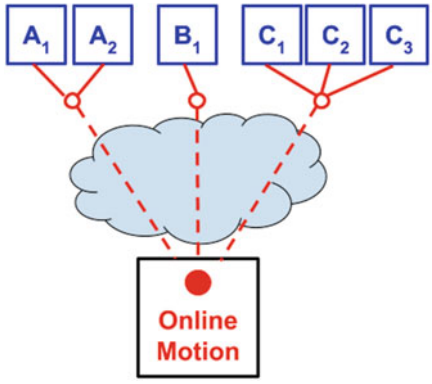


Fig. 17.2 Timing objects (red unfilled circles) mediate access to online motion. Timing objects may be shared by independent media components within the same browsing context

Motion synchronization ensures that timing objects connected to the same online motion are kept precisely synchronized. The logic required for motion synchronization could be supported by Web browsers natively (if standardized) or imported into Web pages as a third-party JavaScript library. Motion synchronization is outlined in Sect. 17.7.3.

Component synchronization implies that a media component continuously strives to synchronize its activity relative to a timing object. As such, component synchronization is a local problem. Media components always interface with timing objects through a well-defined *Application Programmer Interface (API)* (see Sect. 17.7.1). Examples of component synchronization are provided in Sects. 17.7.4 and 17.7.5.

17.5 Temporal Interoperability

Temporal interoperability implies that multiple, possibly heterogeneous media components may easily be combined into a single, consistently timed media experience [5]. We argue that temporal interoperability must be promoted as a principal feature of the Web, and finding the right approach to media synchronization is key to achieving this. In this section we distinguish two basic approaches, *internal timing* and *external timing*, and explain why external timing is better suited as a basis for temporal interoperability. Note that external timing is provided by motions[3] according to the motion model.

17.5.1 Internal Timing

Internal timing (Fig. 17.3—left) is the most familiar approach, where media components are typically media players or frameworks internalizing aspects of timing and control. Synchronizing such media components is an external process, utilizing the control primitives provided by each component.

Internal timing means that media components implement timed operations internally, based on the system clock or some other (hardware) timer and state representing motion. Timed operations may also be affected by other internal factors, such as the buffering of timed data, time consumption in processing, or delays in UI pipelines. The outside world is typically given access to the internal motion of the media component via interactive elements in the *User Interface (UI)* or programmatically through control primitives defined in the component API. For example, the HTML5 media element allows media playback to be requested by clicking the play button or invoking the play method. The media element then organizes media playback in its own time, subject to delays in initialization procedures, buffering, decoding, and AV subsystems.

[3] Mediated by timing objects.

Fig. 17.3 Blue rectangles represent media components, red symbols represent motion, and green symbols represent the process of media synchronization. To the left: internal timing and external media synchronization. To the right: external timing and internal media synchronization

17.5.2 External Timing

External timing (Fig. 17.3—right) is the opposite approach, where media components consider themselves parts of a bigger experience. Such media components are explicitly designed to take direction from an external motion and always do their best to synchronize their own behavior accordingly. If multiple media components are connected to the same external motion, synchronized behavior across media components follows by implication. In this approach, media synchronization is redefined as an internal challenge, to be addressed by each media component independently.

Media control: The external timing approach implies that control over the media component is exercised indirectly, by manipulating the external motion instead of the media component. For instance, if the external motion is paused or time-shifted, the media component must react accordingly. Appropriate controls for the media component may still be exposed through the UI or API of the component. However, such control requests must be routed to the external motion. This ensures that control applies to all media components connected to the same external motion. It also ensures that media components may process control requests without regard to the origin of the request. Media components directed by external motions may still make use of an internal clock. Importantly though, the external motion takes precedence, so deviations must be compensated for by adjusting the internal clock.

Precision: Precision is a key ambition in media synchronization. With internal timing, synchronization with other media is performed using the control primitives that each media component defines. In the Web environment, such control primitives have typically not been designed with precise timing in mind (see Sect. 17.6.1). This makes high-quality synchronization hard to achieve. In this model, media synchronization generally gets more difficult as the number of components increases. Heterogeneity in media types and control interfaces complicate matters further. For precise synchronization, external timing appears to be a better approach. Media synchronization is solved internally in media components, where it can be implemented with unrestricted access to the internal state and capabilities of the component. Furthermore, the synchronization task is shifted from external application developers to the author of the media component. This makes sense, as the author likely has better

understanding of how the media component works. It also ensures that the problem may be solved once, instead of repeatedly by different application developers.

Buffering: Another distinctive feature of the external motion approach is that motion is not sensitive to the internal state (e.g., data availability) of any media component. For instance, external motion might describe playback while a particular media component still lacks data. In the external motion approach, media components must always align themselves with the external motion, to the best of their abilities. For example, media components may adapt by buffering data further ahead, changing to a different data source (e.g., lower bitrate) or even changing to a different presentation mode (e.g., audio only). This way, playback may continue undisturbed and media components join in as soon as they are able to. This is particularly important in multi-device scenarios, where a single device with limited bandwidth might otherwise hold back the entire presentation. On the other hand, if the readiness of a particular media component is indeed essential to the experience, this may be solved in application code, by pausing and resuming the external motion.

Master–Slave: Asymmetric master–slave synchronization is a common pattern in media synchronization. The pattern implies that internal motion of a master media component is used as external motion for slave media components. However, with multiple media components all but one must be a slave. In the external timing approach all media components are slaves, and the external motion itself is the master. This avoids added complexities of the master–slave pattern and provides a symmetric model where each media component may request control via the external motion. On the other hand, if asymmetry is indeed appropriate for a given application, this may easily be emulated. For instance, applications may ensure that only one specific media component may issue control requests to the external motion.

Live and On-demand: Solutions for live media often target minimized transport latency for real-time presentation. In other words, the internal motion of live media components is tied to data arrival. This may be problematic in some applications, as differences in transport latency imply that media components will be out of sync. For example, live Web-based streaming solutions may be seconds apart, even on the same network. Timing issues with live media are even more evident in rich media productions involving multiple live feeds with very different production chains and transport mechanisms. The external timing approach provides the control and flexibility needed for applications to deal with these realities in appropriate ways. With an external motion representing the official live motion, multiple live media sources may be presented in a time-consistent way across components and screens. Such a live motion could be selected to cover at least a majority of viewers. Furthermore, inability to follow the official live motion would be detected by media components internally, potentially triggering application-specific reactions. For instance, the viewer could be prompted to switch to a private, slightly time-shifted motion suitable for his or her specific environment.

17.6 State of the Web

With temporal interoperability established as a goal for the Web, this section surveys current abilities and limitations of the Web with respect to media synchronization. The Web platform[4] is composed of a series of technologies centered around the *Hypertext Markup Language (HTML)*. These technologies have been developed over the years and have grown steadily since the advent of *HTML5* [18], allowing Web applications to access an ever-increasing pool of features such as local storage, geolocation, peer-to-peer communications, notifications, background execution, media capture, and more. This section focuses on Web technologies that produce or consume timed data and highlights issues that arise when these technologies are used or combined with others for synchronization purposes. These issues are classified and summarized at the end of the section. Please note that this section is written early 2017, and references technologies that are still under development.

17.6.1 HTML

First versions of the HTML specification (including HTML3.2 [17]) were targeting static documents and did not have any particular support for timed playback. HTML5 introduced the Audio and Video media elements to add support for audio and video data playback. Web applications may control the playback of media elements using commands such as *play* or *pause* as well as properties such as *currentTime* (the current media offset) and *playbackRate* (the playback speed). In theory, this should be enough to harness media element playback to any synchronization logic that authors may be willing to implement. However, there are practical issues:

1. The playback offset of the media element is measured against a media clock, which the specification defines as: *user-agent defined, and may be media resource-dependent, but [which] should approximate the user's wall clock.* In other words, HTML5 does not impose any particular clock for media playback. One second on the wall clock may not correspond to one second of playback, and the relationship between the two may not be linear. Two media elements playing at once on the same page may also follow different clocks, and thus media offset of these two media elements may diverge over time even if playback was initiated at precisely the same time.
2. HTML5 gives no guarantee about the latency that the software and the hardware may introduce when the play button is pressed, and no compensation is done to resorb that time afterward.
3. The media clock in HTML5 automatically pauses when the user agent needs to fetch more data before it may resume playback. This behavior matches the

[4]In this chapter, the Web is seen through the eyes of an end user browsing the Web with his/her favorite *user agent* in 2017.

expectations of authors for most simple media use cases. However, more advanced scenarios where media playback is just a part of a larger and potentially cross-device orchestration would likely require that the media clock keeps ticking no matter what.

4. The *playbackRate* property was motivated by the fast forward and rewind features of *Digital Video Disc (DVD)* players and previously *Videocassette Recorders (VCR)*. It was not meant for precise control of playback velocity on the media timeline.

To address use cases that would require synchronized playback of media elements within a single page, for instance, to play a sign language track as an overlay video on top of the video it describes, HTML5 introduced the concept of a *media controller* [21]. Each media element can be associated with a media controller, and all the media elements that share the same media controller use the same media clock, allowing synchronized playback. In practice though, browser vendors did not implement media controllers and the feature was dropped in HTML5.1 [20]. It is also worth noting that this mechanism was restricted to media elements and could not be used to orchestrate scenarios that involved other types of timed data.

While sometimes incorrectly viewed as a property of the JavaScript language, the *setTimeout*, *setInterval*, and other related timer functions, which allow apps to schedule timeouts, are actually methods of the *window* interface, defined in HTML5. These methods take a timeout counter in milliseconds, but the specification only mandates that Web browsers wait until at least this number of milliseconds have passed (and only provided the Web page has had the focus during that time). In particular, Web browsers may choose to wait a further arbitrary length of time. This allows browsers to optimize power consumption on devices that are in low-power mode. Even if browsers do not wait any further, the event loop may introduce further delays (see Sect. 17.6.4). Surprisingly, browsers also fire timers too early on occasion. All in all, the precision of timeouts is not guaranteed on the Web, although experience shows that timeouts are relatively reliable in practice.

17.6.2 SMIL and Animations

Interestingly, one of the first specifications to have been published as a Web standard after HTML3.2 [17], and as early as 1998, was the *Synchronized Multimedia Integration Language (SMIL)* 1.0 specification [35]. SMIL allowed integrating a set of independent multimedia objects into a synchronized multimedia presentation. SMIL 1.0 was the first Web standard to embed a notion of timeline (although it was only implicitly defined). The specification did not mandate precise synchronization requirements: *The accuracy of synchronization between the children in a parallel group is implementation-dependent.* Support for precise timing has improved in subsequent revisions of SMIL, now in version 3.0 [36].

No matter how close to HTML it may be, SMIL appears to Web application developers as a format on its own. It cannot simply be added to an existing Web application to synchronize some of its components. SMIL has also never been properly supported by browsers, requiring plug-ins such as RealPlayer [33]. With the disappearance of plug-ins in Web browsers, authors are left without any simple way to unleash the power of SMIL in their Web applications.

That said, SMIL 1.0 sparked the SMIL Animation specification [37] in 2001, which builds on the SMIL 1.0 timing model to describe an animation framework suitable for integration with *Extensible Markup Language (XML)* documents. SMIL Animation has notably been incorporated in the Scalable Vector Graphics (SVG) 1.0 specification [34], published as a Web standard immediately afterward. It took many years for SVG to take over Flash [1] and become supported across browsers, with the notable exception of SMIL Animations, which *Microsoft* [26] never implemented, and which *Google* [14] now intends to drop in favor of *CSS Animations* and of the *Web Animations specification*.

While still a draft when this book is written, Web Animations [42] appears as a good candidate specification to unite all Web Animation frameworks into one, with solid support from *Mozilla* [29], Google, and now Microsoft. It introduces the notion of a *global clock*:

> a source of monotonically increasing time values unaffected by adjustments to the system clock. The time values produced by the global clock represent wall-clock milliseconds from an unspecified historical moment.

The specification also defines the notion of a *document timeline* that provides time values tied to the global clock for a particular document. It is easy to relate the global clock of Web Animations with other clocks available to a Web application (e.g., the *High Resolution Time* clock mentioned in Sect. 17.6.5). However, the specification acknowledges that the setup of some animations *may incur some setup overhead*, for instance when the user agent delegates the animation to specialized graphics hardware. In other words, the exact start time of an animation cannot be known a priori.

17.6.3 DOM Events

The ability to use scripting to dynamically access and update the content, structure and style of documents was developed in parallel to HTML, with *ECMAScript* (commonly known as JavaScript), and the publication of the *Document Object Model (DOM) Level 1* standard in 1998 [10]. This first level did not define any event model for HTML documents, but was quickly followed by *DOM Level 2* [11] and in particular the DOM Level 2 Events standard [12] in 2000. This specification defines: *a platform- and language-neutral interface that gives to programs and scripts a generic event system.*

DOM events feature a *timeStamp* property used to specify the time relative to the epoch at which the event was created. DOM Level 2 Events did not mandate that property on all events. Nowadays, DOM Events, now defined in the DOM4 standard [41], all have a timestamp value, evaluated against the system clock.

The precision of the timestamp value is currently limited to milliseconds, but Google has now switched to using higher resolution timestamps associated with the *high resolution clock* (see Sect. 17.6.5). On top of improving the precision down to a few microseconds, this change also means that the *monotonicity* of timestamp values can now be guaranteed. *Monotonicity* means that clock values are never decreasing. This change will hopefully be included in a future revision of the DOM standard and implemented across browsers.

17.6.4 The Event Loop

On the Web, all activities (including *events, user interactions, scripts, rendering, networking*) are coordinated through the use of an *event loop*,[5] composed of a queue of tasks that are run in sequence. For instance, when the user clicks a button, the user agent queues a task on the event loop to dispatch the *click* event onto the document. The user agent cannot interrupt a running task in particular, meaning that, on the Web; all scripts run to completion before further tasks may be processed.

The event loop may explain why a task scheduled to run in 2 s from now through a call to the *setTimeout* function may actually run in 2.5 s from now, depending on the number of tasks that need to run to completion before this last task may run. In practice, HTML5 has been carefully designed to optimize and prioritize the tasks added to the event loop, and the scheduled task is unlikely to be delayed by much, unless the Web application contains a script that needs to run for a long period of time, which would effectively freeze the event loop.

Starting in 2009, the Web Workers specification [44] was developed to allow Web authors to run scripts in the background, in parallel with the scripts attached to the main document page, and thus without blocking the user interface and the main event loop. Coordination between the main page and its workers uses message passing, which triggers a *message* event on the event loop.

Any synchronization scenario that involves timed data exposed by some script or event logic will de facto be constrained by the event loop. In turn, this probably restricts the maximum level of precision that may be achieved for such scenarios. Roughly speaking, it does not seem possible to achieve less than 1 milliseconds precision on the Web today if the event loop is involved.

[5]There may be more than one event loop, more than one queue of tasks per event loop, and event loops also have a micro-task queue that helps prioritizing some of the tasks added by HTML algorithms, but this does not change the gist of the comments contained in this section.

17.6.5 High Resolution Time

In JavaScript, the *Date* class exposes the system clock to Web applications. An instance of this class represents a number of milliseconds since January 1, 1970 UTC. In many cases, this clock is a good enough reference. It has a couple of drawbacks though:

1. The system clock is not monotonic, and it is subject to adjustments. There is no guarantee that a further reading of the system clock will yield a greater result than a previous one. Most synchronization scenarios need to rely on the monotonicity of the clock.
2. Sub-millisecond resolution may be needed in some cases, e.g., to compute the frame rate of a script-based animation or to precisely schedule audio cues at the right point in an animation.

As focus on the Web platform shifted away from documents to applications and as the need to improve and measure performance arose, a need for a better clock for the Web that would not have these restrictions emerged. The High Resolution Time specification [15] defines a new clock, *Performance.now()*, that is both guaranteed to be monotonic and accurate to 5 microseconds, unless the user agent cannot achieve that accuracy due to software or hardware constraints. The specification defines the time origin of the clock, which is basically the time when the *browsing context* (i.e., browser window, tab, or iframe) is first created. The very recent High Resolution Time Level 2 specification [16] aims to expose a similar clock to background workers and provides a mechanism to relate times between the browsing context and workers.

It seems useful to point out that the 5 microseconds accuracy was not chosen because of hardware limitations. It was rather triggered by privacy concerns as a way to mitigate so-called cache attacks, whereby a malicious Web site uses high-resolution timing data to fingerprint a particular user. In particular, this sets a hard limit to precision on the Web, that will likely remain stable over time.

17.6.6 Web Audio API

At about the same time that people started to work on the High Resolution Time specification, Mozilla and Google pushed for the development of an API for processing and synthesizing audio in Web applications. The Web Audio API draft specification [43] is already available across browsers. It builds upon an audio routing graph paradigm where audio nodes are connected to define the audio rendering.

Sample frames exposed by the Web Audio API have a *currentTime* property that represents the position on the Audio timeline, according to the hardware clock of the underlying sound card. As alluded to in the specification, this clock *may not be synchronized with other clocks in the system*. In particular, there is little chance that

this clock be synchronized with the High Resolution Time clock, the global clock of Web Animations, or the media clock of a media element.

The group that develops the Web Audio API at W3C investigated technical solutions to overcome these limitations. The API now exposes the relationship between the audio clock and the high-resolution clock, coupled with the latency introduced by the software and hardware, so that Web applications may compute the exact times at which a sound will be heard. This is particularly valuable for cross-device audio scenarios, but also allows audio to be output on multiple sound cards at once on a single device.

17.6.7 Media Capture

W3C started to work on the Media Capture and Streams specification [23] in 2011. This specification defines the notions of *MediaStreamTrack*, which *represents media of a single type that originates from one media source* (typically video produced by a local camera) and of *MediaStream*, which is a group of loosely synchronized MediaStreamTracks. The specification also describes an API to generate MediaStreams and make them available for rendering in a media element in HTML5.

The production of a MediaStreamTrack depends on the underlying hardware and software, which may introduce some latency between the time when the data is detected to the time when it is made available to the Web application. The specification requires user agents to expose the target latency for each track.

The playback of a MediaStream is subject to the same considerations as those raised above when discussing media support in HTML5. The media clock is implementation-dependent in particular. Moreover, a MediaStream is a *live* element and is not seekable. The *currentTime* and *playbackRate* properties of the media element that renders a MediaStream are *read-only* (i.e., media controls do not apply), and thus cannot be adjusted for synchronization.[6]

17.6.8 WebRTC

Work on *Web Real-Time Communication (WebRTC)* and its first specification, the WebRTC 1.0: Real-time Communication Between Browsers specification [46], started at the same time as the work on media capture, in 2011. As the name suggests, the specification allows media and data to be sent to and received from another browser. There is no fixed timing defined, and the goal is to minimize latency.

[6] In the future, it may be possible to re-create a seekable stream out of a MediaStream, thanks to the *MediaRecorder* interface defined in the MediaStream Recording specification [25]. This specification is not yet stable when this book is written.

How this is achieved in practice is up to the underlying protocols, which have been designed to reduce latency and allow peer-to-peer communications between devices.

The WebRTC API builds on top of the Media Capture and Streams specification and allows the exchange of MediaStreams. On top of the synchronization restrictions noted above, a remote peer does not have any way to relate the media timeline of the MediaStream it receives with the clock of the local peer that sent it. The WebRTC API does not expose synchronization primitives. This is up to Web applications, which may for instance exchange synchronization parameters over a peer-to-peer data channel. Also, the MediaStreamTracks that compose a MediaStream are essentially treated independently and realigned for rendering on the remote peer, when possible. In case of transmission errors or delays, loss of synchronization, e.g., between audio and video tracks, is often preferred in WebRTC scenarios to avoid accumulation of delays and glitches.

17.6.9 Summary

While the High Resolution Time clock is a step in the right direction, the adoption is still incomplete. As of early 2017, given an arbitrary set of timed data composed of audio/video content, animations, synthesized audio, events, and more there are several issues Web developers need to face to synchronize the presentation:

1. Clocks used by media components or media subsystems may be different and may not follow the system clock. This is typically the case for media elements in HTML5 and for the Web Audio API.
2. The clock used by a media component or a media subsystem may not be monotonic or sufficiently precise.
3. Additionally, specifications may leave some leeway to implementers on the accuracy of timed operations, leading to notable differences in behavior across browsers.
4. Operations may introduce latencies that cannot easily be accounted for. This includes running Web Animations, playing/resuming/capturing media, or scheduling events on the event loop.
5. Standards may require browsers to pause for buffering, as typically happens for media playback in HTML5. This behavior does not play well with the orchestration of video with other types of timed data that do not pause for buffering.
6. The ability to relate clocks is often lost during the transmission of timestamps from one place to another, either because different time origins are used, as happens between an application and its workers, or because the latency of the transmission is not accounted for, e.g., between WebRTC peers. At best, applications developers need to use an out-of-band mechanism to convert timestamps and account for the transport latency.

7. When they exist, controls exposed to harness media components may not be sufficiently fine-grained. For example, the *playbackRate* property of media elements in HTML5 was not designed for precise adjustments, and setting the start time of a Web Animation to a specific time value may result in a significant jump between the first and second frames of the animation.

Small improvements to Web technologies should resolve some of these issues, and discussions are underway in relevant standardization groups at W3C when this book is written. For example, timestamps in DOM Events may switch to using the same *Performance.now()* clock. This is all good news for media synchronization, although it may still take time before the situation improves.

We believe that a shift of paradigm is also needed. The Web is all about modularity, composition, and interoperability. Temporal aspects have remained an internal issue specific to each technology until now. In the rest of this chapter, a programming model is presented to work around the restrictions mentioned above, allowing media to be precisely orchestrated on the Web, even across devices.

17.7 Motion

Motion is a simple concept representing playback state (media clock), as well as functions for accessing and manipulating this state (media controls). As such, similar constructs are found in most multimedia frameworks.

As illustrated in Fig. 17.4, motion represents movement (in real time) of a point, along an axis (timeline). At any moment the point has well-defined position, velocity, and acceleration.[7] Velocity and acceleration describe continuous movements. Velocity is defined as position change per second, whereas acceleration is defined as position change per second squared. Discrete jumps on the timeline are also supported, simply by modifying the position of the motion. A discrete jump from position A to C implies that the transition took no time and that no position B (between A and C) was visited. Not moving (i.e., zero velocity and acceleration) is a special case of movement.

Internal State. Motion is defined by an internal clock and a vector (position, velocity, acceleration, timestamp). The vector describes the initial state of the current movement, timestamped relative to the internal clock. This way, future states of the motion may be calculated precisely from the initial vector and elapsed time. Furthermore, application programmers may control the motion simply by supplying a new initial vector. The motion concept was first published under the name Media State Vector (MSV) [6].

[7] Some animation frameworks support acceleration. Acceleration broadens the utility of motions, yet will likely be ignored in common use cases in classical media (see Sect. 17.7.2).

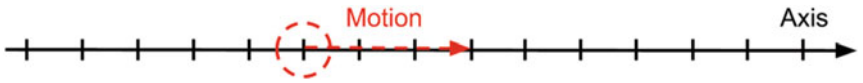


Fig. 17.4 Motion: point moving along an axis. The current position is marked with a red circle (dashed), and forward velocity of 3 units per second is indicated by the red arrow (dashed)

17.7.1 Timing Object API

Timing objects provide access to motions. Timing objects may be constructed with a URL to an online motion. If the URL is omitted, it will represent a local motion instead.

```
var URL = "...";
var timingObject = new TimingObject(URL);
```

Listing 1 Constructing a timing object.

The *Timing object API* defines two operations, *query* and *update*, and emits a *change* event as well as a periodic *timeupdate* event.

query(): The query operation returns a vector representing the current state of the motion. This vector includes position, velocity, and acceleration, as well as a timestamp. For instance, if a query returns position 4.0 and velocity 1.0 and no acceleration, a new query one second later will return position 5.0.

```
var v = timingObject.query();
console.log("pos:" + v.position);
console.log("vel:" + v.velocity);
console.log("acc:" + v.acceleration);
```

Listing 2 Querying the timing object to get a snapshot vector.

update(vector): The update operation accepts a vector parameter specifying new values for position, velocity, and acceleration. This initiates a new movement for the motion. For instance, omitting position implies that the current position will be used. So, an update with velocity 0 pauses the motion at the current position.

```
// play, resume
timingObject.update({ velocity: 1.0 });
// pause
timingObject.update({ velocity: 0.0 });
// jump and play from 10
timingObject.update({ position: 10.0, velocity: 1.0});
// jump to position 10, keeping the current velocity
timingObject.update({ position: 10.0 });
```

Listing 3 Updating the timing object.

timeupdate event: For compatibility with existing HTML5 media elements and an easy way to update graphical elements, a *timeupdate* event is emitted periodically.

change event: Whenever a motion is updated, event listeners on the timing object (i.e., media components) will immediately be invoked. Note that the *change* event

is not emitted periodically like the *timeupdate* event of HTML5 media elements. The change event signifies the start of a new movement, not the continuation of a movement.

```
timingObject.on("change", function (e) {
  var v = motion.query();
  if (v.velocity === 0.0 && v.acceleration === 0.0) {
    console.log("I'm not moving!");
  } else {
    console.log("I'm moving!");
  }
});
```

Listing 4 Monitoring changes to the motion through the change event.

17.7.2 Programming with Motions

Using motions: Motions are resources used by Web applications, and the developer may define as many as required. What purposes they serve in the application is up to the programmer. If the motion should represent media offset in milliseconds, just set the velocity to 1000 (advances the position of the motion by 1000 milliseconds per second). Or, for certain musical applications it may be practical to let the motion represent beats per second.

Timing converters: A common challenge in media synchronization is that different sources of media content may reference different timelines. For instance, one media stream may have a logical timeline starting with 0, whereas another is timestamped with epoch values. If the relation between these timelines is known (i.e., *relative skew*), it may be practical to create a skewed timing object for one of the media components, connected to the motion. This is supported by *timing converters*. Multiple timing converters may be connected to a motion, each implementing different transformations such as scaling and looping. Timing converters may also be chained. Timing converters implement the *timing object API*, so media components cannot distinguish between a timing object and a timing converter. A number of timing converters are implemented in the Timingsrc programming model [3].

Flexibility: The mathematical nature of the motion concept makes it flexible; yet for a particular media component some of this flexibility may be unnecessary or even unwanted. For instance, the HTML5 media player will typically not be able to operate well with negative velocities, very high velocities, or with acceleration. Fortunately, it does not have to. Instead, the media player may define alternative modes of operation as long as the motion is in an unsupported state. It could show a still image every second for high velocity or simply stop operation altogether (e.g., black screen with relevant message). Later, when motion reenters a supported state, normal operation may be resumed for the media player.

17.7.3 Online Motion

The timing object API is particularly designed to mediate access to online motions, as illustrated in Fig. 17.5. *Update* operations are forwarded to the online motion, and will not take effect until notification is received from the online motion. After this, a *change event* will be emitted by the timimg object. In contrast, *query* is a local (and cheap) operation. This ensures that media components may sample the motion frequently if needed. So, through the Timing Object API, online motions are made available to Web developers as local objects. Only the latency of the update operation should be evidence of a distributed nature.

To support this abstraction, precise, distributed *motion synchronization* is required. In particular, the internal *clock* of the motion must be precisely synchronized with the clock at the online motion server. Synchronized system clocks (e.g., *Network Time Protocol (NTP)* [27] or *Precision Time Protocol (PTP)* [13]) are generally not a valid assumption in the Web domain. As a consequence, an alternative method of estimating a shared clock needs to be used, for example by sampling an online clock directly. In addition, low latency is important for user experiences. Web agents should be able to join synchronization quickly on page load or after page reload. To achieve this, joining agents must quickly obtain the current vector and the synchronized clock. For some applications, the user experience might also benefit from motion updates being disseminated quickly to all agents. Web agents should also be able to join and leave synchronization at any time, or fail, without affecting the synchronization of other agents. Motion synchronization is discussed in more detail in [6].

InMotion is a hosting service for online motions, built by the Motion Corporation [28]. A dedicated online service supporting online motions is likely key to achieving nonfunctional goals, such as high availability, reliability, and scalability. Evaluation of motion synchronization is presented in Sect. 17.9.

Finally, the timing object API emphasizes an attractive programming model for multi-device media applications. In particular, by making online motions available under the same API as local motions (see Sect. 17.7.1), media components may be used in single page as well as multi-device media experiences, without modification. Also, by hiding the complexity of distributed motion synchronization, application

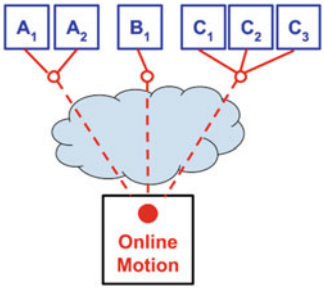


Fig. 17.5 (Fig. 17.2 repeated for convenience.) Timing objects (red unfilled circles) mediate access to online motion. Timing objects may be shared by independent media components within the same browsing context

developers may focus on building great media components using the timing object API. As such, the timing object API provides much needed separation of concern in multi-device media.

17.7.4 Synchronizing Audio and Video

On the Web, playback of audio and video is supported by HTML5 media elements [19]. Synchronizing media elements relative to a timing object means that the *currentTime* property (i.e., media offset) must be kept equal to the position of the timing object at all times, at least to a good approximation. The basic approach is to monitor the media element continuously and try to rectify whenever the synchronization error grows beyond a certain threshold. For larger errors *seekTo* is used. This is typically the case on *page load*, or after timing object *change* events. Smaller errors are rectified gradually by manipulating playbackrate. *SeekTo* is quite disruptive to the user experience, so support for variable playbackrate is currently required for high-quality synchronization.

MediaSync is a JavaScript library allowing HTML5 media elements to be synchronized by timing objects. The MediaSync library targets usage across the most common Web browsers, so it is not optimized for any particular scenario. Though synchronization of HTML5 media is simple in theory, it involves a few practical challenges, as indicated in Sect. 17.6.1. First, *currentTime* is only a coarse representation of the media offset, and it fluctuates considerably when compared to the system clock. The MediaSync library solves this by collecting a backlog of samples, from which a value of currentTime can be estimated. Building up this backlog requires some samples, so it may take more than a second for estimates to stabilize. Another issue relates to unpredictable time consumption in media control operations. In particular, seekTo(X) will change currentTime to X, but it will require a non-negligible amount of time to do so. In the context of synchronization, it aims for a fixed target when it should be aiming for a moving target. The MediaSync library compensates for this by overshooting the target. Furthermore, in order to overshoot by the correct amount, the algorithm collects statistics from every seekTo operation. Surprisingly perhaps, this strategy works reasonably well. Evaluation for the MediaSync library is presented in Sect. 17.9.

17.7.5 Synchronizing Timed Data

Synchronization of timed data using timing objects is an important challenge. Timed data such as subtitles, tracks, scripts, logs, or time series typically include items tied to *points* or *intervals* on the timeline. Synchronization then involves activating and deactivating such items at the correct time, in reference to a timing object. To simplify programming of media components based on timed data, a generic *Sequencer*

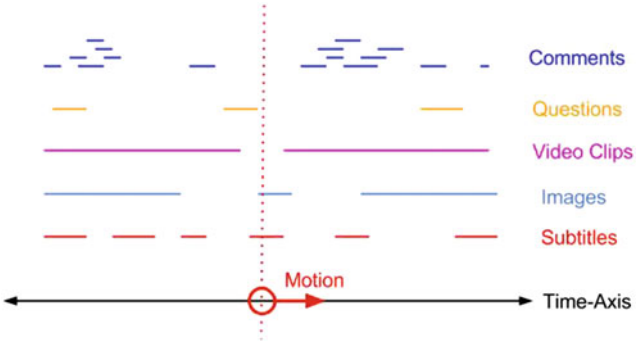


Fig. 17.6 Sequencing five data sources of timed data, with items tied to intervals on the timeline. Motion along the same timeline defines which items are active (vertical dotted line) and precisely when items will be activated or deactivated

is defined (Fig. 17.6). The sequencer is similar to the HTML5 track element [22], but is directed by the timing object instead of a HTML5 media element [19]. Web developers register cues associated with intervals on the timeline and receive event upcalls whenever a cue is activated or deactivated. The sequencer fully supports the timing object, including skipping, reverse playback, and acceleration. It may be used for any data type and supports dynamic changes to cues during playback.

The sequencer is implemented as a JavaScript library and made available as part of the open-source *Timingsrc* [3] programming model (see Sect. 17.10). In the interest of precisely synchronized activation and deactivation and low CPU consumption, the sequencer implementation is not based on frequent polling. Instead, the deterministic nature of the timing object allows events to be calculated and scheduled using *setTimeout*, the timeout mechanism available in Web browsers. Though this mechanism is not optimized for precision, Web browsers may be precise down to a few milliseconds. The sequencer is presented in further detail in [4].

17.8 Flexibility and Extensibility

Modern multimedia increasingly demands high flexibility and extensibility. This is driven by a number of strong trends: device proliferation, new sensors, new data types (e.g., sensor data, 3D, 360 degree video), multiple data sources, live data, personalization, interactivity, responsiveness, and multi-device support. On top of all this, there are also rising expectations to UI design, integration with social networks, and more.

In an attempt to meet such demands, new features have been added to media frameworks allowing programmers to customize the media player to a larger extent. For example, the Flash [1] framework has grown increasingly feature-rich over time, even having partially overlapping features with the Web platform itself.

Media Source Extensions (MSE) [24] in HTML5 provide a way to manipulate the video stream client-side. It is also common for media players to expose events and timed cues, allowing custom functionality to be implemented in application code. The text track system of HTML5 is an example of this. MPEG-4 [30] adds support for synchronization and composition of multiple media streams, including timed data such as graphical objects (2D and 3D). In particular, the MPEG-4 systems part [31] defines an architecture for media clients (terminals) integrating a variety of media formats, delivery methods, interactivity, and rendering.

In short, the need for extensibility has driven a development toward standardization of new data formats and features, leading media players to become increasingly sophisticated, yet also more complicated and heavyweight. We call this the *big player* approach to flexibility and extensibility in multimedia.

17.8.1 Multiple Small Players

The motion model presents an attractive alternative to the big player approach. The key idea is that a big player may be replaced by multiple smaller players, with precisely synchronized playback. As illustrated in Fig. 17.7, the flexibility of the motion model allows a variety of specialized media components to be coupled together, forming custom and complex media experiences from simpler parts. We use the term Composite Media [2] for media experiences built in this way.

17.8.2 Dedicated Media Components

The motion model typically encourages a pattern where each media component is dedicated to solving a small and well-defined challenge: Given timed data and a motion, the media component must generate the correct presentation at all times. Such custom media components are implemented in application code, and an appropriate delivery method may be selected for the particular media type and the task at hand. This way, application-specific data formats may be integrated into a presentation, as well as standardized formats. Importantly, timed data sources may be dynamic and live, implying that presentations may interact directly with live backend systems and update their presentations during playback.

Media components may also be dedicated with respect to UI. For instance, a single media component may implement interactive controls for the motion, thereby relieving other media components from this added complexity. This encourages a pattern where media components are designed for specific roles in an application, e.g., controllers, viewers, and editors, and combined to form the full functionality. Of course, the fact that these media components are independent may be hidden for end users with appropriate layout and styling, giving the impression of a tightly integrated product. In any case, dedicated media components may be reusable across

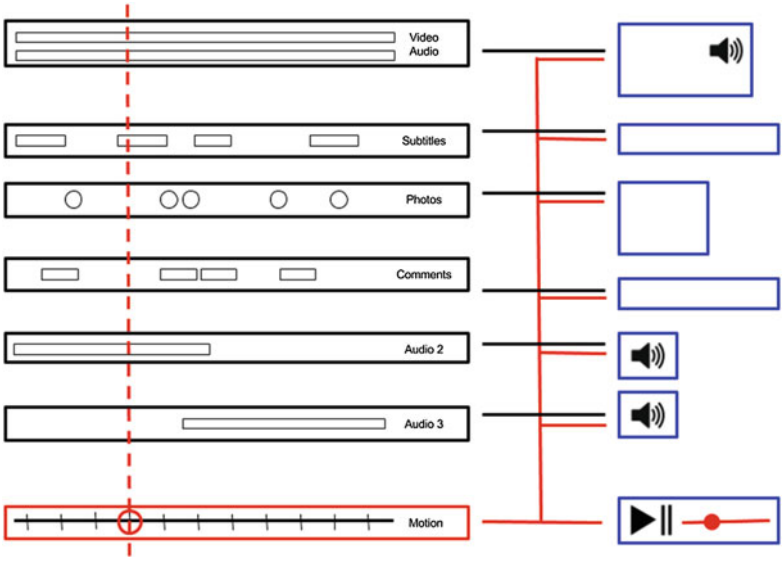


Fig. 17.7 A single media experience made from multiple media components (blue), possibly distributed across multiple devices. Each media component is connected to motion (red) and a source of timed data (black). There are different types of timed data: an AV container, a subtitle track, photographs, comments, and two extra audio tracks. The motion defines the timeline for the presentation, and timed data is mapped to this timeline by each media component. Since all the media components are connected to the same motion, they will operate in precise synchrony. One particular media component (bottom media element) provides interactive controls for the presentation and connects only with motion

different views, applications, devices, or data sets, as long as APIs to data model and motions remain unchanged.

17.8.3 Flexible Coupling

The motion model allows modularity and flexibility by loose coupling of media components. In fact, media components may be coupled only indirectly through shared motions and shared data sources. This ensures that media components can be added or removed dynamically during playback, or even fail, without disrupting the rest of the presentation. This flexibility is also valuable in development, as media components may be coded and tested in isolation or with other components. New components may always be added without introducing any additional increase in complexity, naturally supporting an incremental development process. Also, the model does not impose restrictions on how motions and timed data sources are connected with media components. A single data source may be shared between multiple media components, or conversely, a single media component may use multiple data sources.

The same flexibility goes for motions. There might be multiple aspects of timing and control in an application, requiring multiple motions to be shared between media components.

17.8.4 Client-Side Synthesis

Client-side synthesis is core design principle of the Web platform and central to key properties such as flexibility, extensibility, reusability, and scalability. This principle may now be fully exploited in the context of timed media applications. With the motion model, timed media experiences may be synthesized in real time within the browsing context (client-side), by independent media components working directly on live data sources and motions.

Interestingly, client-side synthesis is not the established approach to linear media, not even in the Web context. With media frameworks such as Flash [1] or MPEG-4 [30], media is typically assembled in a media file or a media container, before being downloaded or streamed to a client-side media player. Essentially, this is server-side synthesis (and client-side playback). While server-side synthesis may have certain advantages (e.g., robustness and simplicity), the disadvantages are also evident. By assembling data within media files and container formats, data is decoupled from its source and effectively flattened into an immutable copy. Introduction of new media types may also be inconvenient, as this must be addressed through standardization of new media and container formats, and support must be implemented by media players. This may be a time-consuming process. That said, server-side synthesis may still be an appropriate choice for a wide range of media products.

Importantly though, in the motion model the choice between client-side synthesis and server-side synthesis is left to application programmers. Established container-based media frameworks are still usable, provided only that the framework can be integrated and controlled by external motion. Ideally, this integration should be performed internally by the framework. If this is done, frameworks can easily be used in conjunction with native media elements, other frameworks or components that support external motion. If not, integration may also be done externally, subject to the limitations of the framework API. In any case, the motion model relieves media frameworks from the challenge of doing everything and highlights their value as dedicated, reusable components.

17.9 Evaluation

The evaluation is concerned with feasibility of the motion model and simplicity for Web developers.

17.9.1 Motion Synchronization

We have used motion synchronization for a wide range of technical demonstration since 2010. An early evaluation of the research prototype is discussed in the paper titled *The Media State Vector* [6]. Though the interpretations of the experiments are conservative, early findings indicated that motion synchronization could provide frame rate levels of accuracy (33 milliseconds). A few years later, a production ready service called *InMotion* was built by spin-off company Motion Corporation [28]. With the introduction of WebSockets [39], results improved significantly. Synchronization errors are in the order of a few milliseconds on all major browsers and most operating systems (including Android). Typically we observe 0-1 milliseconds errors for desktop browsers, compared to a system clock synchronized by NTP. The *InMotion* service has also been running continuously for years, supporting a wide range of technical demonstrations, at any time, at any place, and across a wide range of devices. As such, the value of a production grade online service is also confirmed.

Furthermore, the precision of motion synchronization degrades well with poor network conditions. For instance, experiments with video synchronization in *EDGE* connectivity *Enhanced Data rates for GSM Evolution* have not been visibly worse, except for longer update latency. In this instance, video data was fetched from local files. Conferences are also notorious hot spots for bad connectivity. In these circumstances, availability of media data consistently fails before synchronization.

17.9.2 Synchronization of HTML5 Media Elements

Two technical reports [8, 9] document the abilities and limitations of HTML5 media elements with respect to media synchronization, as well the quality of synchronization achieved by the *MediaSync* library (Fig. 17.8). Synchronization errors of about 7 milliseconds are reported for both audio and video, on desktops, laptops, and high-end smartphones. This corresponds to echoless audio playback. Smartphones and embedded devices such as ChromeCast can be expected to provide frame accurate synchronization.

These results have been consistently confirmed by day-to-day usage over several years. The user experience of multi-device video synchronization is also very good, to the point that errors are hardly visible, as demonstrated by this video [40]. Echoless synchronization with the MediaSync library may also produce various audio effects, like failing to hear one audio source, until volume levels are changed and only the other audio source can be heard. Since these effects are also achieved across browser types and architectures, this is a strong indication that external timing is feasible and already at a useful level.

Synchronization has also been maintained for hours and days at end, without accumulated errors. Loading speeds are also acceptable. Even though the MediaSync library requires about 3 seconds to reach echoless, the experience is perceived as

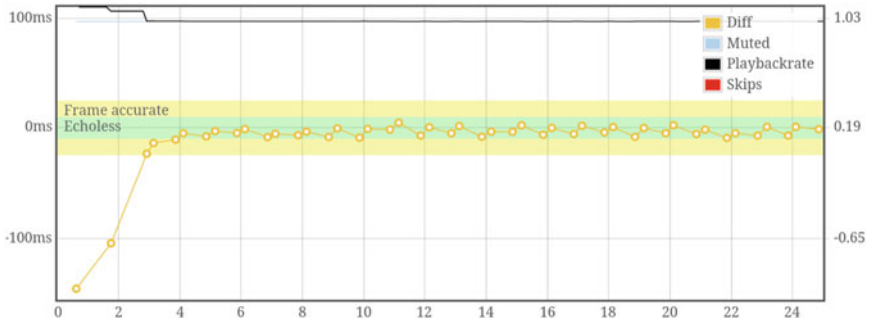


Fig. 17.8 The figure illustrates an experiment with video (mp4) synchronization on Android using Chrome browser. The plot shows currentTime compared to the ideal playback position defined by motion. The X-axis denotes the timeline of the experiment (seconds). The left Y-axis denotes difference *Diff* (milliseconds) between currentTime and motion. The green band (echoless) is ±10 milliseconds and the yellow (frame accurate is) ±25 milliseconds. This is achieved using variable playbackrate. No skips were performed in this experiment. The right Y-axis denotes the value of playbackrate (seconds per second). The media element was muted until playbackrate stabilized

acceptable much before this. A variety of video demonstrations have been published at the Multi-device Timing Community Group Website [32].

Though echoless synchronization is generally achievable, a lack of standardization and common tests makes it impossible to provide any guarantees. The experience might also be improved or become broken across software updates. To be able to support echoless synchronization reliably across browsers and devices, standards must include requirements for synchronization, and testing suites must be developed to ensure that those requirements are met. Ideally though, media synchronization should be implemented natively in media elements.

17.9.3 Summary

Interestingly, the results for motion synchronization and HTML5 media synchronization are well aligned with current limitations of the Web platform. For instance, the precision of timed operation in JavaScript is about 1 milliseconds, and a 60 Hz screen refresh rate corresponds to 16 milliseconds. Furthermore, these results also match limitations in human sensitivity to synchronization errors.

Finally, programming synchronized media experiences in the motion model is both easy and rewarding. In our experience, motions and sequencers are effective thinking tools as well as programming tools. A globally synchronized video experience essentially requires three code statements.

With this, we argue that the feasibility of the motion model is confirmed. It is also clear that synchronization errors in online synchronization are currently

dominated by errors in synchronization in HTML5 media elements. Future standardization efforts and optimizations would likely yield significant improvements.

17.10 Standardization

The Web is widely regarded as a universal multimedia platform although it lacks a common model for timing and media control. The motion model promises to fill this gap and indicates a significant potential for the Web as a platform for globally synchronized capture and playback of timed multimedia. To bring these possibilities to the attention of the Web community, the motion model has been proposed for Web standardization. The Multi-device Timing Community Group (MTCG) [32] has been created to attract support for this initiative. The MTCG has published the draft specification for the TimingObject [7]. It has also published Timingsrc [3], an open source JavaScript implementation of the TimingObject specification, including timing objects, timing converters, sequencers, and the MediaSync library.

Though the ideas promoted by the MTCG have been received with enthusiasm by members within the W3C and within the wider Web community, at present the MTCG proposal has not been evaluated by the W3C.

17.11 Conclusions

We have explored media synchronization between heterogeneous media components and highlighted the need for temporal interoperability on the Web platform. While internal timing is the popular approach to Web-based media, external timing is the key to temporal interoperability.

This chapter provided an introduction to external timing as well as the media model and the programming model that follows from this approach. By focusing on modularity, loose coupling, and client-side synthesis, this media model is well aligned with key design principles of the Web, thereby fully extending the flexibility and extensibility of the Web platform to timed Web applications. Equally important, the external timing approach promises precise distributed playback and media orchestration, enabling precise timing and control also in multi-device Web-based media experiences.

To encourage temporal interoperability on the Web platform, the W3C Multi-device Timing Community Group (MTCG) [32] advocates standardization of the timing object [7] as a common interface to external timing and control. Using the external timing approach, we have demonstrated that the Web is already a strong platform for timed, multi-device media, though it was not designed for it. With standardization it will become even better, likely unleashing a new wave of Web-based creativity across a variety of application domains.

Finally, as the external timing approach to media synchronization works on the Web, it may also be ported to other native applications in the IP environment. This provides a simple mechanism for making distributed media from a mixture of native and Web-based media components.

References

1. Adobe: Adobe Flash. https://www.adobe.com/products/flashruntimes.html
2. Arntzen, I.M., Borch, N.T.: Composite Media, a new paradigm for online media. In: 2013 NEM Summit (Networked Electronic Media), NEM Summit '13, pp. 105–110. Eurescom (2013). http://nem-initiative.org/wp-content/uploads/2015/06/2013-NEM-Summit_Proceedings.pdf
3. Arntzen, I.M., Borch, N.T.: Timingsrc: A programming model for timed Web applications, based on the Timing Object. In: Precise Timing, Synchronization and Control Enabled for Single-Device and Multi-device Web Applications (2015). http://webtiming.github.io/timingsrc/
4. Arntzen, I.M., Borch, N.T.: Data-independent sequencing with the timing Object: a JavaScript sequencer for single-device and Multi-device Web Media. In: Proceedings of the 7th International Conference on Multimedia Systems, MMSys'16, pp. 24:1–24:10. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2910017.2910614
5. Arntzen, I.M., Borch, N.T., Daoust, F., Hazael-Massieux, D.: Multi-device Linear Composition on the Web; enabling multi-device linear Media with HTMLTimingobject and shared motion. In: Media Synchronization Workshop (MediaSync) in conjunction with ACM TVX 2015. ACM (2015). https://sites.google.com/site/mediasynchronization/Paper4_Arntzen_webComposition_CR.pdf
6. Arntzen, I.M., Borch, N.T., Needham, C.P.: The media state vector: a unifying concept for multi-device media navigation. In: Proceedings of the 5th Workshop on Mobile Video, MoVid'13, pp. 61–66. ACM, New York, NY, USA (2013). https://doi.org/10.1145/2457413.2457427
7. Arntzen, I.M., Daoust, F., Borch, N.T.: Timing Object; Draft community group report. http://webtiming.github.io/timingobject/ (2015)
8. Borch, N.T., Arntzen, I.M.: Distributed Synchronization of HTML5 Media. Technical report 15, Norut Northern Research Institute (2014)
9. Borch, N.T., Arntzen, I.M.: Mediasync Report 2015: Evaluating timed playback of HTML5 Media. Technical report 28, Norut Northern Research Institute (2015)
10. Document Object Model (DOM) Level-1 (1998). https://www.w3.org/TR/REC-DOM-Level-1/
11. Document Object Model (DOM) Level-2 (2000). https://www.w3.org/TR/DOM-Level-2/
12. Document Object Model (DOM) Level-2 Events (2000). https://www.w3.org/TR/DOM-Level-2-Events/
13. Eidson, J., Lee, K.: IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In: Sensors for Industry Conference, 2002. 2nd ISA/IEEE, pp. 98–105. IEEE (2002)
14. Google (2017). https://www.google.com
15. High Resolution Time (2012). https://www.w3.org/TR/hr-time-1/
16. High Resolution Time Level 2 (2016). https://www.w3.org/TR/hr-time-2/
17. HTML 3.2 Reference Specification (1997). https://www.w3.org/TR/REC-html32
18. HTML5 (2014). https://www.w3.org/TR/html5/
19. HTML5 Media Elements (2012). http://dev.w3.org/html5/spec-preview/media-elements.html
20. HTML5.1 (2016). https://www.w3.org/TR/html51/
21. HTML5 Media Controller (2014). https://dev.w3.org/html5/spec-preview/media-elements.html

22. HTML5 Text Track (2012). http://dev.w3.org/html5/spec-preview/media-elements.html#text-track
23. Media Capture and Streams (2016). https://www.w3.org/TR/mediacapture-streams/
24. Media Source Extensions (2016). https://www.w3.org/TR/media-source/
25. MediaStream Recording (2017). https://www.w3.org/TR/mediastream-recording/
26. Microsoft (2017). https://www.microsoft.com/
27. Mills, D.L.: Internet time synchronization: the network time protocol. IEEE Trans. Commun. **39**(10), 1482–1493 (1991). https://doi.org/10.1109/26.103043
28. Motion Corporation. http://motioncorporation.com
29. Mozilla (2017). https://www.mozilla.org
30. MPEG-4. http://mpeg.chiariglione.org/standards/mpeg-4
31. MPEG-4 Systems (2005). http://mpeg.chiariglione.org/standards/mpeg-4/systems
32. Multi-device Timing Community Group (2015). https://www.w3.org/community/webtiming/
33. RealPlayer (2017). http://www.real.com/
34. Scalable Vector Graphics (SVG) 1.1 (2011). https://www.w3.org/TR/SVG/
35. Synchronized Multimedia Integration Language (SMIL) 1.0 Specification (1998). https://www.w3.org/TR/1998/REC-smil-19980615/
36. SMIL 3.0 Synchronized Multimedia Integration Language (2008). http://www.w3.org/TR/REC-smil/
37. Smil Animation (2001). https://www.w3.org/TR/smil-animation/
38. Twitter (2017). https://twitter.com/
39. The Web Socket Protocol (2011). https://tools.ietf.org/html/rfc6455
40. Video synchronization by Motion Corporation (2015). https://youtu.be/lfoUstnusIE
41. W3C DOM4 (2015). https://www.w3.org/TR/dom/
42. Web Animations (2016). http://www.w3.org/TR/web-animations/
43. Web Audio API (2015). https://www.w3.org/TR/webaudio/
44. Web Workers (2015). https://www.w3.org/TR/workers/
45. WebGL (2017). https://www.khronos.org/webgl/
46. WebRTC Real-time Communication Between Browsers (2017). https://www.w3.org/TR/webrtc/

Chapter 18
Media Synchronisation for Television Services Through HbbTV

M. Oskar van Deventer, Michael Probst and Christoph Ziegler

Abstract Media synchronisation is getting renewed attention with ecosystems of smart televisions and connected devices enabling novel media consumption paradigms. Social TV, hybrid TV and companion screens are examples that are enabling people to consume multiple media streams on multiple devices together. These novel use cases place a number of demands on the synchronisation architecture. The systems for media synchronisation have to cope with delay differences between various distribution channels for television broadcast (terrestrial, cable, satellite) and Internet-delivered streaming media. Also they need to handle different content formats in use. Broadcasters have started using proprietary solutions for over-the-top media synchronisation, such as media fingerprinting or media watermarking technologies. Given the commercial interest in media synchronisation and the disadvantages of proprietary technologies, consumer equipment manufacturers, broadcasters, as well as telecom and cable operators have started developing a new wave of television products, services and international standards that support media synchronisation from multiple sources. This chapter provides an overview of media synchronisation in a television context as specified the Hybrid Broadcast Broadband Television (HbbTV) specification version 2 and based upon specifications by the Digital Video Broadcasting (DVB) group Companion Screens and Streams (CSS). In addition, we discuss solutions compliant with legacy HbbTV devices. Use cases include synchronisation of audio, video and data streams from multiple sources composed on a TV or on multiple devices including other consumer devices like smartphones.

Keywords Interactive TV · HbbTV · Hybrid Broadcast Broadband Television Media synchronisation · Inter-device synchronisation · Multi-stream synchronisation · Broadcast networks

M. O. van Deventer (✉)
TNO, Anna van Buerenplein 1, 2595 DA Den Haag, Netherlands
e-mail: oskar.vandeventer@tno.nl

M. Probst · C. Ziegler
IRT, Floriansmuehlstraße 60, 80939 Munich, Germany
e-mail: michael.probst@irt.de

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_18

18.1 Introduction

18.1.1 Media Synchronisation

Media synchronisation is especially relevant whenever two or more associated media streams are played back together. The classic example is synchronisation of audio and video for a television broadcast to achieve lip synchronisation (lip-sync). More recent examples are social television (TV), hybrid TV and services for companion screens. Social TV, also known as "watching apart together", has multiple users watching the same TV broadcast while communicating with each other by voice, chat or other social media. Hybrid TV converges multiple media streams from different channels (broadcast, Internet) into one single TV programme experience (e.g. broadcast video with subtitles or alternative audio received via the Internet). Services for companion screens like tablets and smartphones provide user interaction or media consumption on tablet devices associated with a television broadcast (e.g. a play-along quiz, alternative audio or alternative camera views).

Requirements on synchronicity differ per use case. Social TV is the least demanding case. If there is no audio crosstalk, users would not notice delay differences of less than a second, and often they do not even notice a four-second difference [1]. Hybrid TV is the strictest case, since lip-sync requires audio and video to be synchronised within 40 ms [2]. A non-lip-sync companion screen case may be between those two extremes, adding the challenge of achieving synchronisation between two separate devices where communication latency must be compensated for.

Even the least demanding requirement cannot be met by today's media delivery technologies. There can be up to six seconds difference in delivery of a single broadcast channel in a single country via different providers [3]. Transcoding buffers are a major contribution to those delay differences. Transmission delays are also significant. For example, a single satellite hop introduces over a quarter of a second delay due to the non-infinite speed of light. Internet delivery using Content Delivery Networks (CDNs) is by far the slowest delivery technology. When using adaptive streaming, it can easily take thirty seconds to perform all required delivery steps, from transcoding and segmentation to segment buffering at the media player client. A recent test showed a delay of seventy-two seconds between a UK broadcaster's origination of a television channel and its delivery via the Internet outside the UK [3].

Broadcasters have started using over-the-top (OTT) media synchronisation technologies based on audio fingerprinting or audio watermarking for offering synchronised companion screen content, as these technologies are relatively easy to deploy, even in the absence of standards. However, they fail when the audio level is low or if there is background sound in the viewing environment, and considerable confusion may occur when a clip from a programme is being reused in another programme. Any system must make a compromise between factors such as recognition speed, robustness and perceptibility of any changes to the audio or

ability to discriminate across a volume of audio material [4]. Both fingerprinting and watermarking poorly handle user interactions like pause, seek, rewind and fast forward. Another concern is cost. There is the cost for changing the workflow to get the watermark into the audio of a broadcast before the encoder. Licensing fees for commercial solutions typically scale with the number of channels, the amount of content on those services being watermarked and/or the amount of activity from client applications (e.g. searches of an audio fingerprinting database). Finally, the lack of standards may result in high vendor switching costs.

Media synchronisation has re-emerged as an active field of standardisation during the last few years [5]. A set of complementary standards on media synchronisation was produced by various standards bodies [6]. This chapter overviews a set of technologies for media synchronisation in the context of Hybrid Broadcast Broadband Television, as well as relevant standards. This includes DVB-CSS which is part of version 2 of the HbbTV specification and approaches based on time information and additional signalling which can be accessed via application programming interfaces (APIs) also with implementations of older versions of the HbbTV specification, like DSM-CC stream events.

18.1.2 HbbTV—Evolution and Market Role

HbbTV—Hybrid Broadcast Broadband Television—is an industry consortium [7], including consumer equipment manufacturers, broadcasters and operators, that publishes specifications and test materials for OTT broadband Internet services that can be linked with traditional broadcast services based on the requirements of its members.

The first release, HbbTV 1.0, was developed in a cooperation of French and German broadcasters with TV manufacturers and the satellite operator ASTRA in 2010. The majority of today's HbbTV services in Germany are based on this version of the specification. The first update, usually called HbbTV 1.5 and released in 2012, is the basis for many other deployments today, reaching from France to Australia and New Zealand. The main difference of these two versions is the support of adaptive streaming using MPEG DASH (Moving Picture Experts Group—Dynamic Adaptive Streaming over HTTP) and MPEG CENC (common encryption) for supporting various proprietary Digital Rights Management (DRM) systems efficiently.

With the latest version, HbbTV 2.0.1 published in 2016, broadcasters from the UK and Italy started to migrate to HbbTV from their previous MHEG (Multimedia and Hypermedia information coding Expert Group) [8] and DVB MHP (Multimedia Home Platform) [9] solutions, which have similar concepts as HbbTV but were based on different run-time environments which are not used in modern Internet times anywhere else. In contrast, HbbTV is based on standard Web browser technologies like Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript with extensions for including the DVB broadcast world.

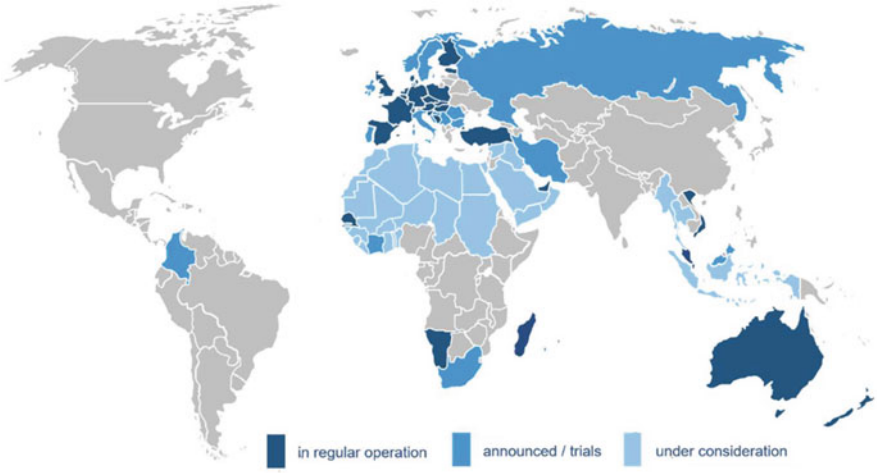


Fig. 18.1 Worldwide deployments of HbbTV, status May 2017 [7]

The map in Fig. 18.1 shows where HbbTV is deployed or considered for market introduction at the time of writing this text.

HbbTV provides a technical platform for broadcasters to use broadband Internet for distributing additional content as well as for providers of Internet-only services. Technically, there are two types of HbbTV applications.

Broadcast-related applications are signalled as part of a broadcast service (television channel) in an Application Information Table (AIT) [10]. Such an application runs in the context of this service and can include the video and audio components of this service, e.g. to provide an interactive overlay for adding a game to a TV programme. The application data, i.e. HTML pages etc., can be included in the broadcast in a cyclic transmission using the object carousel [10] or simply delivered from a Web server with HTTP(S).

Broadcast-independent applications are not started in the context of a broadcast service, but they can use all broadband features of HbbTV. Typical examples are video-on-demand applications. If a broadcast-independent application wants to use broadcast content, the provider of the broadcast has to reference the application in the AIT of the respective broadcast service. This way such an application can transition to a broadcast-related state. The AIT signal ensures there is a commercial agreement with the broadcaster before third parties can integrate broadcast streams into their HbbTV applications. With HbbTV version 1.0 and 1.5, broadcast-independent applications were not widely used. Since HbbTV 2.0 it is possible to launch such applications via the home network, e.g. from a smartphone application, which offers new opportunities for third parties to make use of HbbTV.

Beside the normal evolution of the specification, HbbTV started to define a set of specifications which are complementary to the main specification and fulfil specific

requirements. This includes Internet Protocol television (IPTV) services and operator applications.

The HbbTV platform today is mainly used to offer content produced for broadcast also as on-demand via broadband in a standardised environment. Broadcasters signal several applications on every service such as programme guide, catch-up TV and news applications which include content originally created for teletext, enhanced by graphics and video footage. For commercial broadcasters, HbbTV offers new opportunities to sell targeted advertising. This can be unrelated to the actual broadcast content, e.g. when the user switches to the broadcast channel, or even synchronised, e.g. for linking a micro Web page of an advertisement customer with his ad clips during the standard ad breaks. Means to synchronise HbbTV applications are discussed later in this chapter.

New service types that will be possible with HbbTV 2.0 are mainly based on the new companion screen APIs, allowing applications to discover TV devices or special companion screen applications in the home network, start and communicate with applications and to synchronise media presentation from multiple streams on multiple devices. This enables service providers to create true multi-screen applications, which of course is the main topic of this chapter.

In the following sections, the different aspects for media synchronisation using HbbTV and related standards are presented, and multiple approaches for implementing media synchronisation are discussed.

18.1.3 Use Cases

HbbTV enables use cases for synchronisation which fall into two domains. The first is synchronisation of media presentations on the TV device, which we refer to as intra-device synchronisation. The second is synchronisation of media presentations on the TV and a companion screen device, which we refer to as inter-device synchronisation.

We distinguish between two types of intra-device synchronisation. The first type is App-to-A/V synchronisation, which refers to scenarios where HTML-based content is synchronised to the broadcast content. This can be used, for example, to present interactive information graphics alongside a football match. Another example is pictured in Fig. 18.2, where a popup notifies a viewer of additional information on a product, announced during an advertisement clip. The user can query the additional information by pressing the red button on the remote.

The second type of intra-device synchronisation is multi-stream synchronisation. This refers to situations where distinct media streams, potentially transmitted via different distribution channels, are synchronised on the TV. For example, a video received via broadcast with an audio or video stream received via Internet. This allows, for example, offering access services, like on-demand sign-language interpreters (see Fig. 18.3) or on-demand audio description tracks. Further uses include presentation of a second video with an alternative camera angle, for

Fig. 18.2 Synchronised HbbTV application accompanying an advertisement clip. A flash notification invites the viewer to press the red button on their remotes to call further information on the advertised product. HbbTV app built by Teveo

Fig. 18.3 IP video stream with a sign-language interpreter overlaying a news show

example, for sports content, or playback of an audio stream with an alternative language as opposed to the one transmitted in the broadcast stream.

Potential applications for inter-device synchronisation are manifold. They include, for example, interactive play-along games, presentation of complementary or supplementary information, presentation of alternative camera angles, presentation of alternative audio or targeted advertisement. Figure 18.4 shows a

Fig. 18.4 Screenshot of a synchronised textbook application on a companion screen [11]

synchronised companion application proposed in [11]. The companion application is an interactive textbook presented in time with a video recording of a Shakespeare theatre play on the TV. Text lines in the script are highlighted based on the elocution of lines by characters in the scene.

18.1.4 Outline

The remainder of this chapter introduces different technologies which allow implementing the above-mentioned use cases. The chapter is structured as follows:

Section 18.2 describes the concepts behind DVB-CSS, a standard for inter-device synchronisation. It introduces the underlying concepts for wall-clock synchronisation and timeline modelling.

Section 18.3 describes APIs specified in HbbTV 2.0 for inter-device and multi-stream synchronisation as well as for access to media-timeline position information.

Section 18.4 introduces means defined in HbbTV 2.0 for bootstrapping distributed experiences which make use of inter-device synchronisation features. This includes APIs and protocols for automatic discovery of devices and protocol endpoints as well as means for automatic application launch. Also, it discusses advantages over solutions available for HbbTV 1.X TV devices.

Section 18.5 exemplifies approaches for integrating the different technologies introduced in the previous sections to implement the above-described use cases.

Section 18.6 describes means to synchronise HTML-based content with the broadcast video. This includes means introduced by HbbTV 2.0 (i.e. access to

media timelines) and means introduced by HbbTV 1.0 (stream events, EIT present following, receiver/time clock).

Section 18.7 concludes this chapter by summarising key insights.

18.2 Media Synchronisation Based on DVB-CSS

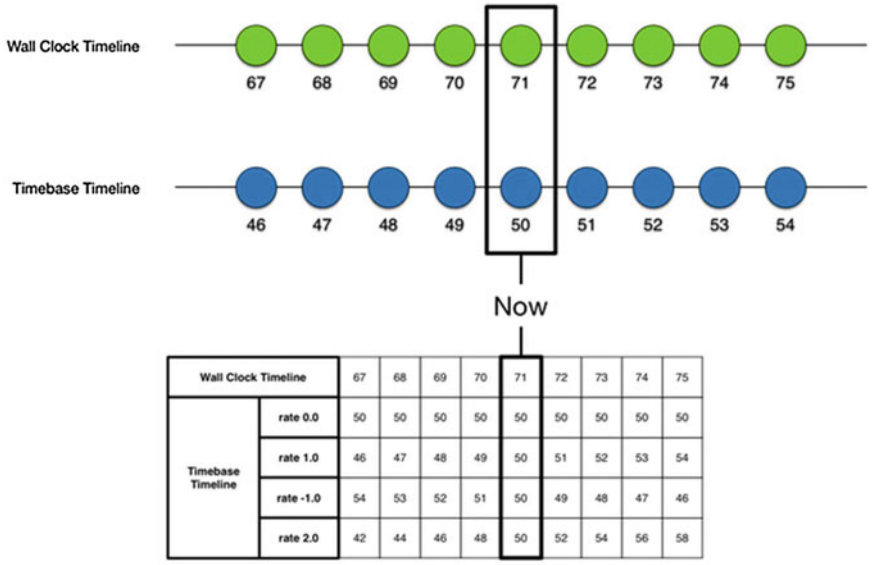
DVB-CSS is a standard [12] to synchronise a media stream (audio/video) on a companion device with a media stream on a television set. The need for media synchronisation is that different media streams take different paths, with different delays due to transmission, routing, encoding, decoding, rendering and other processing. Also, the clocks of the different media streams may have different time bases, with a different clock tick rate, offset, skew, drift and/or jitter. As a result, a video played out at the TV will typically be out of sync with audio played out from the companion device. Media synchronisation buffering is needed to delay some of the media streams in order of all media streams to get in sync with the most laggard one. The set of protocols specified by DVB-CSS enable the coordination of this media synchronisation.

The key concepts for media synchronisation are the wall-clock timeline and the timebase timeline, as specified in the introduction of DVB-CSS [12]. The wall-clock timeline is a timeline of a common (local) clock that advances steadily, and the playback of the timed media streams is timed accordingly to achieve a smooth presentation. To enable this, the media streams are adorned with their own timebase timeline, which is compared to the wall-clock timeline; see Fig. 18.5.

Extending from this basic model, Fig. 18.6 shows how the playback of two independent media streams can be controlled on two independent media players in a coordinated fashion.

The DVB-CSS architecture (Fig. 18.7) has one TV Device (a television (TV) or a set-top box (STB)) and one or more companion screen applications (CSAs) running on companion screen devices that are connected via a home network, typically Wi-fi. Both TV and CSA independently receive media streams from the broadcaster (not shown). The presentation of the media streams is synchronised by using a set of new protocols and a new Material Resolution Service (MRS). At least the companion screen, but possibly also the TV has a media synchronisation buffer that can delay the playout of the media at that device in order to achieve synchronised playout. The media synchronisation is (at least conceptually) separate from, e.g. the decoding buffer, the de-jittering buffer or a DASH segment buffer.

A typical media synchronisation scenario is as follows. The user tunes their TV to a broadcast service. The TV receives the broadcast service, which includes a broadcast stream and metadata for media synchronisation, including a content identification that identifies the current programme and the location (URL) of the MRS. The user pairs their companion device with the TV and starts a CSA. The TV provides the CSA with content identification and the MRS location via the content information and other information (CSS-CII) protocol.



Wall Clock Timeline		67	68	69	70	71	72	73	74	75
Timebase Timeline	rate 0.0	50	50	50	50	50	50	50	50	50
	rate 1.0	46	47	48	49	50	51	52	53	54
	rate -1.0	54	53	52	51	50	49	48	47	46
	rate 2.0	42	44	46	48	50	52	54	56	58

Fig. 18.5 Basic model of time-controlled playback. *Source* [12], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)

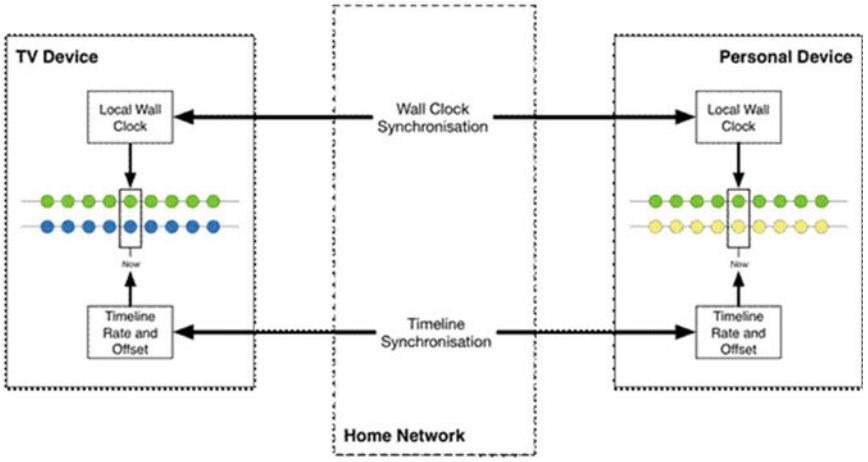


Fig. 18.6 Basic model of synchronising playback between devices. *Source* [12], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)

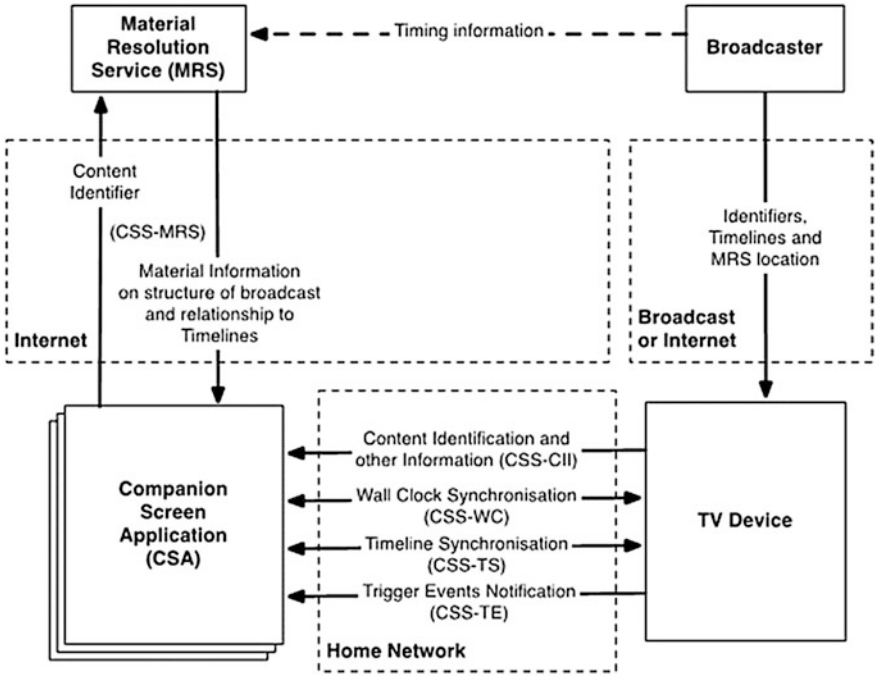


Fig. 18.7 DVB-CSS architecture. *Source* [12], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)

Next, the CSA queries the Material Resolution Server via the CSS-MRS protocol and it obtains material information that describes the structure of the broadcast, that is composition of materials and sub-materials such as programmes, sections within programmes and advertisements. It also describes the relationship between this structure and timelines. DVB-CSS supports several types of timelines, including MPEG transport stream presentation timestamp (PTS), ISO base media file format (ISOBMFF) Composition Time and time relative to the start of a period in an MPEG DASH presentation. It also supports the use of MPEG Timed External Media Information (TEMI) [13] as a timeline.

This combination of information from the TV and the MRS server enables the CSA to determine which streams it should present and how its timeline correlates to that of the media being presented on the TV. However, the CSA manages its own behaviour and is not directly controlled by the TV.

In parallel, the CSA obtains the IP addresses of the wall clock and the timeline synchronisation service endpoints in the TV via the CSS-CII protocol. The CSA synchronises its wall clock with the TV via the CSS-WC protocol. When the user starts a selected media stream on its companion screen, the CSA synchronises the stream's timeline with the timeline of the stream on the TV via the CSS-TS

protocol, as described below. The stream played out on the TV may be a broadcast service (television channel), but it may also be content received via the Internet (e.g. video-on-demand) or recorded content (e.g. personal video recorder). The CSA can also subscribe to trigger events (CSS-TE protocol) that are received by the TV from the broadcaster as part of the signalling within the broadcast stream.

The Wall-Clock Synchronisation protocol (CSS-WC) is a request-response UDP-based protocol that enables the client (CSA) to estimate a clock at a server (the TV) and measure and compensate for network round-trip delay. The protocol design is similar to the client/server mode of Network Time Protocol (NTP) [14], but significantly simplified. Although many devices implement NTP to set their system wide clocks, a CSA running on a companion device cannot always check if an NTP client process is functioning, or query the accuracy of clock synchronisation. Media synchronisation also does not require the shared clock to be with reference to absolute real-world time and can therefore avoid complexities such as leap seconds. Frame-accurate media synchronisation requires accuracy in the order of milliseconds and the chances of achieving this are improved if the protocol operates directly between the TV and CSA instead of via a hierarchy of intermediate servers on more distant network segments. Ultimately, it is the accuracy of the CSS-WC protocol that determines how accurate the media synchronisation can become. Experiments with CSS-WC [15] show that accuracies in the order of 3–20 ms can be achieved in realistic network conditions (Fig. 18.8).

The Timeline Synchronisation protocol (CSS-TS) is a WebSocket-based protocol that carries the timing information needed for coordination between the CSA and TV. Messages conveyed by this protocol describe the relationship between wall-clock time and timeline position. This enables the CSA to accurately estimate the current TV timeline position despite possible network transmission delays. The CSS-TS protocol architecture has synchronisation clients (SC) and a media synchronisation application server (MSAS). An SC measures the timing of the media playout and reports "earliest presentation timestamps" to the MSAS. Timestamps in DVB-CSS are the combination of media timestamps, i.e. a value on the media timeline (e.g. the PTS of a video frame), and the earliest wall-clock time at which it could have been presented. The delay by the media synchronisation buffer is subtracted in the calculation of earliest presentation timestamps, as the earliest presentation time applies to an empty media synchronisation buffer. As an earliest presentation timestamp is relative to the wall-clock time, the CSS-TS protocol is

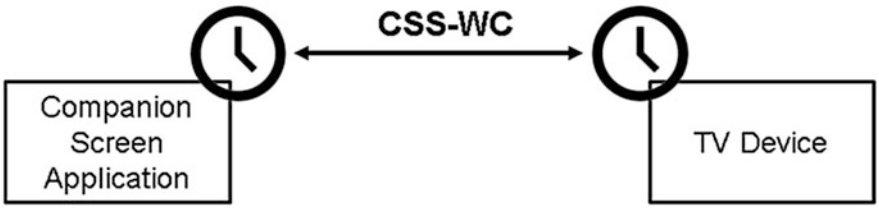


Fig. 18.8 DVB-CSS-WC protocol synchronises the wall clock at the CSA with the TV

insensitive to delays in its message exchange or processing, and only dependent on the accuracy of the wall clocks at both devices. An SC may also report actual and latest presentation timestamps. The latter is the latest wall-clock time at which an identified video frame or audio sample (identified by a media timestamp) could be presented, given the limited size of the media synchronisation buffer. The MSAS collects presentation timestamps from multiple SCs in CSAs and the TV, and it calculates and distributes control timestamps to the SCs. The SC uses a control timestamp to adjust the delay by the media synchronisation buffer, such that the actual presentation matches the control timestamp. As different SCs in different devices perform the same task based on the same control timestamps, timelines are getting realigned across the different devices.

Whereas DVB-CSS [12] specifies the CSS-TS protocol in detail, only part of it is mandatory. First of all, the MSAS function could be located anywhere in principle, but it would be typically integrated with the TV in practice. Also, broadcasters do not like dynamically adjusting the timing of the playout of a broadcast service at the TV. Therefore, the TV may ignore earliest presentation timestamps received from the CSA. This effectively enslaves the playout of media at the CSA to the broadcast service.

Presentation timestamps and control timestamps provided by the CSA and the TV are expected to take account of any delays between the point at which they sample the timeline position in their media pipelines and the presentation of the media to the user (light, sound). Similarly, if a set-top box and an HDMI-connected display are used, then the STB is expected to make a best-effort estimate to compensate for the playout delay of the display. HDMI signalling may be used for this purpose (Figs. 18.9 and 18.10).

As mentioned above, DVB-CSS supports several types of timelines each of which may have a different clock tick rate. However, even if the timed contents received at the TV and CSA have the same type of timeline and the same clock tick rate, there may be an offset between the timestamps and there may be clock drift if

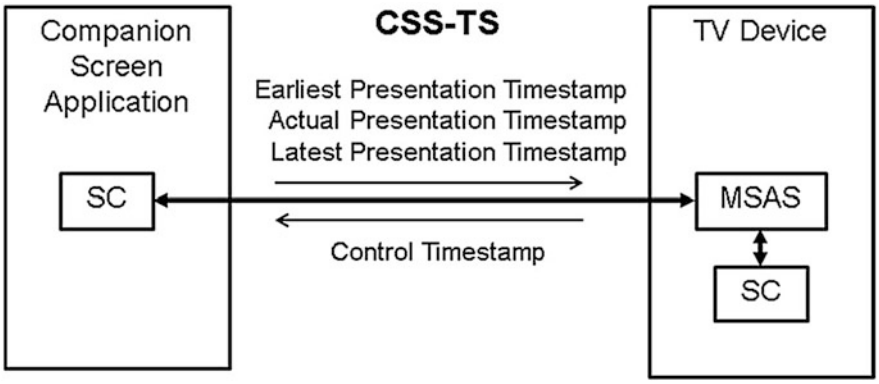


Fig. 18.9 DVB-CSS-TS protocol synchronises media presentation between CSA and TV

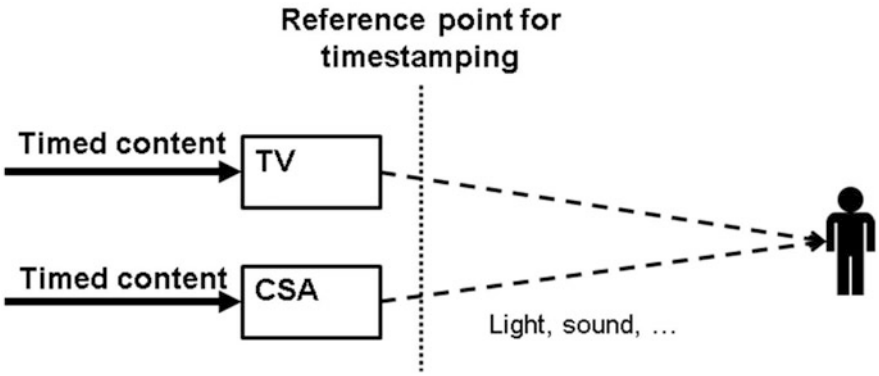


Fig. 18.10 Reference point for timestamping for the DVB-CSS-TS protocol. *Source* [12], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)

the source clocks are not generator locked ("gen-locked"). To support timeline synchronisation between timed content with different timeline types, clock tick rates, offset and/or clock drift, DVB-CSS introduces correlation timestamps. A correlation timestamp maps a point on one timeline to a point on another timeline. A correlation n seconds later in time can be calculated by simple linear extrapolation.

In case of clock drift, the correlation timestamp needs to be revised periodically. How frequently this is needed depends on how much inaccuracy in the correlation timestamp can be tolerated and how rapidly the timing relationship between the timelines drifts (Figs. 18.11 and 18.12).

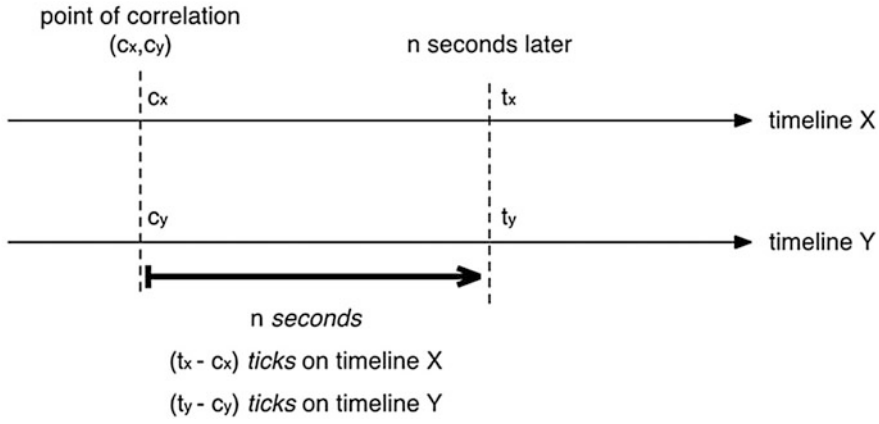


Fig. 18.11 Correlation timestamps map a point on one timeline to a point on another timeline. *Source* [12], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)

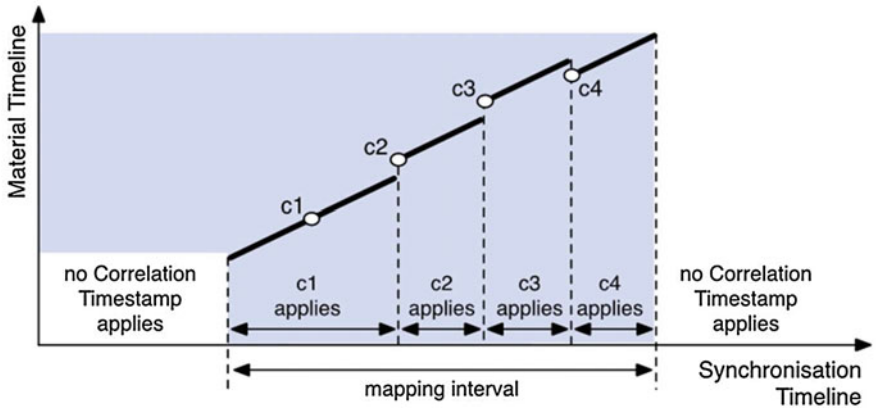


Fig. 18.12 Revised correlation timestamps to compensate clock drift. *Source* [12], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)

18.3 Multi-stream and Inter-device Media Synchronisation in HbbTV 2.0

As mentioned earlier, HbbTV is an industry forum that specifies an API for browser-based applications on TVs based on HTML, CSS and JavaScript. The new HbbTV 2.0 specification [16] includes features for media synchronisation, both multi-stream and inter-device. The former is media synchronisation between multiple streams within the TV where one stream (possibly together with a synchronising subtitle data stream) is received via broadcast and another stream via broadband Internet. The latter is media synchronisation between the TV and a CSA or another HbbTV 2.0 TV acting in the role of a CSA.

HbbTV 2.0 contains a profile of DVB-CSS [12] for media synchronisation. It is activated when an interactive application running on the TV explicitly requests it. The DVB-defined CSS-CII, CSS-WC and CSS-TS protocols are used in HbbTV 2.0, but TVs are not required to implement the CSS-TE protocol. The CSS-MRS protocol is not used, as HbbTV applications are downloaded from a broadcaster Web server, and HbbTV applications stay in contact with their Web server, so they can directly obtain any material information as needed. The media synchronisation functionality between TV and CSA is available for most media types that the TV can be playing, including both broadcast and streamed broadband content.

HbbTV 2.0 specifies a single API that can be used for both single-TV multi-stream synchronisation and inter-device synchronisation. For the latter, an HbbTV terminal can both act as “master” and as “slave”, enabling streams on two TVs to be synchronised with each other. The goal of the API is to control the media synchronisation processes in the TV, e.g. to programmatically start media synchronisation, to control what media streams are to be synchronised and to introduce correlation timestamps where needed. The API acts upon the MediaSynchroniser

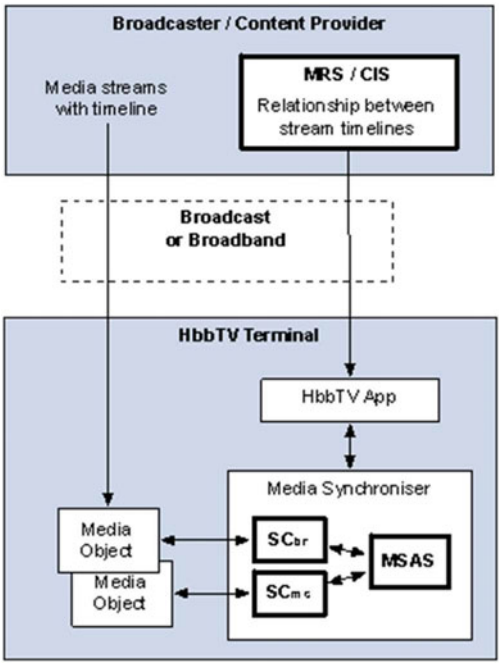


Fig. 18.13 Relationship between MediaSynchroniser object and HbbTV application for multi-stream synchronisation. (*Source* [16], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)). The functions of the SC and MSAS are the same as in DVB-CSS (previous section)

JavaScript object that represents all media synchronisation processes in the TV. This object is initialised by the API and populated with media objects, corresponding to to-be-synchronised media streams. The API also has methods to enable and disable the inter-device protocols explained above.

The HbbTV 2.0 specification includes a set of JavaScript APIs to control the media synchronisation and to trigger the media synchronisation protocols when needed. The APIs act on the MediaSynchroniser embedded object in the HbbTV terminal. The following are the main HbbTV 2.0 JavaScript APIs for media synchronisation (Figs. 18.13 and 18.14).

```
void initMediaSynchroniser() (Object mediaObject, String
timelineSelector)
```

This API initialises a `MediaSynchroniser` object for multi-stream synchronisation and for inter-device synchronisation as a master. `mediaObject` is the "master" media object against which all other media objects will be synchronised. It is typically the broadcast service received at the TV. `timelineSelector` specifies the type and details of the synchronisation timeline of the "master" media object.

```
void initSlaveMediaSynchroniser (String css_ci_service_
url)
```

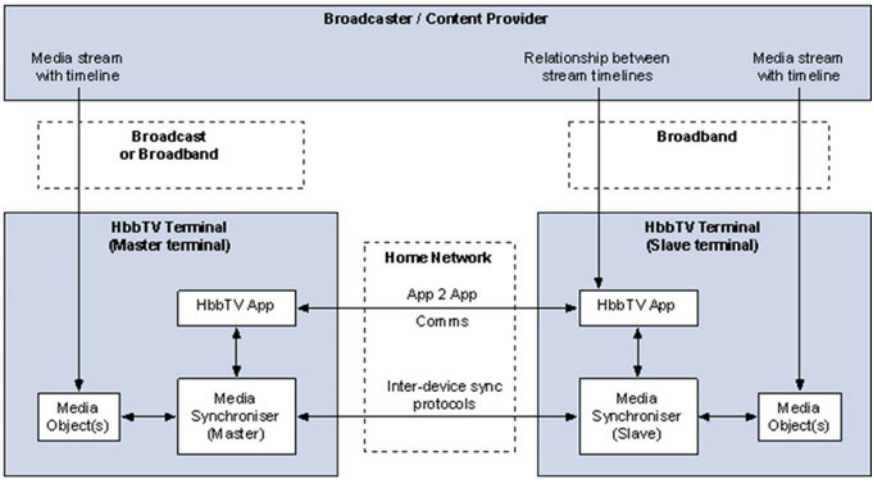


Fig. 18.14 Relationship between MediaSynchroniser object and HbbTV application for inter-device synchronisation with a slave terminal. (*Source* [16], © ETSI (reproduced with permission from the European Telecommunication Standards Institute. Further use, modification, copy and/or distribution is strictly prohibited)). The inter-device sync protocols are CSS-CII, CSS-WC and CSS-TS; see previous section

This API initialises a slave `MediaSynchroniser` object for inter-device synchronisation of the presentation of media objects on this TV ("slave") and on another TV ("master"). `css_ci_service_url` is the URL of the DVB-CSS-CII endpoint at the master TV.

```
void addMediaObject (Object mediaObject, String timelineSelector, CorrelationTimestamp correlationTimestamp, Number tolerance, Boolean multiDecoderMode)
```

This API adds a media object to the `MediaSynchroniser`. Once added, the TV synchronises the added "slave" media object with the "master" media object. `mediaObject` is the added "slave" media object. `timelineSelector` specifies the type and details of the timeline of the added "slave" media object. `correlationTimestamp` provides the (initial) `correlationTimestamp` between the "master" media object and the added "slave" media object. tolerance is an optional synchronisation tolerance in milliseconds. `multiDecoderMode` is a Boolean that specifies whether an additional decoder is needed, e.g. for picture-in-picture.

```
void removeMediaObject (Object mediaObject)
```

This API removes a media object to the `MediaSynchroniser`. `mediaObject` is the to-be-removed "slave" media object.

```
void updateCorrelationTimestamp (Object mediaObject, CorrelationTimestamp correlationTimestamp)
```

This API updates the correlation timestamp. `mediaObject` is the "slave" media object to which the correlation timestamp relates. `correlationTimestamp` is the new correlation timestamp.

`void enableInterDeviceSync (function callback)`

This API enables inter-device media synchronisation. It initiates the CSS-CII, CSS-WC and CSS-TS endpoints. callback is a function that is called when the endpoints are operable.

`void disableInterDeviceSync (function callback)`

This API disables inter-device media synchronisation. callback is a function that is called when the operation is completed.

Correlation timestamps are used in the same way as in DVB-CSS (previous section). These comprise a value on the timeline used by the MediaSynchroniser API for the "master" media object and a value on the timeline of a "slave" media object that correlates to the former value.

A media synchronisation buffer is optional in HbbTV 2.0. Even without a media synchronisation buffer in the TV, media synchronisation may be possible. The broadcaster can preload media streams for the CSA (or a slave HbbTV terminal) on a CDN. The broadcaster could editorially delay the broadcast stream, additional to inherent delay from compositing, encoding, packaging, etc., although this is not typically done for live streams. If any of the media streams is MPEG DASH (HbbTV only supports this type of standards-based adaptive streaming), then these are buffered in a CDN anyway. Moreover, the HbbTV 2.0 specification mandates that even if present, none of the scarce media synchronisation buffer capacity in the TV can be used for buffering MPEG DASH. If the TV has a media synchronisation buffer, then it will be at least 30 MB large. This is sufficient to reliably buffer at least 10 s of encoded high-definition television (HDTV) content assuming it consumes a bandwidth of 15 Mbps.

18.4 Device Discovery, Application Launch and App-to-App Communication

The above-described protocols for exchange of content and timing information require direct communication between the TV and the companion screen. This section explains technological means defined in HbbTV 2.0 for automatic discovery of the respective communication endpoints. Also, it looks at APIs for the launch of applications on discovered devices, a mechanism which is essential for bootstrapping a synchronised media experience. In addition, the section explains the technical concept behind App-to-App communication, which can be used to keep a consistent state between the TV and the CSA. Figure 18.15 illustrates the three core components defined in the specification. These are the App-to-App communication

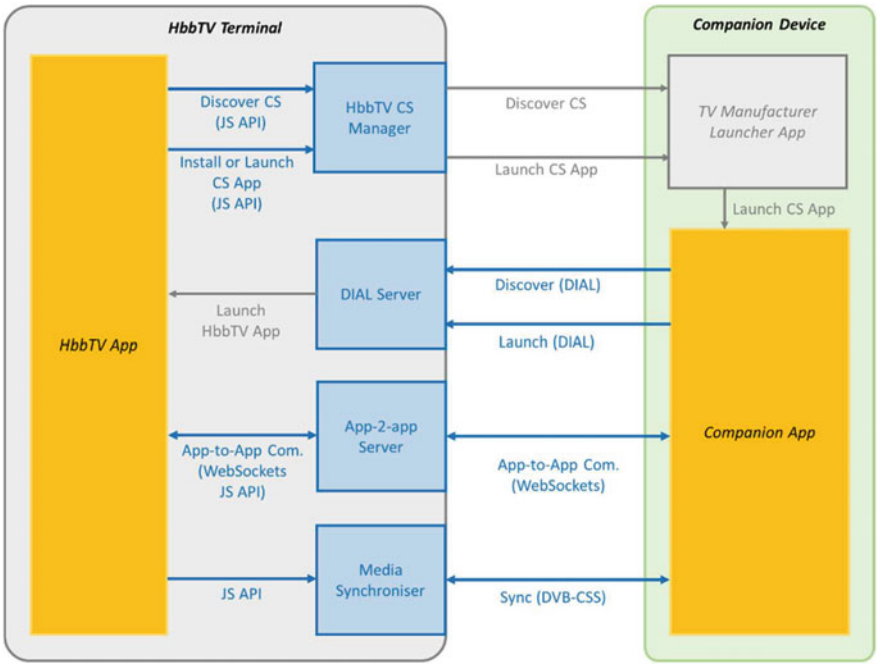


Fig. 18.15 Component diagram picturing the building blocks and the information exchanged between them to realise discovery, launch and App-to-App communication as defined in HbbTV 2.0

server, the DIAL server and the HbbTV CS Manager. This section describes the roles of these components with regard to discovery, application launch and App-to-App communication. Also, it outlines alternative technologies feasible for legacy devices.

18.4.1 Discovery

HbbTV 2.0 distinguishes between scenarios where either an application on a TV terminal or an application on a companion screen initiates the bootstrapping of the shared experience. Protocols and APIs for discovery and launch differ between these scenarios. While the protocol and the API for discovery and launch of applications on HbbTV terminals are based on the open and publicly available DIAL (DIscovery And Launch) specification, the protocols for discovery of and launch on companion screens are out of scope of the HbbTV 2.0 specification. The specification solely describes an API for HbbTV applications to invoke discovery and launch. The design of the underlying protocols is subject to TV manufacturers. In most cases, this means that the HbbTV terminal can only discover companion

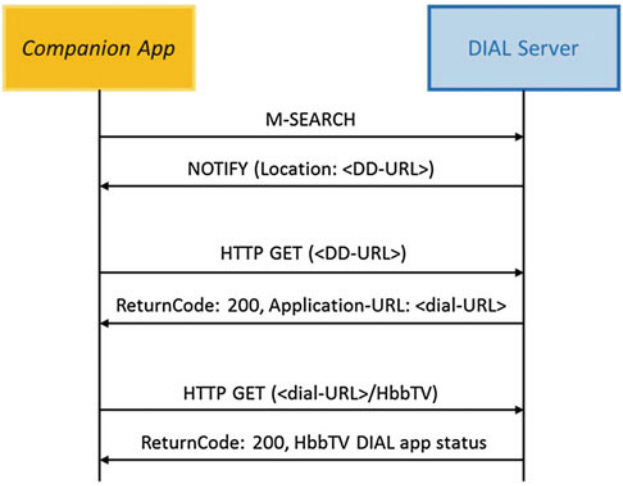


Fig. 18.16 Sequence diagram of discovery and device information exchange

screens running a vendor-specific application which implements the proprietary protocol endpoints.

Discovery of companion screens is handled by the HbbTV CS Manager component, which exposes its functionality through the `HbbtvCsManager` JavaScript object. Discovery is initiated through the call of `discoverCSLaunchers`. The method is passed a callback function, which is called on termination of the discovery process. The callback is passed an array of `CsLauncher` objects, which represent discovered companion screens.

Discovery of HbbTV terminals is accomplished through the Simple Service Discovery Protocol (SSDP) as part of the DIAL specification. Figure 18.16 illustrates the discovery process. To advertise itself on the local network and thus to initiate discovery, the CSA sends a M-SEARCH request to the multicast address 239.255.255.25:1900. The M-SEARCH request is an HTTP request sent via UDP. The HTTP header specifies search attributes, while the HTTP body remains empty. The header of the M-SEARCH request is specified in "14.7.3.1 DIAL Service Discovery" of the HbbTV specification [16].

DIAL servers on available HbbTV terminals respond to the M-SEARCH request with an SSDP NOTIFY message. The packet header of the NOTIFY message contains a location field, which contains the device description URL (<DD-URL>). The CSA sends an HTTP GET request to this resource location to retrieve the Application-URL header, which specifies the root location of the DIAL service's REST interface (<dial-URL>).

18.4.2 Application Launch

Launch of CSAs: Having discovered a companion screen, the HbbTV application can launch a CSA on the companion screen. Companion screen applications can be browser-based applications as well as applications running in the native run-time of the operation system on the mobile device, for example Android or iOS. For native run-times, the HbbTV application can also require the companion screen to install an application, if it is not installed on that device yet. Formats of the payload passed to the launch command of the `HbbtvCSManager` object are specified in Sect. 14.4.2 of the HbbTV specification [16].

Launch of HbbTV apps: HbbTV 2.0 specifies that a DIAL application should be addressable at location <dial-URL>/HbbTV. Through an HTTP GET request to this location, a CSA can query if the discovered device supports the HbbTV 2.0 DIAL feature (return code of the response equals 200). The body of the response from a DIAL-ready HbbTV terminal shall contain an Extensible Markup Language (XML) document comprising the following information:

- `X_HbbTV_App2AppURL`: remote endpoint for App-to-App communication.
- `X_HbbTV_InterDevSyncURL`: location of the CII server to initiate inter-device synchronisation.
- `X_HbbTV_InterDevSyncURL`: user agent of the HbbTV terminal.

The XML document shall conform the schema as defined in "14.7.2 Terminal and service endpoint discovery" of the HbbTV specification [16]. To launch an application on the HbbTV terminal, the companion application sends an XML Application Information Table (AIT) via HTTP POST to the HbbTV DIAL application (<dial-URL>/HbbTV). The AIT contains information on the HbbTV application including resource location and application identifier (App ID). The AIT format is specified in "14.6.2 Launching an HbbTV application protocol" of the HbbTV specification [16]. The return code in the header of the response to the POST request informs the client on whether the launch process succeeded or failed and on reasons for failure. Possible return codes are:

- 201 Launch succeeded.
- 403 Launch rejected by the user.
- 404 Application could not be retrieved by the terminal.
- 500 Launch failed for other reason (e.g. invalid XML AIT).
- 503 Launch rejected by the terminal, because of its current state (e.g. executing channel scan).

Pass protocol endpoints: If the HbbTV application launches a synchronised CSA, it should pass the DVB-CSS protocol endpoints together with the CSA locator, for example as query parameters appended to the launch URL. The TV application can query the address of its CII endpoint as well as the address of its remote endpoint of

the App-to-App communication service from the `HbbtvCSManager` object. In case the experience uses App-to-App communication, the initiator (companion or TV app) should also submit the App ID together with the launch information (see Sect. 18.4.3).

18.4.3 App-to-App Communication

The App-to-App communication facilities defined in HbbTV 2.0 provide crucial means for creating compelling distributed experiences. While DVB-CSS can be used to synchronise media timelines of two media items, App-to-App communication can be used to propagate state changes between companion and TV apps, to keep application states consistent across devices, to invoke expected responses to user interaction and thus to facilitate a plausible experience.

App-to-App communication is handled by the App-to-App communication server running on the TV terminal (see Fig. 18.15). The App-to-App communication server is a WebSocket server which provides endpoints for communication via the WebSocket protocol [17]. The App-to-App communication comprises a local endpoint for the TV application and a remote endpoint for the companion application to connect to. TV application and a Web-based companion application can make use of the W3C WebSocket API [18] to set up a connection, observe the state of the communication channel and to send and receive messages.

Before passing it to the constructor method of the WebSocket object, TV and companion applications need to append the App ID to the respective endpoint address. This is to prevent other HbbTV or companion screen applications from unintended interception of messages. The TV application queries the local endpoint address from the `HbbtvCSManager` object. If the TV application was launched by a companion application, it retrieves the App ID from the respective query parameter in the launch URL. If a companion application has triggered the application launch on the TV, it should query the remote WebSocket endpoint from the <dial-URL>/HbbTV endpoint after the successful application launch, as the address is only guaranteed to be stable for the lifetime of the TV application. If the companion application was launched from a TV application, it should retrieve the remote endpoint address and the App ID from the launch information.

After both applications have instantiated the WebSocket object, they should wait to receive the '`pairingcompleted`' message, which is sent by the TV terminal to both endpoints of the App-to-App communication server once the communication channel has been set up. Now TV and companion applications are ready to communicate.

18.4.4 Alternative Solutions

On HbbTV 1.X devices, App-to-App communication can be realised by means of a Web-based communication service, as shown in [19]. App-to-App communication service can also be used for simple inter-device synchronisation mechanisms, where TV timeline positions are forwarded via the communication channel to the companion. However, accuracy is most likely to be lower compared to inter-device synchronisation based on DVB-CSS. Sources for errors are mainly the accuracy of the wall-clock synchronisation and the accuracy of the TV timeline estimation. Accuracy of the wall-clock synchronisation is mainly limited by the round-trip time of messages between the companion and TV apps. Among network conditions, the round-trip time also depends on the implementation of the communication channel, for example polling [20] versus long polling [21]. Accuracy of the TV timeline estimation may vary between broadcast and broadband video. While for broadband video the play position can be queried from the video broadcast object, for broadcast video there is no API in the HbbTV 1.X specifications to access the broadcast timeline. Instead DVB stream events can be used to estimate the TV timeline position, where accuracy is in the order of magnitude of 500 ms (see Sect. 18.6.3).

The App-to-App communication via a Web service is interoperable with all legacy HbbTV devices and other Internet-connected TV or on-demand video platforms. However, it is at disadvantage with regard to a multitude of aspects. Operation and maintenance of the communication service introduce costs which increase with the number of users. Also, the Web-based communication has implications on the User Experience, as there are no means for automatic discovery of devices on the Web. Instead, user action is required for initialisation of a communication session, for example by scanning a QR code [22] or manually entering a session ID. Furthermore, as already outlined above, Web-based communication increases the time messages need to travel between the TV and the companion application. This reduces responsiveness of the distributed experience, as it takes more time to propagate state changes or control messages from one device to the other.

18.5 Application Scenarios, Integration and Deployment

This section gives a few examples of how the tools that have been described in Sects. 18.3 and 18.4 are used in HbbTV applications. In the application examples, it is shown how the API from Sect. 18.3 is used to synchronise a broadcast service with a media stream received from a broadband connection. Another example illustrates how the different tools are combined to set up media synchronisation between a TV and a companion screen application. The section is concluded by two examples elaborating issues and ways for a broadcaster to create and/or maintain

media timelines for media synchronisation through production, contribution and distribution networks.

18.5.1 Setting up Media Synchronisation from a Companion Screen

This section describes an example sequence how a CSA, e.g. on a smartphone, can use HbbTV to synchronise its media presentation with media on a TV, e.g. for showing multi-view content like multiple feeds in a sport event.

The sequence diagram in Fig. 18.17 shows the following actors:

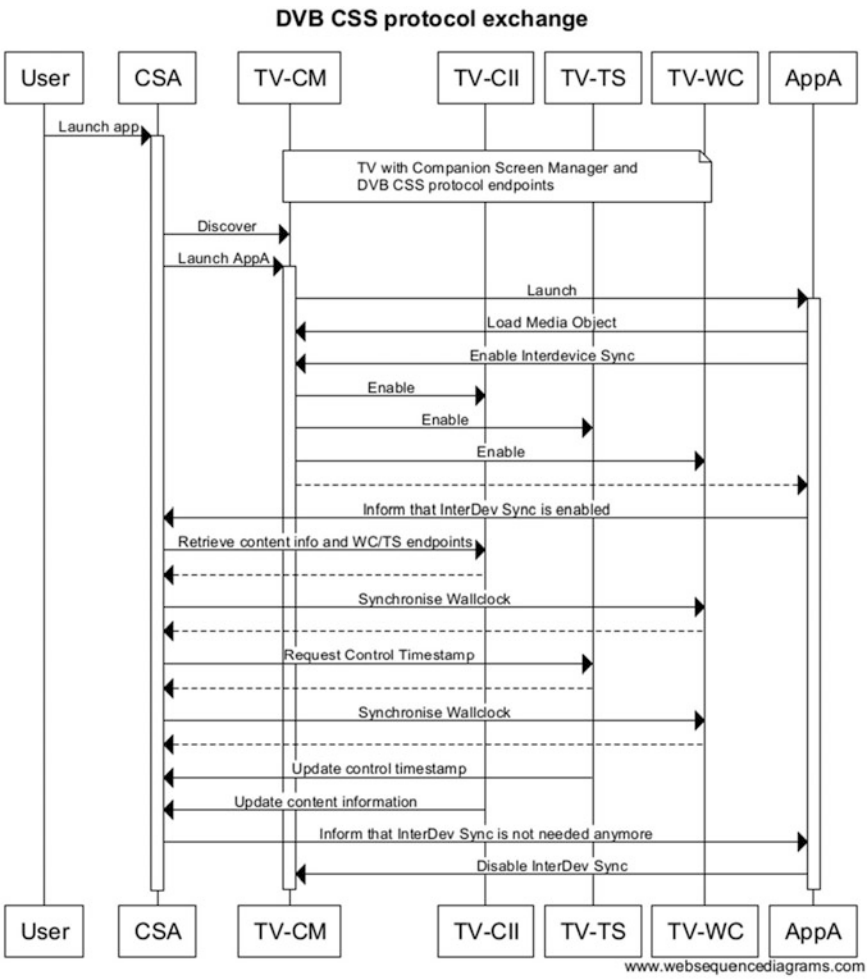


Fig. 18.17 Sequence diagram launching an HbbTV app for multi-screen media synchronisation

- User: the user of a smartphone and an HbbTV enabled TV.
- CSA: an application made available by the service provider. In this example, the user has installed the CSA on the companion screen.
- TV-CM: the companion screen manager on the TV. It is an aggregation of components of the HbbTV terminal related to companion screen functionalities including DIAL, App-to-App communication and the controller for media synchronisation.
- TV-CII,-TS,-WC: the endpoints provided by the terminal to allow inter-device synchronisation. CII: content identification, TS: timeline synchronisation, WC: wall-clock synchronisation.
- AppA: an HbbTV application, launched by the CSA, to control the media presentation on TV.

When launched and thereafter, the CSA looks for TV sets in the home network using the DIAL protocol. It then may offer the user an option to connect with the TV. If the user agrees, the CSA will issue a launch request (part of DIAL) for AppA. The TV has to ensure that the requested HbbTV application is being approved, e.g. by asking the user whether he wishes to launch the application. The HbbTV specification allows for other approval methods, e.g. by the manufacturer whitelisting applications. The CSA gets a response indicating the launch was successful or an appropriate error code.

If AppA started successfully, the TV will enable the App-to-App communication service. This is a WebSocket server with a local and a remote endpoint for passing messages back and forth. It is not shown in the diagram, but it is assumed that CSA and AppA establish a communication link, e.g. for the CSA to send a video URL for playback on TV.

Now AppA can start to set up the actual media synchronisation session. It uses a standard media element and initialises the HbbTV media synchroniser object with the media element and the synchronisation timeline. The external synchronisation endpoints of the TV, i.e. CII, WC and TS, are enabled with a single API call on the media synchroniser. This instructs the TV to act as a synchronisation master. The messages received from the CII endpoint, which is signalled in the handshake of the DIAL discovery process, provide the CSA with information about the current media controlled by the media synchroniser and the URLs of the WC and TS endpoints. Via the WC protocol, the CSA estimates an offset of its internal clock with the TV's wall clock. In the final step of the setup phase, the CSA sends a message to the TS endpoint. The TS endpoint responds with a control timestamp, which contains a timestamp of the synchronisation timeline, a corresponding sample of the TV's wall clock and the current timeline speed.

With the estimation of the TV's wall clock and the control timestamp, the CSA can synchronise its media presentation(s) to the media on the TV. From time to time, wall-clock synchronisation should be repeated, e.g. to compensate for

potential clock drifts. The protocols also provide sufficient information to calculate a dispersion value that allows the CSA to make judgments about the synchronisation accuracy.

This example shows how HbbTV 2.0 is used when starting the user journey from a companion device; see Sect. 18.4.2 for how HbbTV applications running on the TV can connect to and launch apps on companion screens.

18.5.2 Generating and Maintaining Media Timelines

HbbTV is a terminal specification and as such large parts of this section are terminal-centric. However, the new features for media synchronisation also require a few adaptations at the broadcaster's content contribution and distribution networks. To allow resynchronisation of streams delivered through different routes, the correlation of media timelines needs to be carried and maintained from the point of production to the end-user devices.

This section illustrates two scenarios that show how HbbTV 2.0 media synchronisation can be utilised and what is required by broadcasters on the production and playout side, as well in the content distribution networks involving one or more network operators.

It should be noted that different timelines for broadcast services, i.e. TEMI and PTS, are used for specific use cases in these examples. However, the approach to media synchronisation in HbbTV is designed such that timelines can be replaced easily by timelines of a different format. For example, a TEMI timeline could be generated from the PTS values, and instead of introducing a TEMI timeline, PTS could be used in conjunction with suitable correlation timestamps that are passed to the terminal through the MediaSynchroniser API.

18.5.2.1 Scenario 1—Using TEMI with Pre-produced Programmes

In scenario 1, we assume the broadcaster has pre-produced programmes, including a number of additional streams like audio description, clean sound and subtitles. Technically it is no problem to deliver these components over broadcast, but to reduce costs on the delivery path, it can be beneficial to use broadband Internet delivery for those components that are required by only a limited number of viewers. The broadcaster may also want to offer the additional service components for playback on companion screens that typically only support Internet connectivity and have no broadcast reception.

As the programme is pre-produced, the broadcaster can upload media clips with the additional components to its CDN provider in advance using the formats supported by HbbTV in case the components shall be rendered on the TV.

The timestamps used for A/V synchronisation of those media clips will be different from the corresponding broadcast service, so the broadcaster will have to deliver the relation between the start of a programme and the start of such additional components to the end-user devices. From the previous sections, we learned about the concept of media and synchronisation timelines. The media timeline for DASH is counted in ticks per second, the timescale is defined by the application and could be related to the underlying media streams. The timescale starts with zero at one of the DASH periods defined for the presentation. This is why the timeline is called DASH-period-relative timeline (DASH-PR). For our scenario, this timeline is fixed when the broadcaster uploads the file to the CDN and starts with the media presentation, which is the first period of it.

For the broadcast part of the service, which is based on DVB transport stream (i.e. not MPEG DASH), there are two alternative methods available to define the media timeline. One is directly based on the presentation timestamps (PTSs) and the programme clock (PCR) of the service and the other uses additional packets to carry a content-related timeline based on MPEG TEMI [13].

PCR/PTS samples are generated at the broadcast encoder, usually the last step before content distribution, and the TEMI packets are inserted there as well.

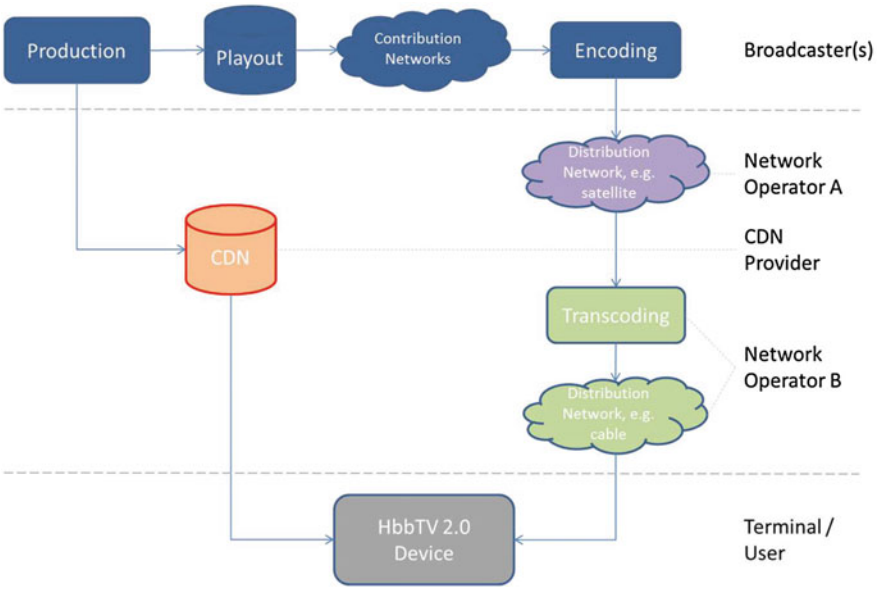


Fig. 18.18 Typical but simplified content flow from production to end-user

Determining the correlation between the media timeline of the broadcast service at the receiver and the media timelines of the content in a CDN is challenging for both types of timelines PCR/PTS and TEMI (Fig. 18.18).

During recording, production and post-production content is usually handled using timecodes, for example, based on the specification of the Society of Motion Picture and Television Engineers (SMPTE) ST 12-2:2014 [23]. Every frame has a timecode which is expressed in hours, minutes, seconds and frames. When single component is extracted at this stage to be delivered via broadband, the logical and the timing relations need to be maintained and stored in metadata that is exchanged between different systems of the broadcaster.

When it is time to distribute a programme in a linear TV channel, the content is played out from a storage system and transferred via the broadcaster's contribution networks to an encoding facility that encodes and multiplexes TV services for distribution, e.g. via a satellite network. The relevant standards for content exchange are (HD)-SDI (Serial Digital Interface) [24] for uncompressed and MPEG transport stream for compressed transfer. Playout servers stream the timecode-based content via SDI to the next component. Timecodes will not be transferred over SDI. Without any modifications, the timing as well as the logical relations to any CDN stored components would be lost. A solution is to use data fields in the SDI signal to carry timing and logical information. From analogue TV times SDI inherited the Vertical Blanking Interval (VBI) that allows to transfer frame-based data as Vertical Ancillary Data (VANC). SCTE-104 [25] is a specification that defines data structures carried over VANC. As it is connected to individual frames, timing relations can be maintained with frame accuracy. The BBC defined a concept [26] based on SCTE-104 to deliver timeline data that is identical to the format defined by MPEG TEMI for MPEG transport streams; i.e. it carries the timeline id and timestamps with a timescale. In the exchange of content from the playout, the SMPTE timecodes are translated into an MPEG TEMI compatible timeline. If there is a change from uncompressed SDI transmission to a compressed transport stream, e.g. for distribution encoding, the timeline can simply be maintained by copying the timeline ID, the timestamps and the timescale carried in the VANC of the SDI signal to the TEMI packets carried in the adaptation field extension of transport stream packets carrying the same video frame, and vice versa if needed.

When the terminal receives the service directly from the distribution network that got the multiplex from the broadcaster, the user can enjoy the extra components offered via the Internet by using the broadcasters HbbTV application. Problematic will be cases where another network operator is involved and broadcast services are redistributed, e.g. from a satellite network to a cable network. The satellite multiplexes usually are rearranged by transcoding or separate decoding and encoding steps, so with existing equipment a TEMI timeline will be lost afterwards. A technical solution could be built on the BBC approach [26] if the decoder and encoder are connected via SDI, assuming the network operator cooperates with the broadcaster which otherwise would block a technical solution based on the timeline approach.

18.5.2.2 Scenario 2—Live Productions

While the approach that is described in scenario 1 is well suited for pre-produced content as well as for collaborations with third parties, e.g. to provide accessibility services, it requires updates within contribution networks of broadcasters. If multiple broadcasters are connected to a network having individual and common services, a common technical solution has to be agreed and be implemented on equipment from a number of different vendors. Scenario 2 describes a solution that reduces the impact on broadcasters' existing deployments and also has advantages for live productions like sports.

In this scenario, the broadband stream will use the same timeline as the broadcast service. In terms of DVB-CSS, both delivery routes use PTS timelines. The components delivered via broadband are extracted after the broadcast encoder. The PTS of the broadband stream is locked to the PCR of the broadcast service. Hence, no correlation information is needed at the terminal side. The implementation on the terminal only needs to care about the gap in reception of the various streams which is likely to be multiple seconds. The concept has been showcased in the HBB-NEXT project on a standard set-top box hardware and was shown in Amsterdam 2014 at the IBC trade fair [27].

As far as production, playout and contribution-side components are concerned, components distributed via broadband are handled as if they were distributed via broadcast, except that there may be some metadata specifying that the component is delivered via broadband, and where it will be available from.

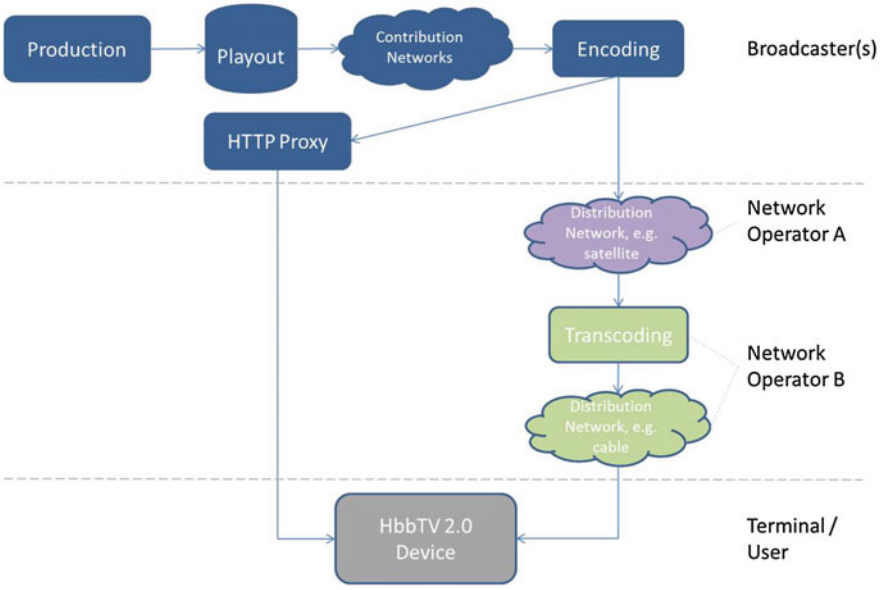


Fig. 18.19 Modified content flow for HTTP streaming of service components

As shown in Fig. 18.19, the broadband content is extracted from the compressed stream encoded for distribution. The components for broadband delivery are sent to an HTTP proxy which delivers them directly to HbbTV terminals on request. This transport has been defined for HbbTV 1.0 and is used for live streaming of events and also linear TV channels [28].

As delivery of live MPEG-2 transport streams via progressive download is not supported by big CDNs like Akamai, some providers have implemented specific solutions for individual broadcasters.

One advantageous side effect of this method is that it does not have a big delay in the reception of the corresponding component relative to the rest of the service received via broadcast. If the broadcast is redistributed via cable or terrestrial, it might well happen that it is actually later than the broadband components. Delivery of individual components also does not require high bandwidths; e.g. audio typically is encoded at 200 kbit/s.

However, it should be noted that there are efforts to standardise optimisations for adaptive streaming technologies like MPEG DASH to allow for broadcast comparable end-to-end delays.

On the terminal side, an additional buffer, large enough to compensate for the delivery delay, is needed. This buffer is defined in HbbTV 2.0 but not mandatory. For inter-device synchronisation, the buffer in the TV may not be necessary if broadcast is received later than the broadband streams by the companion device.

As with the TEMI timeline in scenario 1, there is as well the issue of maintaining the timing relation if the broadcast service is transcoded for redistribution to another network. The clock of the encoder has to be locked to the clock of the decoder, and PTS of individual frames have to be preserved. If PTS cannot be preserved, the broadcaster would have to identify the new relation between the PTS of the broadcast stream and the broadband components. That relation is then passed on to the companion screen application that is presenting the broadband streams or to the terminal by configuring the MediaSynchroniser object if the TV shall synchronise the broadband component to the broadcast stream.

As there are potentially many different cable or terrestrial networks in a larger area that is covered by a broadcaster, it is necessary to identify the distribution network from within the HbbTV application to adapt to the specific timeline modification. Without the help of the network operator, determining the correlation between the PTS timeline of the transcoded broadcast service and the original timeline would mean extra efforts for the broadcaster for each single network where the content is redistributed.

18.6 Synchronisation of HbbTV Applications with Broadcast and Broadband Media

This section describes various ways, including pros and cons, to synchronise an HbbTV application with broadcast services and broadband streams. Applications running on a companion device may use the methods to some extent by combining them with communication channels like the App-to-App communication service that is introduced in Sect. 18.4.3.

The HbbTV standard provides facilities for the creation of programme-related interactive TV applications. However, only few applications available today take advantage of the full potential of HbbTV. Existing applications are often static, TV-tailored websites, such as video-on-demand portals or news apps. While these kinds of applications offer a useful and popular service, they do not utilise the possibilities of HbbTV to engage the viewer with a broadcast programme. Potential use cases range from presenting additional information related to the current event or scene to participation of the viewer in TV shows. An example use case is shown in Sect. 18.1.3.

The HbbTV specification includes a number of solutions for media synchronisation. The best choice depends on the required accuracy and on how the content is distributed: broadcast vs. on-demand vs. broadband live stream. This section will discuss different approaches and their potential timing accuracy.

The two methods discussed first, EIT (Event Information Table) and receiver clock, are less accurate but sufficient for applications looking for some coarse synchronisation, e.g. an application offering background information on guests in a talk show. If there are higher demands for synchronisation as in an interactive show like a quiz or interactive advertisement, stream events or media timelines should be used, which are the methods discussed later in this section.

18.6.1 EIT Present/Following

Digital broadcast services contain signalling for receiver's Electronic Program Guide (EPG), delivered in the Event Information Table (EIT). In DVB, there is signalling for the programme schedule for up to 4 weeks (EIT schedule). More interesting for the purpose of synchronising is the EIT p/f ("present/following"), which includes the current running event (EIT present) and the immediately following television programme. EIT p/f can be used to signal the actual start of an event by the change of the event that is signalled as present; this can be different to the schedule by a couple of minutes to several hours if the broadcaster decides to change the schedule in the short term, e.g. if there are breaking news in case of catastrophes.

An HbbTV application can receive the EIT p/f of a broadcast service via the video broadcast ("v/b") object that is bound to that service. The v/b object allows an

application to control the presentation of the broadcast service, i.e. size, position of the TV picture and selected components, but also to receive DVB service information like the EIT. The programme property of the v/b object will return two programme objects, if EIT p/f is present for the service. The `onProgrammesChanged` event on the v/b object informs the application on a programme switch, i.e. when the EIT following becomes the new EIT present. Broadcasters can use this change in the EIT p/f to allow accurate recordings of broadcast events. The event start time that can be received through the `startTime` property of an object representing the programme, the TV terminal is currently tuned to, usually contains the scheduled time not the actual air time. Depending on the required accuracy, the end-to-end delay of the broadcast distribution needs to be considered as well.

The following sample code shows how to get an event when a new programme starts:

```
vb = oipfObjectFactory.createVideoBroadcastObject();
// add video broadcast object (v/b object) to the HTML
// document
vb.bindToCurrentChannel();
// wait for the V/B object being in the presenting state
vb.onProgrammesChanged = function () {
    // check which programme just started
    eitPresent = vb.programmes[0];
}
```

Further details of the application programming interface (API) can be found in Sects. 7.13.3 and 7.16.2 of the OIPF DAE specification [29]. The actual event can be identified by its event id, start time or event name.

The accuracy of this method largely depends on how accurate the broadcaster keeps the start time in the EIT p/f and how a receiver sets its internal clock, which could be set by the time signalled in broadcast streams (Time Date Table or TDT) or by using Internet time servers via NTP [30].

For live streaming with MPEG DASH, DVB defined an equivalent signalling of programme information in the DVB DASH [31] specification. The API is defined in clause 9.3.2 of HbbTV 2.0 [16].

The achievable accuracy for application-to-A/V synchronisation using the EIT p/f mainly depends on the following factors:

- Accuracy of EIT signalling itself—some broadcasters just use the schedule which does not include short-term programme changes. The start time of an event can be expressed in seconds.
- Accuracy of receiver clock as reported by the JavaScript Date object; see also next paragraph.

18.6.2 Receiver Time/Clock

JavaScript provides the Date object, which gives access to the system clock of the host device and in HbbTV of the receiver. Nowadays, computers synchronise time using the NTP [30] with a few servers available on the Internet. Performed on broadband connections the protocol is accurate enough for application-to-audio-video synchronisation.

Although TV broadcast receivers have an alternative means to receive the current time, they can parse the Time Date Table (TDT) and the Time Offset Table (TOT) which give the current time of the broadcast signal and the time zone of it. The problem with TDT is that it may differ between broadcast streams of different providers. Moreover, there are vendor-specific differences in terminal implementations, as some devices read the time only during start-up or on every channel change.

All these different implementations are valid from an HbbTV perspective as it is not defined from where time is taken. Additionally, some implementations are not handling time zones correctly. The Date object has a set of methods to return the UTC time and a corresponding set for local time. It may happen that the time zone setting of the receiver is not reflected in the values returned by that API; i.e. the local time is the same as UTC.

The only way to handle this situation properly is to calculate the offset of the local time with a server with an NTP-like protocol.

To improve synchronisation accuracy, applications may need to compensate for the delivery delay of the broadcast. Measurements [3, 32] show a span of 15 s between the fastest and the slowest broadcast network—e.g. from analogue to IPTV. It should be noted that these measurements are snapshots and may vary over time when network operators change their infrastructure. To determine the actual broadcast network, HbbTV applications can check the type of network, i.e. satellite, cable or terrestrial and since HbbTV 1.5 also the Network Identifier (NID) which allows to some extent to identify the operator and region of the broadcast.

18.6.3 Stream Events

Stream events in HbbTV are available for broadcast services and for MPEG DASH streams. This subsection will discuss the broadcast events in more detail.

The DASH events have been added in HbbTV version 2.0. They are either carried in the DASH manifest file (“Media Presentation Description”, MPD) or in an ISOBMFF-based structure that is referenced from the MPD. Access to the events from an application level is provided by HTML5 text cues via the HTML5 media elements.

Stream events for broadcast are defined in DSM-CC (Digital Storage Media Command and Control) [33]. DSM-CC specifies a set of protocols that are profiled

by standard bodies like DVB [10] to deliver data and events via broadcast networks. HbbTV uses only the do-it-now stream events. A do-it-now event does not have a presentation timestamp: it is passed to the application on reception by the terminal. So its accuracy depends mainly on its position in the broadcast stream and the processing delay in the receiver.

Stream events are carried as stream event descriptors using the MPEG section syntax. A descriptor contains an event id, a version number and some bytes of payload. A stream event object, carried along with the application, maps event ids to names and identifies the component that carries the descriptors. Applications register for stream events by an API call which passes an event name, the location of the stream event object and a callback function. The access rules defined in HbbTV require the application to be signalled on the broadcast service.

```
vb = oipfObjectFactory.createVideoBroadcastObject();
// add V/B object to the DOM
vb.bindToCurrentChannel();
// wait for the V/B object being in the presenting state
vb.addStreamEventListener('http://myserver.com/event.xml',
    'evt_a', myHandler);
function myHandler (event) {
    log(event.name + ": " + event.text);
}
```

The accuracy of stream events for app-to-A/V synchronisation depends on a number of factors. The following list overviews the main causes for delays introduced by the playout system as well as by the receiver of the broadcast service.

- Accuracy of event information at the broadcast playout centre—Stream events are usually inserted after encoding for content distribution. The challenge for the broadcaster is to synchronise the events with broadcast the content before distribution. As there are usually fixed end-to-end delays, systems are designed to manually adjust for these delays.
- Accuracy of adding a stream event to a transport stream—As the stream event sections have no defined timing, i.e. they carry no presentation timestamp, it is impossible to keep an exact synchronisation of an event with the broadcast A/V content.
- Maintaining stream events while transcoding broadcast streams—Many cable networks receive content through satellites and rearrange content which requires transcoding to efficiently use the available bandwidth. First of all, the cable operator has to take care to also redistribute the stream events and not just drop them. This is a minor technical issue, but may be influenced by business considerations. Secondly, the relative position of the stream event must be preserved, as the transcoding of the A/V content will add an additional delay. If the events are simply passed through unchanged, they will arrive too early.

- Delivery of stream events to the HbbTV application—The accuracy of a stream event also depends on the processing in the TV terminal. The HbbTV specification requires execution of the actions associated with a stream event to the immediately when it is received, irrespective of the A/V content currently presented. In practice, there is also the observation that events have large delays of multiple seconds if the terminal is busy with other tasks, especially after a channel change.
- The time the device needs to actually perform requests for a change of the user interface. This is not directly related to the accuracy of the stream event, but it has to be considered for the use case of application-to-A/V synchronisation. For instance, if an application wants to present a red button, it will modify the HTML DOM (Document Object Model). This is asynchronous and the browser will do it at the next occasion, i.e. when it has a time slot to do that. Depending on the performance of the device, this time difference could be notable.

DSM-CC stream events in HbbTV have no constraints defining their accuracy. Applications using stream events have shown that they are suitable for presenting subtitles or notifications such as a red button for an ad clip.

Fig. 18.20 presents results of an experiment conducted at IRT to quantify how reliably stream events are processed at devices from different vendors and different models of the same vendor. The experiment evaluates the influence of the last two factors of the bullets list above, namely the processing of stream events in the terminal and how fast changes made by an application are visible to the user.

The test stream for the experiment was pre-recorded with a number of stream events and played out in a loop to all devices. This ensures that every device during each loop gets an identical input signal. Additionally, the video of the stream includes a frame counter. An HbbTV application signalled in the stream receives the stream events and immediately changes the graphical overlay on the screen. The counter of the first frame in the video when the change in the overlay can be

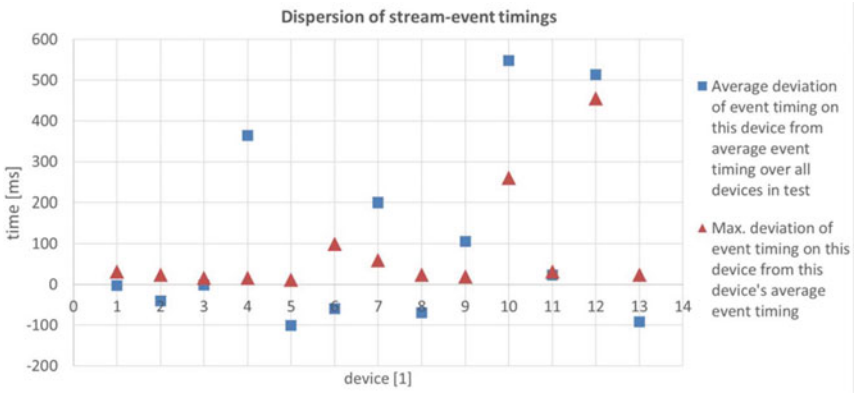


Fig. 18.20 Deviation of stream event triggers on different HbbTV terminals

detected is recorded. For each stream event in the stream, all measurement points give an average value across all tested devices. For each tested device, the average and the maximum deviation in milliseconds from the overall average are calculated. The diagram in Fig. 18.20 shows these two values for 13 different devices from 5 vendors. On the devices in the test field, the maximum deviation in this experiment was 500 ms from the average value.

The first experiment was performed when the HbbTV application was idle except for listening to the events. In a second experiment, the influence of other running tasks in the terminal was evaluated. The testing application included an animation of an image using a JavaScript library. Results are very similar to the first experiment except for two devices from the same manufacturer where the maximum deviation was about 2 s.

To summarise, the accuracy of stream events is usually in the range of a second, but in cases where the terminal is busy with other tasks, which can happen directly after a channel change or when a HbbTV application is consuming a lot of resources, the accuracy cannot be guaranteed.

18.6.4 DVB-CSS Media Timelines

HbbTV version 2.0 adds a new set of features for media synchronisation. While their main purpose is enabling multi-stream and inter-device synchronisation as discussed in the subsequent sections, they also improve synchronisation of HTML applications with media streams, e.g. a broadcast service or on-demand clips.

The concept of the media synchronisation that is defined by DVB-CSS builds on timelines which are associated with every single media stream or broadcast service.

HbbTV 2.0 defines an API that allows applications to access the media timeline of the currently presented media.

In addition, it defines the constraints for the returned value of a timeline and the play position of a media stream. Play position is usually the time span from the start of playback and is available on devices implementing version of the HbbTV specification. But it is not defined where in the media processing the current playback position is measured, nor how accurate the value shall be and how often it needs to be updated. For details of the HbbTV 2.0 constraints; see clause 13.11 in [16]:

- Play position and values on a timeline are measured at the point where application graphics and video are composed, which is different to inter-device synchronisation that defines the display or speaker output as the reference point.
- The value shall be updated each time the application reads it, at least for the A/V object. The accuracy of this measurement shall be 100 ms.
- The accuracy of a measurement relates to the frame rate of a video stream or the sample rate of the audio, i.e. between 20 ms and 42 ms for media supported by HbbTV 2.0.

While the play position will be sufficient for on-demand media, where the play position relates to the media timeline, this is not possible for live streams, where the play position is not related to the absolute media timeline. Applications can achieve much more accurate synchronisation via the `MediaSynchroniser` object of the HbbTV JavaScript API; see also Sect. 18.3. Supported timelines are the following:

- Presentation timestamps (PTSs) that are carried with timed media in MPEG-2 transport streams and TEMI [34] timestamps that define a content timeline which actually is based on PCR/PTS for MPEG-2 TS-based media. These timelines are available for broadcast services and broadband streams using the transport stream format.
- The MPEG DASH-period-relative timeline, which defines a timeline based on the start of a DASH period, e.g. the start of a live stream if there is only one period.
- Composition Time for the ISOBMFF. The format also known as MP4 files is mainly used for on-demand content. Note that the DASH profile of HbbTV is based on the ISOBMFF, but the Composition Time timeline is only used for non-DASH content.

As discussed in Sect. 18.5.2, timelines for broadcast content, i.e. TEMI and PTS, require special care if broadcast services are being transcoded in the delivery chain. The following pseudocode shows how an application can access the TEMI timeline of a broadcast service;

```
vb = oipfObjectFactory.createVideoBroadcastObject();
// add V/B object to the DOM
vb.bindToCurrentChannel();
// wait for the V/B object being in the presenting state
ms = oipfObjectFactory.createMediaSynchroniser();
// initialize the media synchroniser with the v/b object
and
// the reference to the TEMI timeline
ms.initMediaSynchroniser(vb,
    "urn:dvb:css:timeline:temi:8:1");
// read the current value on the timeline
now = ms.currentTime;
```

The example above assumes the broadcaster included a TEMI-based timeline on the current service. The timeline is delivered by means of TEMI descriptors which are carried in the adaptation field (extension) of transport stream packets. The component can be any which uses the Packetised Elementary Stream (PES) format, i.e. video, audio or specific timeline components. The component is identified by the timeline identifier, e.g. “urn:dvb:css:timeline:temi:8:1”. In this example, the component has a component tag “8” and a timeline id of “1”. The component tag is specified in the Programme Map Table (PMT) and the timeline id in the timeline

Table 18.1 Comparison between different time sources for App-to-A/V synchronisation

Approach	PRO	CON
EIT p/f	Reuse of existing broadcast signalling Used to enable accurate recording; i.e. app can detect programme switch quite precisely	Cannot be used to reliably calculate position in a broadcast programme
Receiver time	Does not rely on broadcast signalling	Most inaccurate approach
Stream events	Good accuracy for most use cases Available in all HbbTV devices, playout solutions exist in the market	Accuracy is not defined, hence not part of test suite Not robust for transcoding, e.g. redistribution in cable networks
Media timelines	Well-defined accuracy, that is tested by HbbTV test suite	Not robust for transcoding, e.g. redistribution in cable networks

descriptors. This means a service can have multiple components with multiple timelines. The TEMI timeline is synchronised with the PCR by the PTS of the packet carrying the timeline descriptor and will be re-calculated by the client. It can run continuously or could be restarted with every broadcast programme and paused for ad breaks depending on the needs of the broadcaster.

18.6.5 Summary

See Table 18.1.

18.7 Conclusion

This chapter has presented media synchronisation approaches for television services through HbbTV. HbbTV (Hybrid Broadcast Broadband Television) provides broadcaster-controlled browser overlay applications over television channels, where the television channels are received via regular broadcast and the HbbTV applications (Internet). The new HbbTV 2.0 standard [16] introduces support for media synchronisation, as well as support of tablet devices ("companion screens") that are used in conjunction with the television. This combination enables new use cases like playing out alternative audio on the tablet device synchronously with a broadcast video stream on the television. The new HbbTV standard uses some of the media synchronisation protocol suite specified by the new DVB-CSS standard [12] and augments this with an API to make these protocols usable in the HbbTV context. As a result, frame-accurate media synchronisation is enabled.

The chapter also describes other HbbTV functionalities that are needed to make a media synchronisation application fully work with a companion screen, including companion screen application discovery and launch and application-to-application communication between the television and the companion screen. Two deployment scenarios show how media timelines are generated and maintained for pre-produced programmes and live productions. Finally, additional tools from the HbbTV toolkit are discussed for the synchronisation between HbbTV applications and a television broadcast.

Successful standardisation efforts are key for the industrial partners. Vendors in HbbTV have already committed to implement at least the mandatory aspects of HbbTV 2.0 (including the profile of DVB-CSS) in their new TV products, with the expectation of seeing compliant products in 2017. Meanwhile broadcasters, including the BBC [35], are already exploring the services that media synchronisation will enable. The main remaining challenge will be the creation of compelling media content and applications that can effectively use the new media synchronisation features provided by HbbTV and to integrate these into the workflows and business models of television broadcasters.

Acknowledgements Major contribution to this paper has been made within the framework of the collaborative project 2-IMMERSE. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 687655.

Definitions

Media synchronisation the process of synchronisation of the presentation of media streams received via two or more paths, e.g. broadcast and Internet.

Inter-device media synchronisation the process of synchronisation of the presentation of media streams on two or more devices

Multistream media synchronisation synchronisation of the presentation of media streams received via two or more paths (e.g. broadcast and broadband) on a single device

References

1. Geerts, D., Vaishnavi, I., Mekuria, R., van Deventer, M., Cesar, P.: Are we in Sync? Synchronization requirements for watching online video together. In: ACM Conference on Human Factors in Computing Systems, pp. 311–314, 7–12 May 2011
2. ITU-R BT.1359-1, Relative timing of sound and vision for broadcasting, International Telecommunication Union (ITU) (1998)
3. Kooij, W.J., Stokking, H.M., van Brandenburg, R., de Boer, P.T.: Playout delay of TV signals: measurement system design, validation and results. In: ACM International

Conference on Interactive Experiences for TV and online video (ACM-TVX), pp. 23–30, 25–27 June 2014
4. Mason, A.J.: WHP 296—EBU Tests of Commercial Audio Watermarking Systems. British Broadcasting Cooperation, London (2004)
5. Montagud, M., Boronat, F., Stokking, H., van Brandenburg, R.: Inter-destination multimedia synchronization: schemes, use cases and standardization. Multimed. Syst. **18**(6), 459–482 (2012)
6. van Deventer, M.O., Stokking, H., Hammond, M., Le Feuvre, J., Pablo, C.: Standards for multi-stream and multi-device media synchronization. IEEE Commun. Mag.—Commun. Stand. Suppl. **54**(3), 16–21 (2016)
7. HbbTV: Official website of the HbbTV association (2017). https://www.hbbtv.org/. 5 Aug 2017
8. MHEG: MHEG official website (2014). https://www.mheg.org/users/mheg/index.php. Accessed 8 May 2017
9. MHP: DVB Fact Sheet Multimedia Home Platform, Digital Video Broadcasting (DVB) (2011)
10. ETSI: ETSI TS 102 809 V1.2.1—Signalling and carriage of interactive applications and services in hybrid broadcast/broadband environment, European Telecommunications Standards Institute, Sophia Antipolis (2013)
11. Ziegler, C., Keimel, C., Ramdhany, R., Vinayagamoorthy, V.: On time or not on time: a user study on delays in a synchronised companion-screen experience. In: ACM International Conference on Interactive Experiences for TV and online Video (ACM-TVX), p. to appear, 14–16 June 2017
12. DVB-CSS, ETSI TS 103 286-2—Part 2: Content Identification and Media Synchronisation, European Telecommunications Standards Institute (ETSI), Sophia Antipolis (2015)
13. MPEG, ISO/IEC 13818-1:2013/DAM 6: Delivery of Timeline for External Data, International Organization for Standardization (2014)
14. Mills, D., Delaware, U., Martin, J., Burbank, J., Kasch, W.: RFC 5905: network time protocol version 4: protocol and algorithms specification. Int Eng. Task Force (2010)
15. Vinayagamoorthy, V., Ramdhany, R., Hammond, M.: Enabling frame–accurate synchronised companion screen experiences. ACM Int. Conf. Interact. Exp. TV online video (ACM-TVX) 83–92, 22–24 June 2016
16. HbbTV 2.0.1: ETSI TS 102 796—Hybrid Broadcast Broadband TV, European Telecommunications Standards Institute (ETSI), Sophia Antipolis (2016)
17. RFC 6455: The WebSocket Protocol, Internet Engineering Task Force (IETF) (2011)
18. W3C: The WebSocket API. World Wide Web Consortium (2011)
19. Ziegler, C.: Second screen for HbbTV—Automatic application launch and app-to-app communication enabling novel TV programme related second-screen scenarios. In: IEEE Third International Conference on Consumer Electronics Berlin (ICCE-Berlin), pp. 385–389, 9–11 Sept 2013
20. Wikipedia: Polling (computer science) (2017). https://www.en.wikipedia.org/wiki/Polling_(computer_science). Accessed 8 May 2017
21. Wikipedia: Push technology (2017). https://www.en.wikipedia.org/wiki/Push_technology. Accessed 8 May 2017
22. Wikipedia: QR code (2017). https://www.en.wikipedia.org/wiki/QR_code. Accessed 8 May 2017
23. SMPTE: ST 12-2:2014—Transmission of Time Code in the Ancillary Data Space, Society of Motion Picture and Television Engineers (2014)
24. SMPTE, 292 M-1998: Bit-Serial Digital Interface for High Definition Television, Society of Motion Picture and Television Engineers (1998)
25. STCE, ANSI/SCTE 104 2012: Automation System to Compression System Communications Applications Program Interface, Society of Cable Telecommunications Engineers, Exton (2012)

26. Ware, T.: WHP 296—Programme related metadata in SDI VANC. Brisitish Broadcasting Cooperation, London (2015)
27. European Commission, HBB-NEXT: TV evolution leaves no one behind, 22 Oct 2014. https://www.ec.europa.eu/digital-single-market/en/news/hbb-next-tv-evolution-leaves-no-one-behind. Accessed 8 May 2017
28. HbbTV 1.0: ETSI TS 102 796 v.1.1.1—Hybrid Broadcast Broadband TV, European Telecommunications Standards Institute, Sophia Antipolis (2010)
29. OIPF: Volume 5—Declarative Application Environment Release 1 v1.1, Open IPTV Forum (2009)
30. IETF: RFC 5905—Network Time Protocol Version 4: Protocol and Algorithms Specification, Internet Engineering Task Force (2010)
31. DVB-DASH: ETSI TS 103 285—MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks," European Telecommunications Standards Institute, Sophia Antipolis (2015)
32. Zota, V.: Anpfiff!—Technik für eine ungetrübte Fußball-WM, Heise, 31 5 (2014). https://www.heise.de/ct/ausgabe/2014-13-Technik-fuer-eine-ungetruebte-Fussball-WM-2221818.html. Accessed 8 May 2017
33. MPEG: ISO/IEC 13818-6:1998—Generic coding of moving pictures and associated audio information—Part 6: Extensions for DSM-CC. International Organization for Standardization (1998)
34. MPEG, ISO/IEC 13818-1:2013/DAM 6: Delivery of Timeline for External Data (2014)
35. Jolly, S.J.E., Evans, M.J.: WHP 242—Improving the Experience of Media in the Connected Home with a New Approach to Inter-Device Communication. British Broadcasting Cooperation, London (2012)

Part V
Algorithms, Protocols and Techniques

Chapter 19
Video Delivery and Challenges: TV, Broadcast and Over The Top

Tim Stevens and Stephen Appleby

Abstract The TV production and broadcasting industry predates the ubiquitous computing and IP technologies of today. However, just as these advances have revolutionised other industries, they are also causing production and broadcasting to change. Here, we outline the opportunities that general computing and IP delivery offer this industry, and discuss how the precise synchronisation required by TV services could be implemented using these more generic technologies, and how this in turn could lead to newer ways of delivering TV-like services. We first discuss how today's TV industry has been shaped by its analogue roots, and that the terminology and working practices still in some ways reflect the analogue world. We briefly cover TV history from the 1950s and the evolution of Public-Sector Broadcasting in the UK, before considering how newer services such as digital TV, satellite and video streaming have enabled services, but also throw up new issues around delay and synchronisation. We propose that some of these issues could be mitigated by moving to an IP delivery model, with media elements composed at the client device, and not globally time-locked to precise, system-wide clocks. Finally, we discuss some of the IP delivery technologies such as multicast, adaptive streaming and the newer protocols that are replacing traditional HTTP.

Keywords Broadcast TV · Asynchronous infrastructure · Future IP transport protocols

19.1 Introduction

In many industries, specialist appliances are being replaced by generic computing and network infrastructure running software to provide the specialist capabilities. There are many obvious examples of this over previous decades, including word

T. Stevens (✉) · S. Appleby
British Telecommunications PLC, London, UK
e-mail: tim.s.stevens@bt.com

S. Appleby
e-mail: steve.appleby@bt.com

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_19

processors, calculators, etc. In general, generic capabilities have a much larger market than specialist appliances and so receive much larger investment.

Here, we examine the impact of the trend towards generic infrastructure in the delivery of TV-like services from several perspectives, and the implications of this trend on synchronisation.

In the production and broadcast domain, despite a partial move to digital technology, analogue working practices are still the norm. Studios and broadcasters make heavy use of specialist TV and video equipment, and the lingua franca of interconnection is the Serial Digital Interface (SDI). SDI [1] is a point-to-point connection, which, despite being digital, mimics an analogue interconnection. SDI's core task is to carry video, with all other media and metadata being carried as ancillary data. SDI preserves analogue timing; in particular, the Vertical Blanking Interval[1] (VBI) is maintained, as if video was still raster scanned. The whole production environment synchronises around the VBI. For example, in the analogue days, switching from one camera to another had to happen during the VBI, otherwise there would be visible artefacts in the output picture. SDI switches are of course digital, but they still operate on the VBI for consistency with analogue ways of working. The migration towards generic IT in TV production has been slow, in part due to the very tight time constraints imposed to maintain this synchronisation.

However, practices are changing, clearing the path for IP and Ethernet to replace SDI, and perhaps more importantly, for proprietary appliances to be replaced by generic computing capability. This would result in a shake-up in the broadcast and production industry, not only changing working practices, but potentially changing the ecosystem dramatically.

Arguably, the most critical aspect of migration to an all IP infrastructure in the broadcast and production industries is the preservation of timing. Indeed, despite the fact that current production standards maintain the VBI, the concept really no longer exists—that is, screen displays no longer perform a raster scan which requires a pause in the video signal to allow the raster to move from the bottom back to the top of the screen. However, the desire to preserve this concept in current working practices imposes major constraints on networking and video processing technology, which pose a major barrier to the adoption of generic IT systems in broadcast.

Here, we argue instead that accurate timestamping of source signals is required and that synchronised generation of video frames may also be required. With this approach, the delivery infrastructure no longer needs to maintain precise synchronisation around the VBI. Instead, the timestamps associated with the media are used to decide when to switch between streams, rather than using the system-wide wall clock. We refer to this as an asynchronous approach, since media delivery is not synchronised to the media timestamps. It is worth emphasising the difference, since with the conventional approach, the complete chain from camera to glass requires

[1]The Vertical Blanking Interval is a concept from the time when displays used a raster scan. It is the time allowed in the video signal for the scan to jump from the bottom back to the top of the screen. Screens of course no longer work this way.

complete synchronisation. In the IP world, this is no longer the case, as long as accurate timestamps are provided for all the elements that comprise the media.

We next introduce broadcast TV from a service perspective, since this forms the benchmark against which other services are judged, and discuss how it differs from newer IP-based technologies. Following this, we discuss sources and values of delay, and how these vary with network technology, which leads us to consider synchronisation and timing. We then suggest that IP would allow us to change the synchronisation model to an asynchronous one. The chapter concludes with an introduction to newer IP-based protocols that may displace HTTP for media streaming.

19.2 Broadcast as a Benchmark

From the early days of TV transmission, the UK has operated a Public-Sector Broadcasting (PSB) model, which we discuss below. The BBC was originally the sole UK broadcaster of TV and radio, funded from an annual licence fee. The licence model means that the BBC does not carry commercial advertising and has obligations to maintain political neutrality (unsurprisingly, a difficulty balance to strike), and to cover regional and minority interests. From the mid-1950s, commercial broadcasters began to operate alongside the BBC and were subject to similar obligations. Whilst the BBC remains a world-renowned institution (particularly for its news reporting), there is significant competition from Over-The-Top (OTT) providers, such as Netflix and pay-per-view channels. There are also perennial demands from certain sectors to abolish the BBC's licence fee altogether and to force the BBC to compete on a purely commercial footing.

The whole PSB model is certainly being stress-tested in the Internet age; however, it seems likely to continue in something like its current form for the foreseeable future.

Digital Terrestrial Television (DTT) remains the dominant mechanism for delivering the UK's PSB obligation. The UK PSB model (which is de facto adopted by many other countries [2]) for programming may be defined as:

- Universal geographic accessibility
- Universal appeal
- Attention to minorities
- Contribution to national identity and sense of community
- Distance from vested interests
- Direct funding and universality of payment
- Competition in good programming rather than numbers
- Guidelines that liberate rather than restrict

DTT differs from IP, satellite and mobile TV delivery in several respects:

- DTT is a specialised, broadcast-only technology. Generally, DTT receivers (TVs and STBs[2]) are really only optimised for that one purpose. In contrast, IP technology is application-agnostic and bidirectional.
- DTT (having evolved from decades of analogue TV) is a relatively simple plug-and-play experience. All DTT-TVs are able to receive the DTT service and the in-home wiring is standard, simple, and more than one TV or STB can be driven from a single aerial. IP delivery uses standard Ethernet cabling, or Wi-Fi in situations where there is good connectivity. Satellite connection is less straightforward, since it requires separate cabling for each TV or STB, because the STB needs to send command information back to the receiver dish.
- DTT as a service has a standard programme guide User Interface (UI), regardless of which brand or type of TV is connected, and viewers do not have to learn a new UI when changing TV. The satellite experience is different from DTT, but is at least consistent across a supplier's products (Sky, the UK's Freesat, etc.). However, adding Internet Protocol Television (IPTV) (either via apps on the TV, or additional devices or dongles) breaks this simplicity, with each device and app having different UIs and behaviour, making the experience more confusing for the viewer. Perhaps surprisingly, successive model-year TVs from the same manufacturer may have completely different app UIs. In this discussion, when we refer to IPTV, we include both multicast and also OTT unicast in the loose definition. As well as the technical differences between the two, there is also the service-level distinction, in that with multicast, the IPTV provider is also the broadband provider. In contrast, OTT is a more general Internet service, with programmes available in principal from any IP source: the TV (or STB) is less tightly bound to their ISP.
- DTT (along with satellite) is a monolithic environment: there is effectively a single provider, and whilst the content on the individual channels may be vying for viewers' attention, the platform as a whole is not. IP delivery, however, is more of a free-for-all, with apps and providers in more open competition; a programme may be available on the same TV from more than one source.

19.2.1 End-to-End Delay

When a live event is being broadcast, there is a desire to keep the customer experience as near to real time as possible. The most extreme form is where large screens are used at concerts and events to show the performance to people who can also see the actual performers at the same time. Here, virtually any delay will be apparent since the audience will be able to see both the live performers in the distance, and at the same time one or more large live screens. There is the further complication at large venues of the sound reaching the back of the audience later than the light. Given

[2]STBs (Set-Top Boxes), although of course modern TV design makes it impossible to place them on top!

that those at the back may see the action more clearly on the screens than the stage, the optimum for screen delay could in fact be equal to the sound delay, a tricky act to manage! People are more sensitive to non-causal delays where the sound arrives before the light, since in nature, this can never happen, whereas having the sound lag the pictures is more subjectively acceptable.

Leaving aside the edge case of live-at-event screening, "live" broadcast will typically mean trying to keep end-to-end delay to around 4–6 s. Our own ad hoc laboratory experiments suggest that the glass-to-glass delay of a linear broadcast is typically 3–4 s for over-air distribution and is approximately 40 s for streaming services such as iPlayer (the latter may be easily observed where the same channel is available via both iPlayer and DTT or satellite), and published experiments tend to corroborate these numbers, e.g. [3, 4]. Other streaming services can be a minute or more behind "live", and this delay may also vary slightly depending on network conditions.

Despite the span of delays, all of these are often described as "live" services. As an aside, some of us remember being in the audience at sporting events and listening to the commentary on analogue, long-wave radio whilst watching the action, with one complementing the other. Trying this with Digital Audio Broadcasting (DAB) radio is somewhat less satisfactory, since the digital radio commentary lags the live action by at least a couple of seconds.

As well as overall delay, the relative delays for different viewers watching the same content are important, particularly so for time-sensitive, popular events such as sport. It would be most unsatisfactory on a warm summer's evening to hear the cheering of your neighbours through your open windows just before you yourself see the goal (or miss).

With "catch-up" content, the delay relative to live is of course less important, since the viewer is, by definition, watching it at a potentially much later time of their own choosing. Delivery is further simplified since synchronisation between viewers is no longer important; the only synchronisation-related parameters in the catch-up case are start-up delay, and to prevent buffer underruns during playback. The types of delivery, latency and synchronisation are summarised in the Table 19.1, with illustrative values, although note that implementation details in the TV or STB could introduce further delays or reduce multiviewer or inter-destination synchronisation.

Note that "complete" in the table implies that all viewers perceive the same frame at the same instant, although there will of course be end-to-end delay. Also in the table, "fast-pull" means that HAS and VoD mechanisms generally pull media as segments, in bursts and gaps, with the bursts pulled as fast as the network will allow. They are buffered on the client and played out at the correct average rate. This pre-buffering allows network variations to be smoothed out, and the buffer fill state allows throughput to be more accurately estimated. The pre-buffering potentially facilitates faster channel change.

Various components in the contribution chain introduce delays. The typical processing pipeline for video transfer in a live event includes multiple compression/decompression and other media processing functions. The event location and studio are likely to be in different places, and the video from both is spliced at a

Table 19.1 Minimum illustrative latencies for types of delivery

	Concert (big screen)	Over-air (DTT/satellite)	IP (multicast)	IP (linear, unicast)	IP (VoD)
Delay from "live"	~500 ms	~4 s	~5 s	~40 s +	Not applicable
Synchronisation between screens	Complete	Complete	Complete	From complete to 1 segment duration	None
Channel change or stream start-up time	0	~200 ms to switch, however STB/TV performance typically ~5 s	~1 s to join stream, however STB/TV performance typically ~5 s	~1 s, providing sufficient bandwidth available to fast-pull first segments	~1 s, providing sufficient bandwidth available to fast-pull first segments

(third) location for playout where it is passed on for final end-consumer distribution encoding and delivery. An end-to-end chain can often include up to three encode–deliver–decode stages. At the contribution side (where the final edit and composition is made), the video is kept as high a quality as possible whilst overlays, fades and mixes are added, then passed on to a final distribution compression stage. It is worth noting here that dynamic advertising may be inserted into the stream after this stage, and this can be performed at the client application or close to the network edge.

For over-air broadcast via DTT or satellite, we would typically like to stay within an end-to-end delay budget of around 4 or 6 s. 1–2 s of this budget will be used by the encoding and multiplexing for broadcast, which leaves of the order of 1–2 s for images to get from the cameras to the broadcast encoders. There will also necessarily be some buffering in the STB or TV of at least a couple of seconds.

19.2.1.1 IP Multicast

A similar delay budget is achievable with IP multicast. In this context, IP multicast refers to an IP network that is capable of sending copies of User Datagram Protocol/Internet Protocol (UDP/IP) packets to multiple clients simultaneously. At a service level, this allows a server to send a single byte stream, and this stream could potentially be distributed to a single client, or to millions, with no additional overhead for the server. To perform the stream distribution, multicast-capable network switches detect multicast UDP packets and associated signalling, and, providing that a downstream switch or device has registered its interest in a particular stream, the upstream switch will replicate the multicast packets and forward them to the downstream device. This is a simplified explanation; there are several variants of multicast, operating at International Standards Organisation (ISO) layer 2 and also at

layer 3. There is also the concept of Application Layer Multicast (ALM), where the replication and management is performed by the application software, using conventional unicast UDP for transmission. Although this requires more complex application software, it does allow the applications to work on non-multicast networks.

Client devices register their interest in receiving multicast by using specific protocols. The most common of these are the ISO layer 3 Internet Group Management Protocol (IGMP) for IPv4 and Multicast Listener Discovery (MLD) with IPv6. There are several other protocols associated with management of multicast, and discussion of these is beyond the scope of this chapter. Further information may be found in RFC 4604 and associated standards [5].

Multicast is however not generally supported on the open Internet and is therefore mostly restricted to Internet Service Provider (ISP)-controlled networks. There are two key technical reasons for this.

Firstly, being UDP, there is no inherent Quality of Service (QoS), important for live video that has little buffering. Whilst UDP packets are likely to get through at the expense of Transmission Control Protocol (TCP), this cannot be guaranteed (nor can the packet ordering), and so to ensure a reliable TV service, multicast would typically be carried in a dedicated Virtual Local Area Network (VLAN) providing prioritisation and control. ISPs may also operate Active Queue Management (AQM) [6], where the network routers attempt to reduce congestion by pre-emptively discarding packets before the internal queues actually become full. There are various approaches available, some of which may impose rate-limiting on UDP traffic, therefore reducing UDPs inherent tendency to dominate TCP-based protocols.

Secondly, without management, multicast could swamp the network. This is mainly because, since it is point-to-multipoint, the switches in the network will replicate the packets to any downstream node that requests them via an IGMP join message. Furthermore, being UDP which lacks flow control, the packets will generally get through at the expense of other protocols such as TCP. This could be exploited by malicious clients, or defective software for Denial of Service (DoS) attacks; for example, a rogue application installed on home computers could join a multicast stream from thousands or millions of locations and, by joining a specific multicast, could flood the entire network with copies of that stream. Multicast is therefore blocked by most Internet switches and, if used at all, tends to be used within an ISPs own network, where the ISP has control over the nodes that can send, replicate and request multicast.

It is however a powerful and useful technology for distributing linear IPTV, and for example, BT in the UK delivers their linear TV using multicast in a QoS-enabled VLAN, with the bandwidth reserved through to the Residential Gateway (RG) such that when a client joins a channel, the network prevents that bandwidth from being used by other traffic. Multicast will have small additional intrinsic setup delays compared to unicast, since client STBs will need to issue IGMP join requests [5] to their upstream router. Depending on the complexity of the router tree, this may require further upstream requests and possibly other configuration changes before the multicast packets begin to arrive. To increase reliability, some multicast implementations employ Retransmission (RET) servers, which cache typically 3 s of programme data.

Multicast is usually encapsulated inside Real-Time Transport Protocol (RTP) [7] packets, and a multicast client can detect missing multicast packets by parsing the RTP headers for gaps in the sequence number contained in these packets. Upon finding a gap, the client may make a unicast request to the RET server for the missing packet, and (providing the loss can be recovered before playout time) can insert the packet from the RET server into the playout stream. Some clients may exploit RET servers to try to reduce the join delay by making immediate RET requests for packets at the same time as they issue the IGMP join request. The client can then stitch the multicast packets to the RET packets as the system stabilises. The additional traffic generated with this approach may however not be popular with the RET provider! A further complication arises if the content is protected by encryption. In this case, the STB must make an HTTP request to a server for the decryption keys, and we have observed channel change delays of approximately 6 s, attributable to this key retrieval and processing period. So whilst multicast is in principal able to change channel as quickly as DTT, there may be additional delays of a few seconds caused by the key acquisition.

19.2.2 OTT Streaming

OTT Internet streaming has more relaxed timing expectations, where a few 10 s of seconds' end-to-end delay is common. Much of this is introduced by the use of HTTP Adaptive Streaming (HAS) over TCP, where the client devices will often buffer 20–30 s of content. Whilst TCP guarantees delivery, it cannot reserve bandwidth, and this can vary dramatically. Having many seconds of buffering allows the media segments to be pulled by the client in advance of playout, and short-term restrictions in bandwidth can therefore be accommodated. HAS generally makes several versions of the same content available, encoded at different bitrates and qualities, in short segments of typically 10 s length. As segments are pulled by the client, it can measure the time taken to download them and the state of its buffers, and use this information to request higher or lower rate segments. Short-term bandwidth fluctuations can thus be handled with buffering, and longer-lasting ones by varying quality.

19.3 Standards and Timing

Both terrestrial and satellite TV services deliver programmes inside MPEG-2 Transport Stream (TS) [8]. These transport streams are designed specifically for broadcast TV requirements, although they can also deliver supplemental non-media payloads, such as electronic programme guide (EPG) data. TS is well-suited to broadcast because the important signalling, timing and other metadata are distributed and repeated throughout the stream at intervals of less than a second. This helps with fast channel join; a receiver can begin to decode the TS packets that comprise a pro-

gramme's video, audio and other data streams almost immediately. Data is carried in small 188-byte packets, with each video, audio or other type of information forming a distinct Packetized Elementary Stream (PES) series of packets. A programme comprises one or more PES streams; for example, a programme with one video and one audio track would have two PES streams. Within a TS, there may be multiple programmes, and there will be other metadata tables, plus in general, Forward Error Correction (FEC) packets. All these are interleaved, and since the key tables are frequently repeated, a receiver can join mid-stream and quickly determine the state and begin to decode a specific programme.

Timing is of course crucial for this to operate correctly, and TS sends a Programme Clock Reference (PCR) regularly in the stream [8], Sarginson. The PCR allows receivers to correctly align their timing with the sender and to correct for drift. Two other key timing elements are the presentation timestamp (PTS) and Decoding Timestamp (DTS), which are carried in the payload of TS packets. The PTS specifies the time at which an element should be presented to the viewer. The specification assumes that this can be performed instantaneously, although in practice, the transfer of data into the decoder and the decoding process itself is certain to take a finite time, and the decoder must compensate for this. For streams that carry video, further timing information is needed, and this is provided by the DTS. For efficiency, some MPEG video elements (known as Bidirectional (B) frames) may be deltas of previous and of subsequent other elements, and so we may not be able to decode a certain element until we have received others. This means that picture elements may be transmitted in a different order from their display order. The DTS tells the decoder which time (and therefore order) to extract the elements from the stream, so that they can be correctly decoded. If a stream contains B frames, then the DTS and PTS will differ, and the DTS will always be earlier than the PTS.

The discussion above focuses on the transport of encoded media, and its playout, essentially ensuring that elements come out of the pipeline in the correct order, and at the correct time. With TS, we are not simply dealing with individual frames of images, since the nature of the encoding causes image data to be spread across multiple frames, and it is this that the PTS/DTS timestamps are designed to handle. More details about timing issues in MPEG2-TS and other MPEG standards can be found in Chap. 10.

19.4 Timing and Asynchronous Production

19.4.1 Media Source Synchronisation

In a broadcast environment, a common Generator Locking (genlock) signal is provided to ensure that all cameras produce video frames synchronously. Synchronising sources in this way meets several requirements when switching from one video source to another:

- The inter-frame interval is preserved
- The frame on which switching occurs suffers little or no distortion
- All video media streams are switched at the same time (although other types of media, if present, would require additional synchronisation mechanisms)

Although these features appear very similar, it is worth bearing in mind that they are different, as when migrating to generic IT, they could be achieved by separate means.

19.4.2 Responsiveness

Responsiveness is the delay between making a control decision and seeing the result of it happen so that the system appears interactive. Typically, response delays of up to 100 ms are unlikely to be perceptible [10]. Latency for communications and other duplex links will be important; ideally, this should be kept below 150 ms (for example, ITU-T G.114 recommends a maximum of 150 ms one-way latency [11]).

19.4.3 Playout Synchronisation

Playout involves switching between various sources to make a complete channel. The channel could then be broadcast over air, or made available over the Internet. Timing within playout will involve mixing linear and nonlinear sources, each with its own clock. For each media source, playout will need to map from a media-specific clock reference to a global clock reference.

19.5 Asynchronous Infrastructure

We have seen that digital media timing operates on the concept of presenting elementary media samples at given times via the PTS timestamp, where, each media sample has a presentation timestamp (PTS) associated with it and is presented at this time according to a master PCR clock. Each elementary stream renderer would have access to this master clock in order to ensure rendering synchronisation over the different elementary streams.

In the asynchronous case, different elementary streams can arrive at different times at the various rendering devices. The rendering devices could even be physically separate, providing that they agree on the common clock. Synchronised rendering is achieved in this case by buffering those samples which arrive early.

This TS-based approach is a fundamentally different technique to use low latency as a means to achieve global synchronisation. Since elementary stream

synchronisation is achieved at the renderers, there is less need to be as concerned about latency, since synchronisation of the processing and switching elements is not necessary. It is still necessary to be concerned about end-to-end delay, but this is very different to the precision required to achieve VBI synchronisation throughout the delivery infrastructure, which, according to SMPTE-168, is of the order of 10 s.

Once we are released from concerns over maintaining intermediate synchronicity, we have the flexibility to allow different media streams to follow different paths, each possibly with a different latency.

It is fundamental to the design of packet networks that they introduce latency: the basic node architecture is of a queue of packets waiting to be dispatched. Latency can of course be kept extremely low by ensuring that this queue is kept empty, or that traffic requiring low latency bypasses other traffic using multiple queues or AQM, in conjunction with high-performance processing. However, this will require much higher performance networks with more complex management systems than would a higher latency network.

We now revisit some of the timing requirements to interpret them in the context of an asynchronous timing model.

Media Source Timestamps

This is even more critical in the asynchronous timing model than in the synchronous model. Unlike in the synchronous model where media passes through every component of the system close to real time, we cannot make the assumption that there is any useful relationship between capture time and real time at any of the intermediate nodes.

Media Source Synchronisation

In addition to assigning accurate timestamps, media source synchronisation requires that frames of video are produced synchronously. If this were not the case, then when switching from one source to another, the inter-frame interval could change for the two frames either side of the switch point. In principle, as long as each sample had an accurate timestamp assigned, it may be possible to work with variable inter-frame intervals and not require source synchronisation. However, this is likely to significantly increase the complexity of processing and rendering, so source synchronisation will still be necessary with an asynchronous timing model.

The IEEE-1588 Precision Time Protocol (PTP) [12] can be used to synchronise sources. For this to work accurately, the Ethernet switches involved must be PTP-aware. PTP data is very low bandwidth, relative to the bandwidth required to carry the media packets themselves, and PTP can be confined to just that part of the network required for synchronised sources. The latency requirements of PTP need not be imposed on the high bandwidth media streams.

Responsiveness

The delay between an action being taken and the effect of that action being seen needs to be kept below approximately 100 ms for it to appear instantaneous. The introduction of a few milliseconds of latency in the asynchronous model is therefore not significant, since other factors, such as the responsiveness of software involved are likely to dominate. Actions such as frame switching require waiting for a frame boundary. Since the interval between frames (at 50 fps) is 20 ms, the average frame switching delay must be at least 10 ms (half the inter-frame interval).

Playout Synchronisation

This will take place in a similar way to rendering. The playout engine will need to have a clock which runs at the same speed as the clock used to generate the media source timestamps. The playout engine could introduce a fixed delay between the source timestamps and the playout time. This would allow media samples with similar timestamps to be dispatched at similar times. This would reduce the rendering buffering that needed to be added.

When mixing different assets with different timestamps, the playout engine will need to modify these to allow the renderer to play them out smoothly. For example, archive assets will have timestamps which bear no relationship with real time. These will need to be adjusted to be consistent with the main playout stream.

End-to-End Delay

End-to-end delay is important from a user experience point of view. In the asynchronous model, delays are introduced intentionally to allow re-synchronisation of media streams at points where this is important (e.g. on the client device). The various components en route must work within a delay budget to ensure that end-to-end delay is kept acceptable.

19.6 An Asynchronous Future?

The truly asynchronous model sounds attractive, although whether or not it is adopted depends on many technical and commercial factors. Step change in any industry is disruptive, and the production and broadcast industries have vast experience with the status quo: there will need to be significant external reasons to move from the current, well-understood approach. The TV broadcast industry demands very high levels of reliability, and this tends to make them averse to any change that might compromise this. Change is threatening to domain experts!

Assuming the perceived threats can be overcome, allowing the infrastructure to be asynchronous reduces the requirements on networking equipment so that generic Ethernet switches and routers can be used. Equally importantly, the more relaxed timing constraints of an asynchronous infrastructure allow software to be used for

processing and manipulation, as an asynchronous infrastructure can deal with the relative unpredictability of execution time in most software services. The key requirement is that timestamps are preserved.

Therefore, a timing-tolerant system is a key enabler to allowing processes to run in software, and this in turn begins to pave the way for the greater use of cloud technologies and hosting some services by third parties. This has the potential to dramatically change the media distribution industry.

One of these changes will be the ability to produce more flexible media assets easily. In the asynchronous IP world, all the elementary media streams that constitute an asset would share a common timing system, but would not have to be delivered together, that is, they do not all have to pass over the same SDI cables. Subtitles, metadata, etc., can all be sourced and distributed relatively independently.

An asynchronous infrastructure would also allow the standards for the different media types to evolve independently, since it would no longer be a requirement that every new piece of data also has a SMPTE standard to specify how it would be encapsulated in SDI and sit alongside all other ancillary data.

19.7 Future Transport Protocols

Ideally, we should be able to carry data (which may be video) over any network; fixed or mobile, as multicast or unicast, and be able to handle dynamic handover from one to another. This should be an internal network decision, not one that the Content Service Provider should be directly concerned with.

Currently, terrestrial and satellite TV services deliver MPEG-2 TS as the lowest layer of their protocol stack. These transport streams are designed specifically for broadcast TV requirements and whilst it is possible to deliver non-media payloads such as EPG data, these are intended to supplement the TV service.

In contrast to this, Internet protocols tend to be generic. HTTP now dominates as the protocol of choice for non-conversational services. Even services such as video streaming which would previously have used a specialised protocol now use HTTP. However, HTTP is designed for downloading files, and its underlying TCP transport protocol means that although the files are guaranteed to arrive intact, the bitrate and delay are highly unpredictable, and this unpredictability does not make HTTP a good candidate for a video streaming protocol. It is possible to argue that, whilst HTTP can be used for almost all (non-conversational) services, it is also sub-optimal in most cases.

So why is HTTP ubiquitous? It is attractive because it is generic and reusable. Content Delivery Networks (CDNs) need only implement the one protocol stack, and, whilst performance may be tuned according to the mix of services, fundamentally the use of HTTP dramatically simplifies network infrastructure. Further, reuse of the single protocol stack means that all services benefit from its capabilities. For example, Transport Layer Security (TLS), caching, etc., can be used "for free" by any service that uses HTTP. Additionally, HTTP-based protocols will traverse standard

firewalls and survive Network Address Translation (NAT), whereas more esoteric protocols may not by default.

The unpredictable throughput and the concatenation of files that represent the media segments into a single stream are dealt with at the connection end points. For video streaming, for example, a combination of increased buffering on the client devices and adaptive bitrate streaming reduces the occurrence of stalling and therefore permits a sufficient degree of quality control. The question then arises: is there a wider advantage to replacing specialised TV and video distribution protocols with Internet-type protocols, and how could this be done? There are a number of advantages to using a non-specialised protocol stack:

- It is much easier to carry any data efficiently, not just media.
- The same protocol stack could be used for different network types (fixed, mobile, over-air broadcast).
- A common protocol stack will facilitate multicast carriage of otherwise unicast data and will facilitate dynamic switching between multicast and unicast.
- A common protocol stack will facilitate switching between network types (e.g. between fixed and mobile).
- If the protocol is designed to carry HTTP-like data, then it can easily interface with content service providers, end devices and network caches or CDNs.

HTTP is of course a bidirectional, unicast protocol and relies on a reliable transport protocol underneath. One-to-many networks are not inherently reliable, and some such as terrestrial broadcast have no associated upstream path. The key emerging technical challenge for mass delivery of media is to make a non-specialised protocol stack work over both multicast and broadcast networks, as well as IP and mobile ones. It is not cost-effective to put relatively complex CDN nodes at or close to the edges of the fixed network, and with mobile radio-based networks, we need over-air broadcast for high-volume efficiency. Therefore some form of multicast or broadcast will always be required for efficient, simultaneous delivery of data in the access network.

It is worth stressing here that HTTP is a client-based and pull-based protocol, whereas multicast is really server-based and push-based. If we are to use an HTTP-like protocol for broadcast, then we need to somehow aggregate the multiple client-pull requests. This is important for two main reasons: firstly, from a service delivery perspective, it is less efficient to handle many simultaneous requests for the same data than it is to serve one copy that is subsequently replicated within the network. Secondly, people understand the term broadcast to mean that everyone receives the same material at the same instant, and we therefore want all clients to receive the same bytes at the same time.

It is worth reflecting here that as broadcast has moved from terrestrial analogue TV over air, through digital terrestrial TV over air, the reception has generally become less instantaneous because of encoding, transmission and decoding delays. Similarly, satellite TV adds additional delay. However in these cases, all receivers using the same generation of technology will receive the same pictures at the same

instant (at least as far as viewers can perceive), since the signal between transmitter and receiver travels at the speed of light and arrives at the same instant at all receivers, and two neighbouring houses will see the event at exactly the same time. It is of course possible for different suppliers STBs or TVs to implement internal buffering that introduces small variation.

However when moving to IPTV, this is no longer guaranteed, since the information is now carried over IP networks, with varying numbers of nodes, switches and congestion. Multicast over IP will be completely synchronised for receivers in adjacent houses; however, unicast HTTP will not unless additional steps are taken.

It is certainly possible to aggregate multiple requests by using network caching; however, doing so introduces synchronisation delay. There will be commercial implications here, since the content provider may be unwilling to accept this additional delay without some incentive such as lower costs.

In summary, there are separate protocols and different networks that are optimised for their specific domains, without an overarching single scheme, and IP appears to be the natural underlying protocol if we are to unify these differences. We now briefly discuss technology and standards which can contribute to a solution to these challenges. In particular, we introduce two newer IP protocols that are interesting in this area, since they build on the strengths of IP and help to reduce some of the issues around traditional HTTP.

19.7.1 HTTP/2

The HTTP/2 protocol has been developed as RFC-7540 [13] to address some of the perceived shortfalls of HTTP. HTTP/2 is now supported in the common browsers and Web servers, and as of May 2017, approximately 14% of Web site traffic is HTTP/2 [14]. At a high level, HTTP/2 compared to HTTP/1.1:

- Is binary, instead of textual.
- Is fully multiplexed, instead of ordered and blocking (and streams can be prioritised).
- Can therefore use one connection for parallelism.
- Uses header compression to reduce overhead.
- Allows servers to "push" responses proactively into client caches.

It is therefore more efficient in its use of the network, and its push support is conceptually nearer the multicast "push" model. Its multiplexing allows media elements to be prioritised within a single TCP connection, and therefore, providing the bytes themselves arrive at the client, the server can ensure that important control and layout information should arrive at the client before less-important data. In a Web page, for example, it is better to receive the CSS and layout code before the images and so allow the layout engine in the client the opportunity to arrange the page, rather than possibly having to wait with buffered images until the layout instructions arrive.

From our video distribution perspective, there are no disadvantages with HTTP/2 compared to HTTP/1,[3] and (whilst still not being directly multicast-compatible) it is a better scaling candidate than HTTP/1. It is also being adopted now, and the trends indicate that it will overtake HTTP/1 within a few years.

19.7.2 QUIC

Google and the IETF are developing their Quick UDP Internet Connection (QUIC) protocol [15] which, although using UDP instead of TCP, shares many of the design philosophies of HTTP/2. Both HTTP/2 and QUIC are growing in popularity and are likely to become mainstream, and looking forward, these will displace HTTP/1. QUIC is UDP-based, and therefore, packet dispatch and congestion are handled in user-mode QUIC software rather than in the operating system kernel, since UDP does not itself provide these dispatch, queue and reassembly functions. Since the behaviour of the packet delivery is handled in application code, it becomes much easier (in principle) to take into account application requirements, such as latency. This is because kernel-mode software such as the TCP stack has, by its nature, to support all applications equitably and is unable to take account of the specific delivery requirements of sensitive ones such as video delivery. Furthermore, kernel-mode components are complex and slow to evolve, and absolute reliability is the top priority, since failure in the kernel will crash the whole machine. User-mode code lacks this reliability, but the consequence of failure is generally just a single application crash. Being in user-mode, the application developers can tune their QUIC/UDP stack to optimise it to meet the needs of their specific application, and the code itself can be modified in much shorter timescales than if it were part of the kernel.

Both of these newer protocols are particularly interesting for media delivery since they enhance the server-push model with the PUSH_PROMISE option. This allows the server to tell the client in advance about important elements; the client can allocate resources and may be able to make more reasoned decisions about its subsequent element requests if it's told of their impending arrival. The client is free to either accept the promised elements or can decide to tell the server not to actually send them.

Stream prioritisation is another important component, since it allows the server to send the more vital parts, such as Cascading Style Sheet (CSS) elements and scripts, before the presentation components such as images or video. Once again, this can benefit the client since it can decide what to do before the content elements arrive, as opposed to receiving them and then having to wait to decide what to do with them. As of 2017 however, there are unanswered questions around how these newer protocols will evolve. QUIC, for example, uses a connection ID rather than the lower-level

[3] HTTP/2 strongly recommends encrypting the payload, which might displease network operators wishing to perform packet inspection or traffic shaping. However, HTTP/1 traffic is increasingly being carried over encrypted TLS links in any case, so this point may be moot.

IP address/port combination to identify endpoints (the rationale being that a session could handoff from one network to another, changing IP address but maintaining higher-level state). This connection ID is defined by the client, not by the server. Unfortunately, this is the wrong way round for multicast, where the server pushes data to clients that may or may not exist. QUIC will need to develop some technique to address this, allowing a server to specify the connection ID, if it is to be truly capable of delivering multicast.

Another area ripe for study is how video will be delivered over 5G mobile networks (and at the time of writing, it is far from clear what a 5G network actually will be). 5G will support both multicast and unicast delivery of high-bitrate video and, like fixed networking, proposes significant challenges in unifying them. Perhaps the ultimate synchronisation opportunity is to define formats and technologies that would allow compelling programming to be sent over both fixed and mobile networks, by either unicast or multicast, with all these routing decisions left wholly to the network, and with the viewer being unaware of any changes in delivery.

19.8 Conclusion

Traditional TV production and distribution has been refined over several decades to deliver excellent programming, both creatively and technically. The story is one of evolution, first from black & white to colour, and later from analogue to digital. However, these changes are also disruptive: the traditional benevolent broadcaster has largely given way to content from newer commercial organisations, and technological evolution has allowed terrestrial, satellite, and then IP and latterly mobile to all become carriers for similar TV-like services, using different underlying technologies.

Evolution does not stand still: IP is still being refined, and the standards for broadcast over mobile/5G, in particular, are currently immature. The techniques used in traditional TV are bespoke and expensive. Emerging IP-based protocols and the lower costs of generic computing are enablers for a future where programmes can be produced, distributed and repurposed more simply and cheaply. In a world where the content must be delivered to many types of devices, over fundamentally different kinds of networks, and when the distinction between live and catch-up is blurred, generic protocols that support precise timing synchronisation present interesting and in some cases compelling alternatives. Furthermore, developing these protocols for diverse types of networks and at the same time reducing costs provides major challenges and, of course, significant opportunity.

References

1. SMPTE.: Transport of high bit rate media signals over IP networks. ST 2022–6:2012 (2012). https://doi.org/10.5594/SMPTE.ST2022-6.2012
2. E.g. Raboy, M.: Public broadcasting for the 21st century, p. 610. Indiana University Press (1995). ISBN 1-86020-006-0
3. Kooij, W., Stokking, H., van Brandenburg, R., de Boer, P-T.: Playout delay of TV signals: measurement system design, validation and results, ACM TVX 2014, Newcastle, UK (2014)
4. Boronat, F., Mekuria, R., Montagud, M., Cesar, P.: Distributed media synchronization for shared video watching: issues, challenges, and examples. In: Ramzan, N., van Zwol, R., Lee, J.-S., Clver, K., Hua, X.-S. (eds) Social Media Retrieval, Springer Computer Communications and Networks Series, pp. 393–431. Springer, London, UK (2013)
5. Holbrook, H.: Using internet group management protocol version 3 (IGMPv3) and multicast listener discovery protocol version 2 (MLDv2) for source-specific multicast (2006). https://tools.ietf.org/html/rfc4604
6. See e.g. https://en.wikipedia.org/wiki/Active_queue_management as a starting point
7. Schulzrinne, H., et al.: RTP: a transport protocol for real-time applications. Internet Standard (2003). https://doi.org/10.17487/RFC3550
8. ISO/IEC.: Generic coding of moving pictures and associated audio information: systems. International Standard 5th Edition ISO/IEC 13818 1 (2014)
9. Sarginson, P.: MPEG-2: overview of the systems layer. BBC R&D Report (1996). http://www.bbc.co.uk/rd/publications/rdreport_1996_02
10. Nielsen, J.: Usability engineering (Interactive Technologies), chapter 5 (1994). ISBN-13: 978-0125184069
11. ITU-T.: One-way transmission time. Transmission systems and media, digital systems and networks, Series G (2003)
12. IEEE standard for a precision clock synchronization protocol for networked measurement and control systems (2008). https://doi.org/10.1109/IEEESTD.2008.45797.60
13. Belshe, M.: Hypertext transfer protocol version 2 (HTTP/2) RFC 7540 (2015). ISSN: 2070-1721
14. W3Techs—World Wide Web technology surveys. https://w3techs.com/technologies/details/ce-http2/all/all data Copyright (C) 2009–2017 Q-Success DI Gelbmann GmbH. Accessed May 2017
15. Bishop, M.: Hypertext Transfer Protocol (HTTP) over QUIC. IETF draft (2017)

Chapter 20
Camera Synchronization for Panoramic Videos

Vamsidhar R. Gaddam, Ragnar Langseth, Håkon K. Stensland, Carsten Griwodz, Michael Riegler, Tomas Kupka, Håvard Espeland, Dag Johansen, Håvard D. Johansen and Pål Halvorsen

Abstract Multi-camera systems are frequently used in applications such as panorama videos creation, free-viewpoint rendering, and 3D reconstruction. A critical aspect for visual quality in these systems is that the cameras are closely synchronized. In our research, we require high-definition panorama videos generated in real time using several cameras in parallel. This is an essential part of our sports analytics system called *Bagadus*, which has several synchronization requirements. The system is currently in use for soccer games at the Alfheim stadium for Tromsø IL and at the Ullevaal stadium for the Norwegian national soccer team. Each Bagadus installation is capable of combining the video from five 2 K cameras into a single 50 fps cylindrical panorama video. Due to proper camera synchronization, the produced panoramas exhibit neither ghosting effects nor other visual inconsistencies at the seams. Our panorama videos are designed to support several members of the

V. R. Gaddam · H. K. Stensland · C. Griwodz · M. Riegler · P. Halvorsen (✉)
Simula Research Laboratory, Fornebu, Norway
e-mail: paalh@simula.no

V. R. Gaddam
e-mail: vamsidhar@simula.no

H. K. Stensland
e-mail: haakonks@simula.no

C. Griwodz
e-mail: griff@simula.no

R. Langseth · T. Kupka · H. Espeland
ForzaSys AS, Fornebu, Norway
e-mail: ragnar@forzasys.com

T. Kupka
e-mail: tomas@forzasys.com

H. Espeland
e-mail: haavares@forzasys.com

D. Johansen · H. D. Johansen
UiT – The Artic University of Norway, Tromsø, Norway
e-mail: dag@cs.uit.no

H. D. Johansen
e-mail: haavardj@cs.uit.no

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_20

trainer team at the same time. Using our system, they are able to pan, tilt, and zoom interactively, independently over the entire field, from an overview shot to close-ups of individual players in arbitrary locations. To create such panoramas, each of our cameras covers one part of the field with small overlapping regions, where the individual frames are transformed and stitched together into a single view. We faced two main synchronization challenges in the panorama generation process. First, to stitch frames together without visual artifacts and inconsistencies due to motion, the shutters in the cameras had to be synchronized with sub-millisecond accuracy. Second, to circumvent the need for software readjustment of color and brightness around the seams between cameras, the exposure settings were synchronized. This chapter describes these synchronization mechanisms that were designed, implemented, evaluated, and integrated in the Bagadus system.

Keywords Camera array · Panorama video · Frame stitching · Shutter synchronization · Exposure synchronization

20.1 Introduction

Media synchronization is certainly not a new research area. Over the last decades, various mechanisms have been proposed, covering a wide span of usage scenarios. In our research on the Bagadus system [1], we developed a camera array system for real-time video panorama recording. To generate high-quality wide field-of-view panoramas, Bagadus ensures that multiple components between the back-end processing machines and cameras are accurately synchronized in various ways. For instance, frame-accurate synchronization is required to avoid errors around moving objects, different colors, and luminances. As Bagadus aims at running live in real-world scenarios, real-time performance and a perceptually good video quality are key requirements. This chapter addresses mainly the challenges we faced related to synchronization of cameras to achieve our video quality goals.

Our first challenge was related to performance and image consistency issues when handing camera synchronization in software. As Bagadus was developed to capture soccer games, the objects it records often move at speeds that demand accurate camera trigger synchronization. To avoid parallax errors and ghosting effects due to moving objects captured in the seam region by several cameras, the cameras must be shutter-synchronized with sub-millisecond accuracy. In this respect, the fairly widespread software approach to handling camera synchronization, which consists of measuring the cameras' temporal drift to sort the drift-compensated frames before panorama rendering, fails to solve the consistency problem. Experience with software-triggered exposure suffered from operating system scheduler limitations. Our solution was to add an external trigger signal that is broadcast synchronously to all cameras.

Our second challenge was related to the inconsistent visual quality we initially observed across different cameras. It is important for the visual quality of panorama videos that there is a consistent brightness and color balance across the pixels that are captured by the individual cameras. We can install identical color lookup tables on all cameras, but we cannot fix exposure. Since Bagadus is deployed outdoors in a geographic region with unstable weather conditions, a constant exposure setting would yield intervals of over- or underexposed frames. Likewise, we cannot use auto-exposure on all cameras. Since auto-exposure relies on a brightness histogram, and all cameras see different parts of the field without a common overlap, each camera will make independent and different exposure decisions, both in case of a full-frame histogram and a histogram for a region of interest (ROI). For Bagadus, we therefore developed a solution where the cameras use one pilot camera that performs auto-exposure. Its exposure settings are then read by host software and copied to all other cameras. This achieves very good exposure synchronization while adapting to most external conditions.

20.2 The Bagadus Sport Analysis System

Bagadus [1–3] is a sports analysis systems for real-time panorama video presentation of sport events. The system is currently installed in two stadiums in Norway: Alfheim stadium in Tromsø (Tromsø IL) and Ullevaal stadium in Oslo (the Norwegian national soccer team). An overview of the architecture and interaction of the different components is given in Fig. 20.1. As shown in the figure, Bagadus consists of three main sub-systems: a player tracking system, a coach annotation system, and a video system.

Using these sub-systems, Bagadus implements and integrates many well-known components to support the selected sport application scenario where the combination of different technologies raises requirements on both spatial synchronization and temporal synchronization of multiple signals from independent sources. The novelty lies in the combination and integration of components enabling automatic presentation of video events based on the positional sensor and analytics data that are synchronized with the video system. For example, Bagadus supports automatic presentation of video clips of all the situations where a given player runs faster than 10 miles per hour or when all the defenders were located in the 18-yard penalty box [4]. Furthermore, we can select and follow single players and groups of players in the video, and retrieve and repeat the events annotated by expert users. Thus, where people earlier used a huge amount of time for analyzing the game manually, generating video summaries, and making long reports, Bagadus serves as an integrated system where the required operations and the synchronization with video are managed automatically.

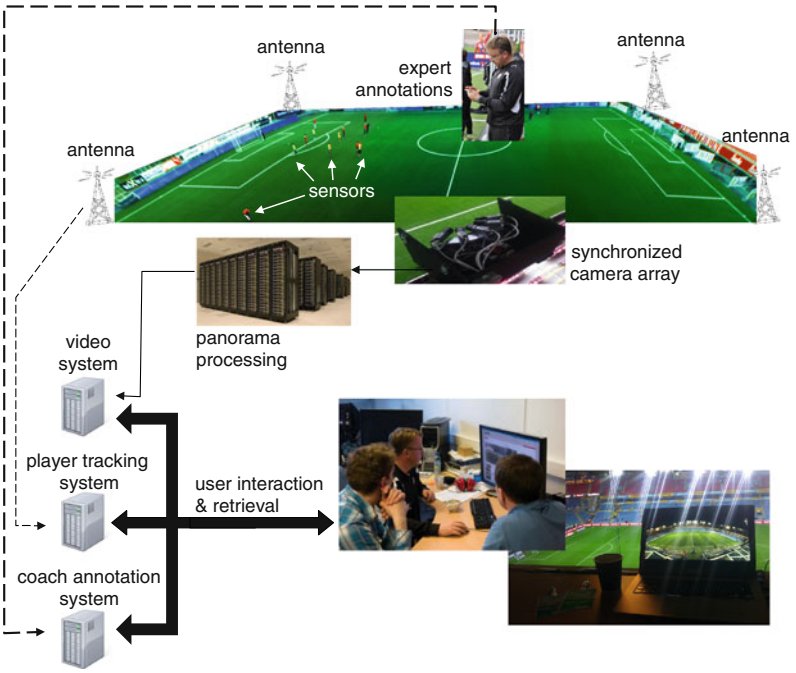


Fig. 20.1 Architectural overview of the Bagadus system

20.2.1 Player Tracking System

There exist several player tracking systems based on, for example, GPS, radio, or optical technologies. Bagadus potentially can use any type of tracking, but in the current installation, the *player tracking system* imports player positions from the ZXY Sport Tracking system [5]. ZXY is designed for fixed installations using advanced radio technology for both its positioning and data communication, and it collects several objective data elements including the players' positions typically at 20 or 40 Hz.

In Bagadus, the player positions are used not only to gather player movement statistics, but also to highlight players in the videos and to extract video clips based on player movement. For this, the time stamps of the video clips and the time stamps of the sensor data must be synchronized. Fortunately, this is not a very hard synchronization problem since the clocks on all machines are synchronized using the network time protocol (NTP) [6]. The clock synchronization accuracy provided by NTP is sufficient for aligning positional data with video clips. This synchronization issue will therefore not be covered further in this chapter.

20.2.2 Coach Annotation System

The *coach annotation system* [7] replaces the traditional pen-and-paper annotations that coaches previously had to (and still) rely on for annotating soccer games. Using the Bagadus mobile app, coaches can annotate the video quickly with a press of a button. The system is based on hindsight recording where the end of the event is marked, and the system then captures video prior to this mark. Furthermore, the registered events are stored in an analytics database that can later be shown along with the corresponding video of the event. Here, the time of the tagged event and the video must be synchronized, and the mobile device therefore synchronizes its local time with the NTP machines. However, it is only a fairly loose second-granularity time-sync requirements, and hence, it is not further discussed here.

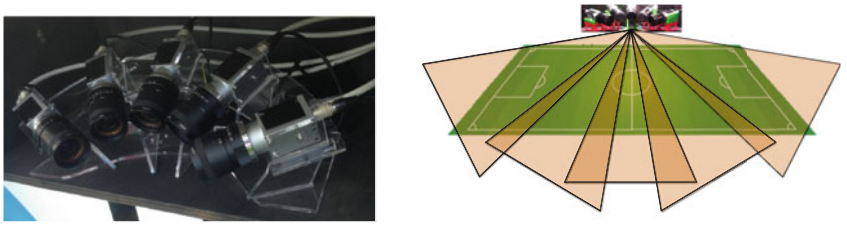
20.2.3 Video System

The *video system* consists of multiple small shutter and exposure synchronized cameras that record high-resolution video of the soccer field. The cameras are set up to cover the full field with sufficient overlap to identify common features necessary for camera calibration and panorama frame stitching. The frame stitching is based on a homographic mapping of each camera image into the panorama. The mapping matrices for each camera are statically generated through an offline calibration process. Nevertheless, the shutters and exposures of each camera must still be coordinated and dynamically adjusted.

20.2.3.1 Camera Setup

To capture the entire soccer field, Bagadus uses five 2K Basler industry vision cameras [8], each delivering a maximum resolution of 2046 × 1086 pixels at 50 frames per second (fps) over Gigabit Ethernet. We use an 8-mm lens [9] with virtually no image distortion and avoided having to apply lossy debarreling operations. To maximize the panorama resolution, the cameras are rotated by 90°, giving a per-camera field-of-view of 66°.

The cameras are statically mounted in a circular pattern, each covering a different part of the soccer field, as shown in Fig. 20.2. The cameras are pitched, yawed, and rolled to look directly through a common point about 5 cm in front of the lenses in an attempt to reduce parallax effects. As a result, only minor adjustments are required before the images can be stitched together into the panorama frame.



(a) Camera rig **(b)** Overlapping camera capture regions

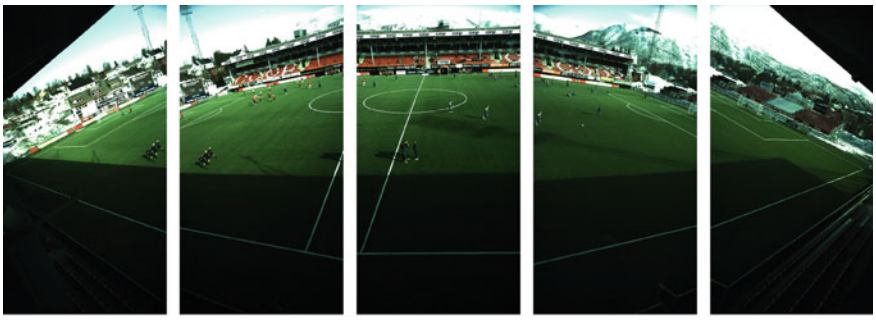
Fig. 20.2 Camera setup at the stadiums

20.2.3.2 Video Processing Pipeline

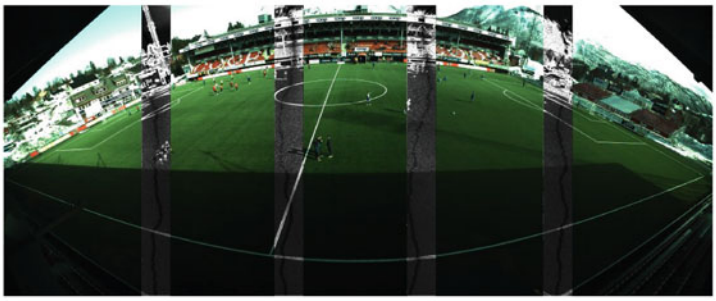
To support live video feeds, our camera output is processed through a custom real-time video pipeline [10, 11]. In its basic configuration, Bagadus' video pipeline consists of a background subtractor, a warper, a color corrector, a stitcher, and an encoder. The pipeline can also be configured to include YUV and RGB converters, Bayering modules [12], a debarreler, and a high dynamic range (HDR) module [13]. Most pipeline components can utilize both the central processing unit (CPU) and the graphics processing unit (GPU) [14, 15] for high-performance data processing. Figure 20.3 illustrates the data processing used in our two Bagadus deployments, with samples of individual recorded video frames as shown in Fig. 20.3a. Figure 20.3b depicts how these frames are stitched together in the overlapping regions by a dynamically calculated seam. The final panorama video (see Fig. 20.3c) has a frame resolution of 4096×1680 pixels and is encoded and compressed with the x264 encoder [16] into the H.264/MPEG-4 Advanced Video Coding (AVC) compression format.

20.2.3.3 Interactive Pan-Tilt-Zoom

When encoding video, we prioritize video quality over compression. As this leads to high demands on available network bandwidth to transfer the full-resolution panorama video, we explored the use of personalized views to reduce bandwidth requirements. Personalized interactive Pan-Tilt-Zoom (PTZ) operations that lead to server-sided encoding achieve lower bandwidth consumption by transcoding the panorama to a final view at a resolution appropriate for the receiver [17]. Moreover, personalized views for a large number of interactive receivers can be constructed from tiles that are stored in several representations of different quality, allowing each receiver to stream high-quality sub-streams (tiling) only for a region of interest [18, 19]. This means that a client only retrieves the video streams (tiles) at high resolutions that are relevant to the user experience. The other tiles can be either retrieved in low quality or omitted completely. Such tiling is also possible using the Spatial Relationship Description (SRD) extension [20] of the Dynamic Adaptive Streaming



(a) Alfheim stadium individual camera frames



(b) Alfheim stadium stitching areas

(c) Alfheim stadium generated panorama

(d) Ullevaal stadium generated panorama

Fig. 20.3 Panorama examples from Alfheim stadium and Ullevaal stadium

over HTTP (DASH) standard, and combined with the new tiling feature of the High Efficiency Video Coding (HEVC) standard [21], a tiled video can be stitched in the compressed domain and decoded on a single CPU while switching tile quality based on the active region of interest. Nevertheless, the quality (an invisible seam) of the original panorama, tiled or not, is of high importance as the users may dynamically move their regions of interest across the seams between the cameras.

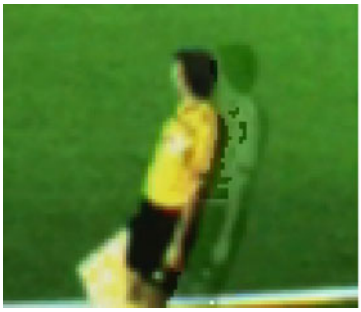
20.2.4 Synchronization Challenges

As outlined in this section, we encountered several synchronization issues while building and deploying Bagadus in order to ensure a high-quality panorama video. Synchronization of video frames with data from the positional and event annotation systems was trivially overcome by having hardware clocks synchronized using NTP. The sub-second accuracy provided by NTP [22] is sufficient for this purpose. We did, however, also encounter two additional and non-trivial synchronization challenges, both directly related to the visual quality of the final panorama video. These were: (1) *shutter synchronization*, which is required to avoid that moving objects appear at different positions in the corresponding frames from different cameras; and (2) *exposure synchronization*, as the cameras have different metering points for auto-exposure potentially resulting in mismatching lighting levels in the various cameras. In the remainder of this chapter, we focus on how Bagadus overcomes these two challenges.

20.3 Camera Shutter Synchronization

To ensure an errorless generation of panorama frames with respect to parallax errors stemming from moving objects, we need a synchronization signal that triggers the camera shutters. In this section, we show the potential problems of parallax errors stemming from lack of camera trigger synchronization. We then describe our solution and show how we have solved the problem using external trigger signals.

The initial processing stages of the Bagadus pipeline are processed independently, making use of three computers and multiple threads on each of them. This is necessary to deal with the bandwidth of five gigabit Ethernet-connected cameras. The three machines are running in the same local area network and synchronized by NTP to the same time server, which ensures a fairly accurate clock synchronization and suppresses drift [22]. To keep track of frames from different streams that contribute to the same panorama frame, we write a local Unix time stamp into the header of each frame as it is retrieved from the camera. This time stamp is maintained throughout the processing modules, but otherwise ignored until the frame is rendered into the panorama. At that point, we read the time stamps and use them as a barrier before writing the panorama. We consider two time stamps t_1 and t_2 equivalent if they are



(a) Ghosting from feathering (weighted average between the two overlapping images)

(b) A parallax error where a player appears on both sides of the static vertical seam just selecting pixels from the cameras on both sides

Fig. 20.4 Ghosting effects using different stitching techniques

closer in time relative to the video's frame rate. Given a video with an f Hz frame rate, we have that:

$$t_1 \equiv t_2 \implies |t_2 - t_1| < \frac{1}{2 \times f}$$

To ensure that all moving objects are located in the same place when different cameras record corresponding frames, the camera shutters need to be synchronized. If they are not, the frames can be captured any time within the time interval for a given frame set, and even small deviations can be seen when soccer players move through the stitching seam. As an example, consider an athlete running 100 m in 10 s. If we are running our video recorder at 25 fps, the maximum time shift between frames is 40 ms, and the athlete would move 40 cm per frame.[1] At that speed, there will be a highly visible ghosting effect in the generated panorama frames, as shown in Fig. 20.4. In particular, Fig. 20.4a depicts a stitch where we have used a simple weighted average between the two overlapping images which are out of sync. Figure 20.4b shows a similar example where the player is visible on both sides of the seam.

To create a smooth transition between the stitched frames in the panorama, each source frame must be captured at the exact same time. Even small deviations can, as we illustrated above, be seen when soccer players move through the stitching seam. An important part of the frame synchronization operation is therefore to make sure that the frames are captured simultaneously. After evaluating existing approaches, we ended up with a solution where the cameras are triggered externally. The approaches for this can be divided into two main classes: software-based triggers and hardware-based triggers.

[1] This is slightly slower than the 100-m sprint world record from 2009 by Usain Bolt of 9.58 s [23]. Bolt's movement between frames would be approximately 42 cm. The speed of the Ronny Heberson's free kick [24] was clocked at a whooping 221 km/h, which would result in a difference of 2.5 m between frames from different cameras.

20.3.1 Software-Based Triggers

Earlier work on shutter synchronization has to a large extent focused on extracting the temporal relation of frames in a software post-processing step and using those relations to recreate a temporally consistent scene. Hasler et al. [25] rely on the cameras' built-in microphone to capture sound and use the audio signal to align video frames. An important problem is that audio samples are usually not time-aligned to sample accuracy but only roughly to frame accuracy. Pourcelot et al. [26] considered post-processing based on the timing of a flickering light source recorded with a shutter speed of 1000 Hz. Also, Shrestha et al. [27, 28] use flashes to synchronize individual camera timelines. Ruttle et al.'s approach of a spinning disk [29] provides high accuracy with less visual disturbance. Bradley et al. [30] go far beyond this by even removing motion blur by lighting a scene up with stroboscopic light. We found this kind of method not applicable in practice in a soccer stadium because light sources are either obstructing the field-of-view, overexpose the entire frame, or become indiscernible from lighting change in the recorded scene. In the literature, the accuracy achieved by rough clock synchronization is frequently considered acceptable for both panorama stitching and 3D reconstruction. Duckworth et al. [31] and Moore et al. [32] have illustrated the inaccuracy that is introduced if a 3D reconstruction is undertaken from unsynchronized frames. Consequently, synchronization should be the first step in 3D reconstruction. Synchronization from silhouettes has received quite a bit of attention [33–36]. Haufmann et al. [37] integrate synchronization into 3D volume carving by minimizing variations in epipolar geometry. Nischt and Swaminathan [38] extract 3D points and minimize the variation in the fundamental matrix between cameras. Shankar et al. [39, 40] sort frames by minimizing the variance between trajectories of scale-invariant feature transform (SIFT) feature points, Tao et al. [41] use point triplets, and Velipasalar and Wolf [42] use the trajectories of bounding boxes around extracted objects, while Whitehead et al. [43] use textured objects. These approaches are not compatible with the Bagadus camera setup. All of them require camera positions that allow reasonably accurate 3D localization of points. Our cameras, however, are arranged to allow panorama creation by 2D homographic transformation only, which requires (roughly) identical camera centers (i.e., a baseline of zero). Obviously, that voids all attempts of 3D reconstruction.

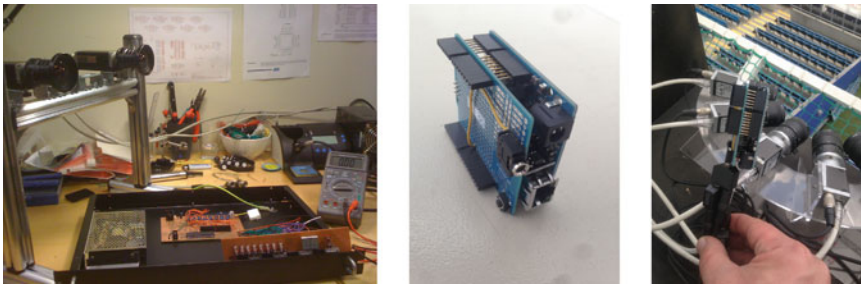
20.3.2 Hardware-Based Triggers

After evaluating existing software-based trigger solutions for Bagadus, we selected a hardware approach where the cameras are triggered externally. In this respect, several existing systems use hardware triggers. Genlock [44] is a classical method for synchronizing cameras in the world of broadcast entertainment; i.e., the video output of one source, or a specific reference signal from a signal generator, is used to

synchronize other television picture sources together. Smith [45] listed it as a basic requirement for video support in athlete training. Collins et al. [46] used it to acquire multi-view video. Alternatively, a shutter synchronization circuit for stereoscopic systems (RS232) was proposed [47] where the switching of the LCD shutter and the vertical synchronization scanning signal of the VGA graphic card can be synchronized. Blue-C used an unspecified hardware synchronization method [48]. Wilburn et al. [49] carry the synchronization signal over Firewire to an array of camera sensors and redistribute it via Ethernet (CAT5) cables in the array. A non-standardized method uses the transmission of the Society of Motion Picture and Television Engineers (SMPTE) time codes over wireless networks such as Wi-fi [50]. Litos et al. [51] constructed a PCB to refine the NTP-based rough synchronization of computers with Firewire cameras by recording and evaluating a high-resolution clock constructed from LEDs. Obviously, this yields very accurate knowledge of synchronicity, but it relies on the computer's operating system and camera implementation for synchronous recording. Sousa et al. [52] use the power supply itself to synchronize self-timed cameras attached to a single field-programmable gate array (FPGA) platform. Nguyen et al. [53] built a triggering circuit using an Arduino board to synchronize a structured light projector with a camera. Such hardware triggers seem to be the best solution for Bagadus, and we therefore built our own similar solution to meet the system requirements.

20.3.3 The Bagadus Shutter Synchronization

For Bagadus' intended deployment scenarios, the easiest approach was to directly connect the recording machines to the cameras and send trigger signals from the machines. Here, the challenge is the synchronization accuracy of the signals sent to the cameras. If the recording machines send signals to their own cameras, the trigger signal processes must compete for the CPU on highly loaded machines. Furthermore, for a perfectly stitched frame, we need machine synchronization at a more accurate level than NTP can deliver. An improvement would be to use one single process on only one of the machines, but the process is still competing for the resources. Our first approach was to develop a fairly generic hardware synchronization box for industry vision cameras, as shown in Fig. 20.5a. This box was powered and generated trigger signals for up to eight cameras over a dedicated cable. Furthermore, the user could set a frame rate for the box, with multipliers for every individual camera port. The box provided power to the cameras as well, because the cameras' 6-pin Hirose connector is used for both synchronization and power. This box worked fine, but due to own hardware design, the need for own power supply, a large box size, and a small clock drift varying with the large Norwegian temperature changes, we opted for a system using off-the-shelf equipment (Fig. 20.5b). It has a simpler operation in software, supports an arbitrary number of cameras, can draw power from an Ethernet cable, and provides synchronization with the recording machine to avoid drift. Figure 20.5c shows the box installed at Ullevaal stadium.



(a) Building trigger box, V1 **(b)** Trigger box, V2 **(c)** V2 in use

Fig. 20.5 Bagadus trigger boxes

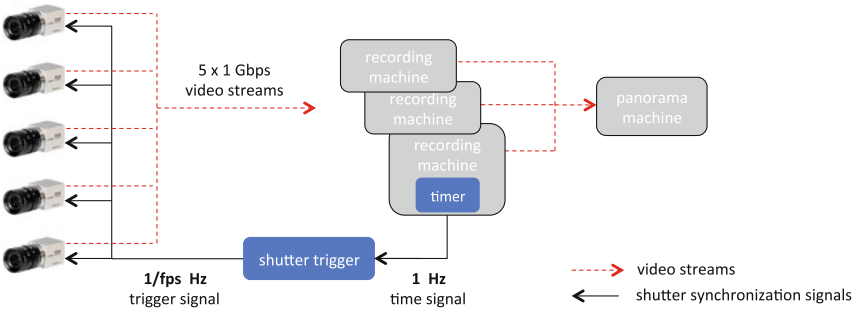


Fig. 20.6 Camera shutter synchronization in Bagadus. We have one timer running on one of the NTP synchronized recording machines syncing the trigger box once a second, and a shutter trigger box sending a signal every 1/fps second

As shown in Fig. 20.6, our trigger box[2] is divided into two parts, where we both ensure that the trigger signal is generated without interrupts (the shutter trigger) and avoid clock drift between the trigger box and the recording machines (the timer). One of the recording machines is synchronized with an NTP server and runs a process that maintains a timer. This process ensures that the clock of the synchronization box does not drift from the recording machines that is synchronized with the trigger box. The timer process sends a signal once a second to restart the timer on the synchronization box; i.e., it is synchronized once every second. In order to ensure resource availability, the process runs with the highest scheduling priority on Linux using the `SCHED_FIFO` class. Furthermore, we use an external trigger box that sends a signal to all the cameras to capture an image. Here, we use an Arduino device which uses its local clock to send a broadcast shutter trigger signal every $1/\text{fps}$ seconds. It divides the number of CPU cycles per second to the frame rate and counts the cycles between each signal.

[2]The Bagadus trigger box design and code are available here: https://bitbucket.org/mpg_code/micro-trigger-box.

Our approach ensures that all the cameras capture a new frame at exactly the same time, with an accurate frame rate. It also means that the frames will typically be either completely synchronized, or one trigger interval off, making it easier to identify when a camera falls behind and loses a frame.

20.4 Camera Exposure Synchronization

Another challenge when capturing synchronized frames from cameras having a different field-of-view is contrasting lighting conditions between these cameras. In this section, we evaluate approaches for creating a completely automatic camera array capture system with a particular focus on dynamic camera settings. There are other settings that also might be synchronized. For example, in the current scenario using several cameras to generate a panorama video, the aperture must be kept constant to avoid changing the depth of the field. This means that the only parameters that one can (or should) control are the exposure time and the gain. However, we do not have full freedom in controlling both these parameters. The electric gain introduces noise in the system, so the highest image quality is achieved when the gain is as low as possible. The exposure time has an upper limit both because it can cause motion blur during the game and also because there is a hard limit set by the frame rate. So, a priority-based estimation must be used which changes the exposure time until it reaches the threshold and then modify the gain if the exposure needs more compensation.

The cameras in our setup allow for both continuous automatic and manual exposure configuration. In an outdoor setting, lighting conditions can change quite rapidly, and we found that some form of adaptive exposure control was required. One of the main challenges in building such a system is that there is no common area of the soccer pitch visible to all the cameras (see Fig. 20.2b) that can be used for metering the light in order to set the appropriate camera parameters. The video camera is able to determine the optimal automatic exposure. However, that will be independently set for each camera. Due to different fields of view, it causes potentially large exposure differences, which becomes a visual annoyance when the images are stitched together. That can, for example, be a major problem if some cameras contain areas that are particularly bright or dark, e.g., strong sun, bright snow, or strong shadows. Examples of some of the experienced visual effects due to different exposure settings in the camera array are shown in Fig. 20.7a and b.

When doing any form of image mosaicking, color correction generally needs to be applied on all individual source images. In an earlier version of Bagadus, this operation was part of the panorama processing pipeline [11]. However, we later adapted an approach where Bagadus controls the image exposure by broadcasting one identical configuration to all cameras [54]. For this to work, however, the cameras must have identical physical configurations to ensure that the cameras produce combinable results. Our setup with identical cameras, lenses, and capture software proved to work well with an identical exposure setting on every camera. This can be seen

(a) Panorama with independent auto exposure. There is snow around the field, and the metering system has to compensate for this and make a good choice of exposure values. Here, we have used a flat panorama with static (vertical) seams. One can clearly see the differences between the cameras

(b) Panorama with independent auto exposure. Here, we have used a cylindrical panorama with dynamic seams, and the differences between the cameras are again very visible

(c) Panorama with identical camera exposure values, i.e., all cameras use the same exposure setting. The visual improvement is huge, and the seams are already very hard to identify

Fig. 20.7 Panorama with independent auto-exposure versus identical exposure

in Fig. 20.7c, which shows a panorama generated from cameras with identical exposure settings. The individual auto-exposure sometimes produces a good result, e.g., the two rightmost images in Fig. 20.7b, but often, the automatic (non-synchronized) exposure results in very different exposure times. The effects of this is even greater on sunny days, as the differences increase.

Setting the exposure manually is, however, not always sufficient. The lighting conditions often change during a game due to the weather and the movement of the sun. We therefore present an approach where the time between two temporal frames

is exploited to communicate the exposures among the cameras where we achieve a perfectly synchronized array. An analysis of the system and some experimental results are presented. In summary, a pilot camera approach running in auto-exposure mode and then distributing the used exposure values to the other cameras seems to give best visual results.

20.4.1 Existing Exposure Approaches

When an array of multiple cameras produces images that are not captured using similar exposure parameters, there will be visual differences between adjacent camera images. Often, it can be solved by color correction approaches that handle the images post-recording [55]. Tian et al. [56] highlight challenges with color distortion in panoramic imaging and introduce a new approach for color correction. Another way for color matching is also proposed by Doutre and Nasiopoulos [57]. Xu et al. [58] provide a good performance evaluation of several color correction approaches. Xiong et al. [59] proposed an elegant color correction approach which applies a color transformation that is optimized over all the images to minimize drastic changes per image. Ibrahim et al. [60] provide an interesting approach for selecting the reference for color correction.

Nevertheless, even though color correction approaches can provide good results in panorama images, they can introduce artifacts like flicker and unnatural colors when it comes to the stitched videos. Furthermore, a better solution would be to attack the problem at the source rather than correcting for the error afterward; i.e., this problem can be handled even before the recording of the videos in a constrained space like a sports stadium.

20.4.2 The Bagadus Automatic Exposure Approaches

From our initial experiments and existing work in the field, we conclude that we need some kind of dynamic automatic exposure control synchronizing the camera settings across the camera array. In this respect, we use the cameras' internal metering mechanism to estimate the exposure parameters. The region of interest that is considered for metering can be modified for each camera. We make use of this functionality in the three exposure setting approaches described here (for a quick overview, see Table 20.1). Furthermore, we also present some experimental results showing the visual differences between the approaches and the importance of a synchronized exposure when generating panorama images or video frames. Finally, we present two other related approaches to improve the image quality, i.e., high dynamic range (HDR) and seam blending (see Table 20.1).

Table 20.1 Overview of the implemented automatic exposure approaches (independent metering, pairs metering, and pilot camera), high dynamic range, and seam blending in Bagadus

Approach	Description	Region selection	Visual output
Independent metering	Use a same colored, but not the exact same, area (the green grass) as a suited surface for metering to perform individual auto-exposure. The internal mechanism decides on a specific exposure for each individual camera	Manual	Fig. 20.7a, b
Pairs metering	As shown in Fig. 20.2b, adjacent cameras have an overlapping region of pixels which is used for metering to perform pairwise synchronized auto-exposure	Manual	Fig. 20.8
Pilot camera	One pilot camera functions in fully auto-exposure mode where the pilot camera's exposure parameters are transferred to all the other cameras	Automatic	Figs. 20.3d, 20.7c, 20.10
High dynamic range	Capturing the initial video frames with alternating exposure, switching between high (bright) and low (dark) exposure times which is combined using radiance and tone mappers	Manual	Fig. 20.14
Seam blending	A simple feathering approach where a weighted average between the two overlapping areas of the images is performed	–	Fig. 20.16

20.4.2.1 Independent Metering

The most trivial approach for an automatic exposure system is independent metering. Since our target space is a soccer stadium, we can use the green surface of the soccer field, which is a well-suited surface for metering, to evaluate the exposure parameters. Initially, a manual selection of metering region is selected per camera, and the cameras are configured to make an automatic exposure. The internal mechanism decides on a specific exposure value and gain to achieve a pre-defined gray value for the average of all the pixels from the metering region. An upper limit can be imposed on the exposure time to force the camera to use a higher gain in case of low light, instead of increasing the exposure time.

We already demonstrated the effects of such an approach in the beginning of this section where Fig. 20.7a depicts a scenario where snow is on the soccer field. The metering system has to compensate for this and make a good choice of exposure values. The influence of snow can also be observed in the independent metering

approach. Figure 20.7b shows another example using a different panorama generation approach also in a contrasting lighting setting with less snow. Again, we can observe differences between the cameras. The problem is that the exposures are different in each of the cameras, and even though each image is well exposed, they are not synchronized, i.e., introducing large visual differences in the generated panorama image.

20.4.2.2 Pairs Metering

This approach can be considered as an improvement of, but still a special case of, the independent metering presented above. In this approach, we exploit the fact that the adjacent cameras have an overlapping region. Therefore, camera pairs are formed which have defined regions of interest that point to the same physical space on the field. The selection of the regions of interest is performed manually to minimize the effect from the players or other elements on the field. Then, the cameras are run independently to perform automatic exposure, but metering based on the selected patches that are overlapped. Since the camera pairs are physically close to each other, the directional reflections will have minimum effect on the exposure. However, the first camera pair and the second pair are at a distance of 4 m from each other in this experiment (tested only on an early version of the pipeline [11]).

Figure 20.8 shows the pairs metering approach in one of the possible light conditions, i.e., when the sky is partially cloudy. In this approach, a clear difference can be seen at the center of the field due to the pairwise exposure settings. However, the left and the right camera pairs (using the same region of interest for metering) of the panorama are perfectly seamless. Nevertheless, our goal is to have a perfect setting for the entire panorama, and this approach was therefore early abandoned.

20.4.2.3 Pilot Camera

The pairs metering approach shows the potential of synchronizing the camera setting, but we need an automatic solution that is camera array wide. The idea here is therefore to use a pilot camera that functions in auto-exposure mode where the pilot camera's exposure parameters are transferred to the other cameras. Here, let the m

Fig. 20.8 Panorama generated using the pairs metering approach under a partially cloudy sky

cameras be named C_j, where $j \in [1, m]$, and C_p be the pilot camera. Let e_j and g_j be the exposure time and gain of camera C_j. Then, given e_p and g_p from the pilot camera which operates in auto-exposure mode, we need to compute e_j and g_j for the rest of the cameras. Furthermore, let T_j be the transformation function from the pilot camera to camera C_j. Then,

$$(e_j, g_j) = T_j(e_p, g_p). \tag{20.1}$$

The transformation function depends on the relation of camera C_j to the camera C_p. In an ideal situation where the cameras are all identical and have exactly the same settings for aperture and focal length, T_j will be identity function. However, this is not the general case because physically different cameras do not have identical spectral response curves thus leading to difference in exposures. Other factors that can cause differences are the imperfections in adjustment of the aperture size. Generally, the cameras need a prior calibration step to estimate the corresponding transformation functions.

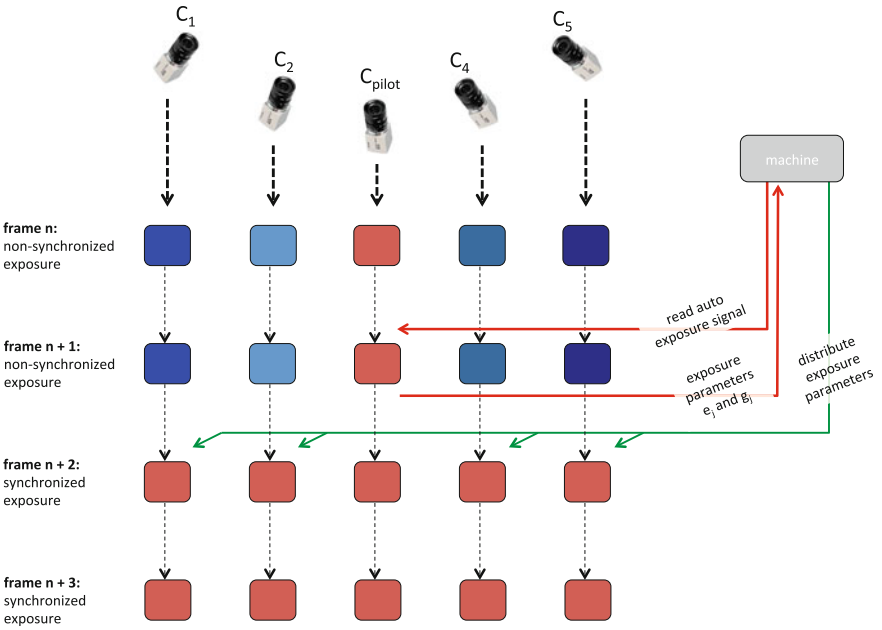


Fig. 20.9 The pilot camera approach. For frames n and $n + 1$, the colors of the frames indicate different exposure values (e_j and g_j). Camera C_{pilot} gets a signal from the controlling machine to read the auto-exposure values for frame $n + 1$. The values are sent back to the controlling machine, which again broadcasts the exposure values to the other cameras. Once received by all cameras, they all use the same exposure

The general processing flow is shown in Fig. 20.9. There are two types of threads that are running concurrently: one for controlling and communicating with the pilot camera and the others for the other cameras. All threads have a synchronization barrier at the end of every frame. Periodically, the pilot camera thread sends a trigger to the pilot camera to make an auto-exposure signal and lock the exposure until the next trigger. In Fig. 20.9, this can be seen before acquisition of frame n. After the exposure, the exposure parameters e_p and g_p are transferred back to the controlling machine. These parameters are communicated to other threads which in turn transfer these individually to the other cameras applying the appropriate transformation.

It can be observed that the frames n of the other cameras are not synchronized in exposure with the pilot camera, but we have observed empirically that the light conditions change slowly over the period of the exposure updating trigger. One more important detail is that the frame rate sets a hard upper bound on the execution time and thus on exposure time too. The formulation of transformation function cannot guarantee this because one of the transformations can demand a higher exposure time than the upper limit, especially when the cameras have lower response to light than the pilot camera. This problem can be handled in two ways. One way is to embed this property into the transformation function by placing an upper bound. The other way is to handle it in the driver before setting the camera parameters. We experienced that the driver solution is safer and more robust to further changes in the algorithm.

Figure 20.10 shows the pilot camera approach when there is an overcast sky, and using the flat panorama pipeline. Here, it can be observed from the figure that the exposure in the whole of the panorama is perfectly synchronized as there are no visual differences between the different parts in the stitched image. There is no specific color correction applied when stitching the panorama. Similar results can be observed for the later cylindrical panorama in Fig. 20.11c where the colors are the same and the stitches are hardly visible.

We therefore allow a single pilot camera to use automatic exposure. Then, at a given interval, for example, every ten seconds, we read this camera's automatic exposure values and broadcast these values to all the other cameras. This is done asynchronously by sending an exposure event message to the system server, i.e., the processing machine, which further broadcasts the message to all connected camera modules over the Transmission Control Protocol (TCP). This is not a time critical operation and can be done through a best effort approach on each individual camera stream. Figure 20.11 shows an extreme case, where each camera asynchronously receives the first exposure update. Once the recording is up and running, there are typically only minor changes on each update.

Fig. 20.10 Panorama generated using the pilot camera approach under an overcast sky

(a) Initial metering (frame k)

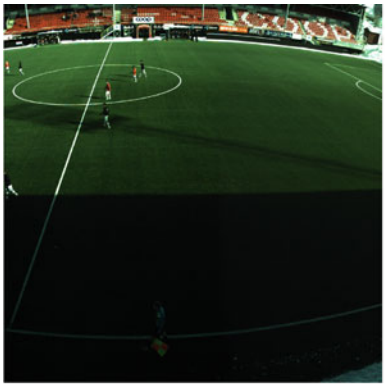
(b) The exposure update did not reach all cameras (frame $k+1$)

(c) All cameras' exposures are updated (frame $k+2$)

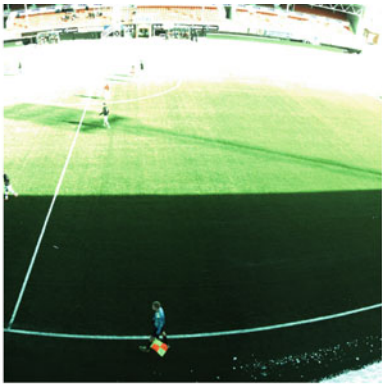
Fig. 20.11 Panoramas during an exposure synchronization. Initially, the frames are dark, and we here display how the exposure for different cameras changes during the metering process. Three frames are captured 20 ms apart at 50 fps. Usually, updated exposure settings are received by all machines between two frames

20.4.2.4 High Dynamic Range

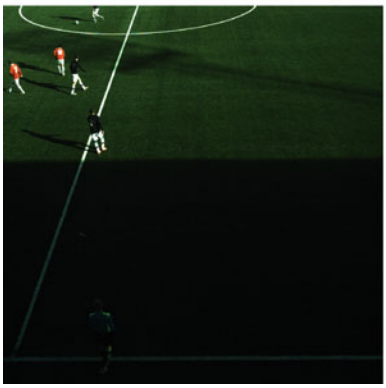
In Bagadus, as mentioned above, we also have support for HDR in order to deal with the challenging lighting conditions of a large outside environment in the presence of strong sun. This mechanism is achieved by capturing the initial video frames with alternating exposure, switching between high (bright) and low (dark) exposure times. However, in the case of exposure synchronization, the inclusion of the HDR module complicates the operation as the cameras must alternately switch between high and low synchronized exposure values. In order to generate a usable panorama, the two exposure times to use must be selected with care. Figure 20.12 shows an example output, with the low and high exposure images stitched together (displaying only parts of the entire panorama). Note how the two images complement each other by



(a) Low exposure set



(b) High exposure set



(c) Low exposure set (cropped)

(d) High exposure set (cropped)

Fig. 20.12 Sample exposure sets, used by the HDR module, showing the low and high exposure frames. Note that these have been stitched for visual purposes, though in the actual module, they are computed on the raw input image set, before stitching

showing details in different regions. This is vital to the success of the HDR process. By selecting exposure values that are too close together, we see less gain from the HDR module, and we may see a lot of noise in dark areas, as these are boosted.

This is especially challenging as we still wish to utilize dynamic exposures, setting automatic exposure on a single camera and allowing the other streams to duplicate the configuration. Setting automatic exposure in our system takes two or three frames when operating at 50 fps before the values are successfully updated. In this period, we decide to drop the frames causing a small hiccup of 50 ms. However, we experienced that the automatic exposure updates can be fairly infrequent; i.e., every 10 or 20 s is sufficient.

The actual low and high exposure times can be determined in several ways. One approach is to use two regions of interest: one positioned in the sunlight and one positioned in the shadow. This works well, although the resulting exposure values tend to be too far apart for the HDR module to handle it well. However, we learned that this was difficult to do in real-time updates, because changing this region of interest in the camera is extremely slow. Another approach is to perform auto-exposure with a single region of interest, preferably solely in the sun as this is more sensitive to changing light conditions, and then set the other exposure time as a static offset or a static percentage increase/decrease. This can work well in many situations, but we do not necessarily know the difference between the two regions, as this scales with brightness of the scene. We could also set a few static values and use whichever is closest based on the result of the auto-exposure.

We chose to use a mixture of these two approaches, by first defining the two regions as shown in Fig. 20.13. Then, we spend a few seconds at the beginning of the recording to estimate the optimal exposure times in each of these two regions. This

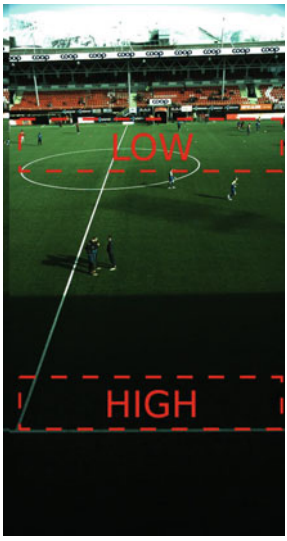


Fig. 20.13 Regions of interest used for determining auto-exposure for HDR

gives us the relative difference between the two regions. This difference is typically way too large, and we adjust the high exposure time E_{high} by:

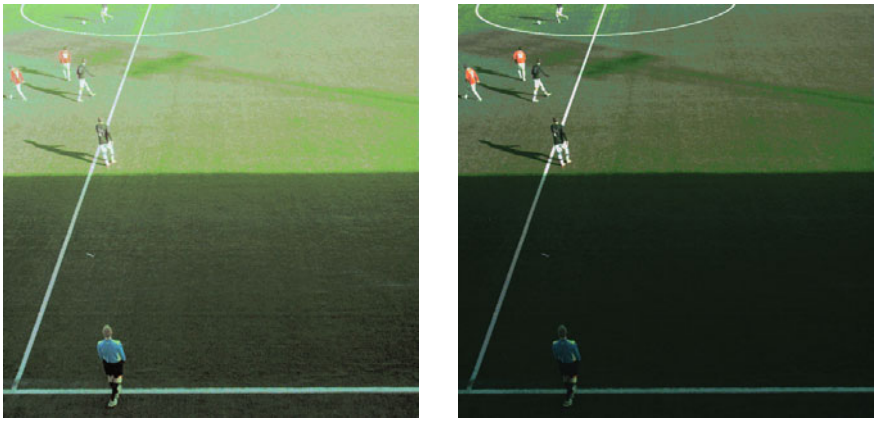
$$E_{high} = E_{low} + \frac{E_{high} - E_{low}}{2} \tag{20.2}$$

Then, we use the low exposure region of interest, i.e., sunny region, to determine the automatic exposure of E_{low} throughout the recording, updated as aforementioned. The high exposure time is determined by:

$$E_{high} = E_{high} + \frac{(E_{updated} - E_{low}) \times E_{low}}{E_{high}} \tag{20.3}$$

where $E_{updated}$ is the new low exposure time. This way, we maintain a correlation between the two exposure times. This assumes that the initial correlation was accurate, however, which may not be the case if the weather conditions move from cloudy to sunny during the recording.

Figure 20.14 shows example outputs of the HDR module using the images in Fig. 20.12c and d as input. There are multiple ways to combine the low and high exposure frames (see more examples in [13]), but here we have used the Debevec radiance mapper [61] combined with the Larson [62] and Reinhard [63] tone mappers. We believe that using the Debevec radiance mapper paired with the Larson tone mapper is a good solution taking the visual quality and the execution overhead into account. Although Reinhard's tone mapper also shows promising results, we could not make it pass the real-time requirements, and it also consumed a lot of memory on the GPU, thus affecting other components of the pipeline.



(a) Debevec - Larson **(b)** Debevec - Reinhard

Fig. 20.14 Visual quality after HDR using different radiance and tone mapping algorithms using the input images in Fig. 20.12c and d

20.4.2.5 Seam Blending

Even though we believe the pilot camera exposure synchronization approach is sufficient, we can occasionally, primarily in very bright weather, see some vignetting effects from the camera, i.e., a reduced brightness near the edges of the image compared to the center. An example with an artificially enhanced effect for the illustration is shown in Fig. 20.15a.

A frequently used solution is seam blending. Traditional solutions include Gaussian filtering and pyramid blending [64]. These are, however, too computational expensive for our real-time pipeline. Therefore, we use the simple feathering approach that performs a weighted average between the two overlapping images:

$$OutPixel(x, y) = \frac{i}{i_{max}} \times img_a(x, y) + \left(1 - \frac{i}{i_{max}}\right) \times img_b(x, y)$$

where i_{max} represents the size of the blending region and i the index within the region. Furthermore, we also perform a selective blend, where we ignore the pixels that are widely different, we filter out pixels that have a high edge cost in the stitch, and if not enough samples are found, we use the non-blended seam (more details are found in [65]). Finally, we experienced that it sufficient to perform the blending only in the luminosity component, as we primarily want to focus on removing very slight differences in exposure in the two images. Thus, the blending is performed by finding the average difference in luminosity between all the homogeneous pixels within a blending region of 40×20 pixels, around the original seam. The final result is visible in Fig. 20.16 where we can observe a panorama frame from a recording process,

(a) Stitch without blending **(b)** Stitch with blending

Fig. 20.15 Example of two images with large color differences, showing the effect of performing a post-stitch blending. Note that in the system large differences are very rare, and the images presented have been modified to increase the lighting difference for the illustration

Fig. 20.16 Panorama with pilot camera exposure synchronization and seam blending

implemented and running on a GPU using the compute unified device architecture (CUDA), with pilot camera exposure synchronization and stitch blending.

20.5 Conclusions

In our Bagadus system, the users expect a nice visual experience using the generated videos for sport analysis. In order to generate the high-resolution panorama covering the entire field-of-view of a soccer stadium, we have presented a camera array recording system which requires frame-by-frame synchronization, both with respect to shutter synchrony and camera exposure. The shutter synchrony is required to avoid ghosting effects, and this is solved using a hardware trigger where our trigger box sends shutter trigger signals at 1/fps using its local clock, and the trigger box itself is synchronized with the controlling machine to avoid clock drift. The camera exposure synchronization is required to minimize the visibility of the seams when stitching the individual frames into a full field-of-view panorama video. Our solution to this problem is to use a pilot camera that operates in fully auto-exposure mode, and its captured exposure parameters are transferred to and used by all the other cameras. We presented different approaches, and as shown in Figs. 20.3c and 20.16 for Alfheim stadium and Fig. 20.3d for Ullevaal stadium, the visual output of the proposed solutions is good.

Acknowledgements This work has been performed in the context of the iAD Centre for Research-based Innovation (project number 174867), and it is also supported in part by the EONS project (project number 231687)—both funded by the Research Council of Norway. Furthermore, there are numerous students and researchers that have worked on Bagadus or post-Bagadus solutions. For the synchronization of cameras, the authors want to acknowledge in alphabetical order: Alexander Eichhorn, Martin Stensgård, and Simen Sægrov.

References

1. Halvorsen, P., Sægrov, S., Mortensen, A., Kristensen, D.K., Eichhorn, A., Stenhaug, M., Dahl, S., Stensland, H.K., Gaddam, V.R., Griwodz, C., Johansen, D.: Bagadus: An integrated system for arena sports analytics a soccer case study. In: Proceedings of ACM MMSys, pp. 48–59 (2013)
2. Sægrov, S., Eichhorn, A., Emerslund, J., Stensland, H.K., Griwodz, C., Johansen, D., Halvorsen, P.: Bagadus: an integrated system for soccer analysis (demo). In: Proceedings of ICDSC, pp. 1–2 (2012)
3. Stensland, H.K., Gaddam, V.R., Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H.K., Mortensen, A., Langseth, R., Ljødal, S., Landsverk, Ø., Griwodz, C., Halvorsen, P., Stenhaug, M., Johansen, D.: Bagadus: an integrated real-time system for soccer analytics. ACM TOMCCAP **10**(1s) (2014)
4. Mortensen, A., Gaddam, V.R., Stensland, H.K., Griwodz, C., Johansen, D., Halvorsen, P.: Automatic event extraction and video summaries from soccer games. In: Proceedings of ACM MMSys, pp. 176–179 (2014)
5. ChyronHego: ZXY Sport Tracking. http://www.zxy.no/
6. Mills, D., Martin, J., Burbank, J., Kasch, W.: Network time protocol version 4: protocol and algorithms specification. RFC 5905 (Proposed Standard) (2010)
7. Johansen, D., Stenhaug, M., Hansen, R.B.A., Christensen, A., Høgmo, P.M.: Muithu: smaller footprint, potentially larger imprint. In: Proceedings of IEEE ICDIM, pp. 205–214 (2012)
8. Basler aca2000-50gc. http://www.baslerweb.com/products/ace.html?model=173
9. Azure-0814m5m. http://www.azurephotonicsus.com/products/azure-0814M5M.html
10. Stensland, H.K., Gaddam, V.R., Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H.K., Griwodz, C., Halvorsen, P., Johansen, D.: Processing panorama video in real-time. Int. J. Semant. Comput. **8**(2) (2014)
11. Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H.K., Stensland, H.K., Gaddam, V.R., Johansen, D., Griwodz, C., Halvorsen, P.: Efficient implementation and processing of a real-time panorama video pipeline. In: Proceedings of IEEE ISM (2013)
12. Langseth, R., Gaddam, V.R., Stensland, H.K., Griwodz, C., Halvorsen, P., Johansen, D.: An experimental evaluation of debayering algorithms on gpus for recording panoramic video in real-time. Int. J. Multimedia Data Eng. Manag. **6**(6) (2015)
13. Kellerer, L., Gaddam, V.R., Langseth, R., Stensland, H.K., Griwodz, C., Johansen, D., Halvorsen, P.: Real-time HDR panorama video. In: Proceedings of ACM MM, pp. 1205–1208 (2014)
14. Stensland, H.K., Wilhelmsen, M.A., Gaddam, V.R., Mortensen, A., Langseth, R., Griwodz, C., Halvorsen, P.: Using a commodity hardware video encoder for interactive applications. Int. J. Multimedia Data Eng. Manag. **6**(3), 17–31 (2015)
15. Wilhelmsen, M.A., Stensland, H.K., Gaddam, V.R., Mortensen, A., Langseth, R., Griwodz, C., Johansen, D., Halvorsen, P.: Using a commodity hardware video encoder for interactive video streaming. In: Proceedings of IEEE ISM (2014)
16. VideoLAN: x264. http://www.videolan.org/developers/x264.html
17. Gaddam, V.R., Langseth, R., Ljødal, S., Gurdjos, P., Charvillat, V., Griwodz, C., Halvorsen, P.: Interactive zoom and panning from live panoramic video. In: Proceedings of ACM NOSSDAV, pp. 19–24 (2014)
18. Gaddam, V.R., Bao Ngo, H., Langseth, R., Griwodz, C., Johansen, D., Halvorsen, P.: Tiling of panorama video for interactive virtual cameras: overheads and potential bandwidth requirement. In: Proceedings of IEEE PV, pp. 204–209 (2015)
19. Gaddam, V.R., Riegler, M., Eg, R., Griwodz, C., Halvorsen, P.: Tiling in interactive panoramic video: approaches and evaluation. IEEE Trans. Multimedia **18**(9), 1819–1831 (2016)
20. Niamut, O.A., Thomas, E., D'Acunto, L., Concolato, C., Denoual, F., Lim, S.Y.: Mpeg dash srd: Spatial relationship description. In: Proceedings of MMSys (2016)
21. Sanchez, Y., Skupin, R., Schierl, T.: Compressed domain video processing for tile based panoramic streaming using hevc. In: Proceedings of IEEE ICIP, pp. 2244–2248 (2015)

22. NTP.org: NTP faq—How does it work? http://www.ntp.org/ntpfaq/NTP-s-algo.htm
23. Wikipedia: 100 metres. https://en.wikipedia.org/wiki/100_metres
24. Quora: How fast can a soccer ball be kicked? https://www.quora.com/How-fast-can-a-soccer-ball-be-kicked
25. Hasler, N., Rosenhahn, B., Thormahlen, T., Wand, M., Gall, J., Seidel, H.P.: Markerless motion capture with unsynchronized moving cameras. In: Proceedings of IEEE CVPR, pp. 224–231 (2009)
26. Pourcelot, P., Audigié, F., Degueurce, C., Geiger, D., Denoix, J.M.: A method to synchronise cameras using the direct linear transformation technique **33**(12), 1751–1754 (2000)
27. Shrestha, P., Barbieri, M., Weda, H., Sekulovski, D.: Synchronization of multiple camera videos using audio-visual features. IEEE Trans. Multimedia **12**(1), 79–92 (2010)
28. Shrestha, P., Weda, H., Barbieri, M., Sekulovski, D.: Synchronization of multiple video recordings based on still camera flashes. In: Proceedings ACM MM. New York, USA (2006)
29. Ruttle, J., Manzke, M., Prazak, M., Dahyot, R.: Synchronized real-time multi-sensor motion capture system. In: Proceedings of ACM SIGGRAPH ASIA (2009)
30. Bradley, D., Atcheson, B., Ihrke, I., Heidrich, W.: Synchronization and rolling shutter compensation for consumer video camera arrays. In: Proceedings of IEEE CVPR, pp. 1–8 (2009)
31. Duckworth, T., Roberts, D.J.: Camera image synchronisation in multiple camera real-time 3D reconstruction of moving humans. In: Proceedings of DS-RT, pp. 138–144 (2011)
32. Moore, C., Duckworth, T., Aspin, R., Roberts, D.: Synchronization of images from multiple cameras to reconstruct a moving human. In: Proceedings of IEEE/ACM DR-RT, pp. 53–60 (2010)
33. Chang, R., Ieng, S., Benosman, R.: Shapes to synchronize camera networks. In: Proceedings of IEEE ICPR, pp. 1–4 (2008)
34. Sinha, S., Pollefeys, M.: Synchronization and calibration of camera networks from silhouettes. In: Proceedings of ICPR, vol. 1, pp. 116–119 (2004)
35. Sinha, S.N., Pollefeys, M.: Camera network calibration and synchronization from silhouettes in archived video. Int. J. Comput. Vis. **87**(3), 266–283 (2010)
36. Topçu, O., Ercan, A.Ö., Alatan, A.A.: Recovery of temporal synchronization error through online 3D tracking with two cameras. In: Proceedings of ICDSC, pp. 1–6 (2014)
37. Haufmann, T.A., Brodtkorb, A.R., Berge, A., Kim, A.: Real-time online camera synchronization for volume carving on GPU. In: Proceedings of AVSS, pp. 288–293 (2013)
38. Nischt, M., Swaminathan, R.: Self-calibration of asynchronized camera networks. In: Proceedings of ICCV Workshops, pp. 2164–2171 (2009)
39. Shankar, S., Lasenby, J., Kokaram, A.: Synchronization of user-generated videos through trajectory correspondence and a refinement procedure. In: Proceedings of CVMP, pp. 1–10 (2013)
40. Shankar, S., Lasenby, J., Kokaram, A.: Warping trajectories for video synchronization. In: Proceedings of ARTEMIS, pp. 41–48 (2013)
41. Tao, J., Risse, B., Jiang, X., Klette, R.: 3D trajectory estimation of simulated fruit flies. In: Proceedings of IVCNZ (2012)
42. Velipasalar, S., Wolf, W.H.: Frame-level temporal calibration of video sequences from unsynchronized cameras. Mach. Vis. Appl. **19**(5–6), 395–409 (2008)
43. Whitehead, A., Laganiere, R., Bose, P.: Temporal synchronization of video sequences in theory and in practice. In: Proceedings of IEEE WACV/MOTION, pp. 132–137 (2005)
44. Kovacs, J.: AN005 Application Note—An Overview of Genlock. http://www.mivs.com/old-site/documents/appnotes/an005.html
45. Smith, S.L.: Application of high-speed videography in sports analysis. In: Proceedings of SPIE 1757 (1993)
46. Collins, R.T., Amidi, O., Kanade, T.: An active camera system for acquiring multi-view video. In: Proceedings of ICIP, pp. 520–527 (2002)
47. Lin, M.Y.: Shutter synchronization circuit for stereoscopic systems (1998). https://www.google.com/patents/US5808588

48. Gross, M., Lang, S., Strehlke, K., Moere, A.V., Staadt, O., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Van Gool, L.: Blue-c: a spatially immersive display and 3D video portal for telepresence. In: Proceedings of ACM SIGGRAPH (2003)
49. Wilburn, B., Joshi, N., Vaish, V., Levoy, M., Horowitz, M.: High-speed videography using a dense camera array. Proc. IEEE CVPR **2**, 294–301 (2004)
50. Meyer, F., Bahr, A., Lochmatter, T., Borrani, F.: Wireless GPS-based phase-locked synchronization system for outdoor environment. J. Biomech. **45**(1), 188–190 (2012)
51. Litos, G., Zabulis, X., Triantafyllidis, G.: Synchronous image acquisition based on network synchronization. In: Proceedings of CVPR Workshops (3DCINE), pp. 167–167
52. Sousa, R.M., Wäny, M., Santos, P., Dias, M.: Multi-camera synchronization core implemented on USB3 based FPGA platform. In: Proceedings of SPIE 9403 (2015)
53. Nguyen, H., Nguyen, D., Wang, Z., Kieu, H., Le, M.: Real-time, high-accuracy 3D imaging and shape measurement. Appl. Opt. **54**(1), A9 (2015)
54. Gaddam, V.R., Griwodz, C., Halvorsen, P.: Automatic exposure for panoramic systems in uncontrolled lighting conditions: a football stadium case study. In: Proceedings of SPIE 9012—The Engineering Reality of Virtual Reality (2014)
55. Hasler, D., Ssstrunk, S.: Mapping colour in image stitching applications. J. Visual Commun. Im. Represent. **15**(1), 65–90 (2004)
56. Tian, G.Y., Gledhill, D., Taylor, D., Clarke, D.: Colour correction for panoramic imaging. In: Proceedings of IV (2002)
57. Doutre, C., Nasiopoulos, P.: Fast vignetting correction and color matching for panoramic image stitching. In: Proceedings of ICIP, pp. 709–712 (2009)
58. Xu, W., Mulligan, J.: Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In: Proceedings of IEEE CVPR, pp. 263–270 (2010)
59. Xiong, Y., Pulli, K.: Color correction for mobile panorama imaging. In: Proceedings of ICIMCS, pp. 219–226 (2009)
60. Ibrahim, M., Hafiz, R., Khan, M., Cho, Y., Cha, J.: Automatic reference selection for parametric color correction schemes for panoramic video stitching. Adv. Visual Comput. Lect. Notes Comput. Sci. **7431**, 492–501 (2012)
61. Debevec, P.E., Malik, J.: Recovering high dynamic range radiance maps from photographs. In: Proceedings of SIGGRAPH, pp. 369–378 (1997)
62. Larson, G.W., Rushmeier, H., Piatko, C.: A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Trans. Visual. Comput. Graph. **3**(4), 291–306 (1997)
63. Reinhard, E., Stark, M., Shirley, P., Ferwerda, J.: Photographic tone reproduction for digital images. ACM Trans. Graph. **21**(3), 267–276 (2002)
64. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall, Inc. (2006)
65. Langseth, R.: Implementation of a distributed real-time video panorama pipeline for creating high quality virtual views. University of Oslo (2014)

Chapter 21
Merge and Forward: A Self-Organized Inter-Destination Media Synchronization Scheme for Adaptive Media Streaming over HTTP

Benjamin Rainer, Stefan Petscharnig and Christian Timmerer

Abstract In this chapter, we present *Merge and Forward*, an IDMS scheme for adaptive HTTP streaming as a distributed control scheme and adopting the MPEG-DASH standard as representation format. We introduce so-called IDMS sessions and describe how an unstructured peer-to-peer overlay can be created using the session information using MPEG-DASH. We objectively assess the performance of *Merge and Forward* with respect to convergence time (time needed until all clients hold the same reference time stamp) and scalability. After the negotiation on a reference time stamp, the clients have to synchronize their multimedia playback to the agreed reference time stamp. In order to achieve this, we propose a new adaptive media playout approach minimizing the impact of playback synchronization on the QoE. The proposed adaptive media playout is assessed subjectively using crowd sourcing. We further propose a crowd sourcing methodology for conducting subjective quality assessments in the field of IDMS by utilizing *GWAP*. We validate the applicability of our methodology by investigating the lower asynchronism threshold for IDMS in scenarios like online quiz games.

Keywords IDMS · Distributed algorithm · Distributed systems · Crowd sourcing · Multimedia streaming · MPEG-DASH

21.1 Introduction

Over the past decade, leisure time communication has evolved extensively through appearance of social platforms, such as Facebook, Twitter, and Google+. These

B. Rainer (✉)
Austrian Institute of Technology (AIT), Seibersdorf, Austria
e-mail: benjamin.rainer@ait.ac.at

S. Petscharnig · C. Timmerer
Institute of Information Technology, Klagenfurt, Austria
e-mail: stefan.petscharnig@itec.aau.at

C. Timmerer
e-mail: christan.timmerer@itec.aau.at

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_21

cutting-edge forms of social interaction help us to create, distribute, and view multimedia content, but also they impose new requirements on the underlying technologies. The traditional TV scenario as we know it, watching TV with friends and family, is becoming increasingly location independent with people wanting to experience multimedia together even if they are geographically distributed. This new form of togetherness utilizes real-time communication channels such as text, voice, or even video telephony in order to share the experience.

The presence of a real-time communication channel requires a synchronized playback of the multimedia content among the participating users. Asynchronous playback may lead to an unpleasant viewing experience and may diminish the feeling of togetherness of the users [1]. For example, consider a group of friends watching a soccer game together using multiple devices at multiple locations. Some users may experience that the media playback is a few seconds ahead of the others. This behavior may lead to a low system acceptance. Thus, playback synchronization must be a key feature of such systems. In general, the synchronization of the playback among geographically distributed users is termed IDMS [2]. IDMS includes both the synchronization on network level which is about determining a reference time stamp and the synchronization on playback level which is actually adjusting the current playback to the reference. The technical challenges of IDMS can be summarized based on the type of streaming technology employed (e.g., pull- or push-based), the selection of an appropriate synchronization point, and a smooth and imperceptible synchronization of the multimedia playback at the individual clients. An IDMS system trying to tackle the aforementioned challenges relies on the following mechanisms:

1. **Session management**: responsible for mapping clients to sessions.
2. **Information signaling**: allows the exchange of timing information and, if necessary, control information between clients.
3. **Reference playback time stamp negotiation**: deals with the selection of a playback time stamp within a session. Clients have to synchronize their playback according to that reference time stamp.
4. **Playback synchronization mechanism**: overcomes the identified asynchronism by altering the multimedia playback of the concerned clients.

Thus, IDMS has to provide two synchronization mechanisms one on the network level (agreeing on a reference time stamp) and one on the actual playback level (synchronizing the playback of each client to the reference playback time stamp). In this chapter, we will first discuss the synchronization on the network level and introduce a novel distributed algorithm. Second, we will discuss how the playback itself can be synchronized while considering the user's QoE. This means that we try to *imperceptibly* synchronize the playback of each client to the reference.

Regarding the synchronization on the network level, we distinguish between three major schemes, MS, SMS, and DCS. The common assumption of most IDMS solutions is that clocks are already synchronized using, i.e., the network time protocol (NTP) [3] or the precision time protocol (PTP) [4].

Master/Slave scheme (MS) (cf. Fig. 21.1a): This scheme uses a dedicated master for signaling timing information. Figure 21.1a depicts a MS with a master (red node),

the media source (green node), and two clients. The master may be elected among the participating clients or determined by the media source. If the selected master leaves the session, a new master has to be selected or elected among the clients. Furthermore, the clients have to *trust* the master whether control and timing information received by it is correct and valid. The advantage of this scheme is that the instance which is responsible for signaling timing information is elected among the participating clients. An approach that follows the Master/Slave scheme was first proposed in [5]. The approach presented in [5] builds on top of multicast, and the synchronization protocol employed also maintains intra- and inter-stream synchronization. The Master/Slave scheme suffers from scalability issues because timing information is exchanged between each peer and the selected master which may lead to bandwidth shortages using unicast and a certain number of clients. Therefore, most solutions that implement on a MS will require multicast.

Synchronization Master Scheme (SMS) (cf. Fig. 21.1b): It is a centralized approach where the synchronization is controlled by a synchronization master which may be the media source or a dedicated synchronization node (not a client that consumes multimedia content). Figure 21.1b depicts the SMS with three clients and a media source (green node) that has the additional task of being the synchronization master. The synchronization master collects timing information and sends timing and control information to which the clients have to adhere. This approach suffers from scalability issues because a central instance can only handle a certain number of clients. Furthermore, if more than one synchronization master is used, there has to be dedicated communication between them in order to gain consensus about the synchronization information send to the clients. The advantage compared to the MS is that the clients can trust the synchronization master because it is in control of the content provider. A SMS is presented in [6], which adopts the local lag and time warp algorithm compensating for media playback inconsistencies [7]. Further SMS solutions are presented in [8, 9]. Both approaches extend the Receiver and Sender

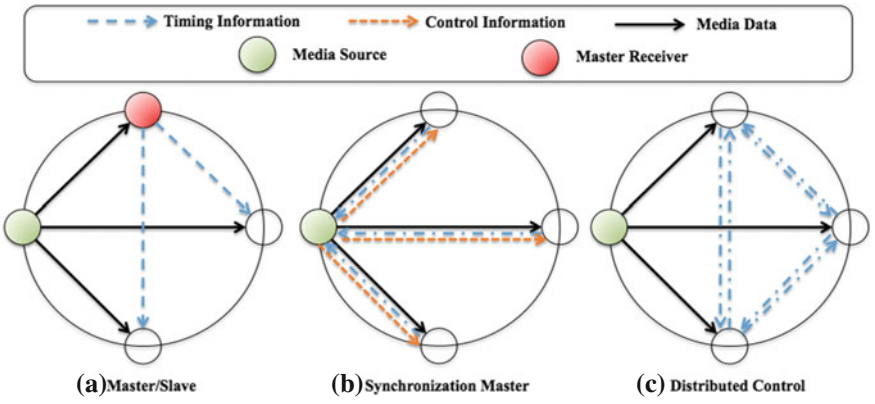


Fig. 21.1 IDMS schemes: Master/Slave scheme, Synchronization Master scheme, and Distributed Control scheme

Reports (RR and SR) defined within the RTP/RTCP in order to carry the necessary timing and control information [10]. Furthermore, the extensions to the RTP/RTCP protocol are standardized under RFC 7272 [11]. RFC 7272 supports multicast and unicast.

Distributed Control Scheme (DCS) (cf. Fig. 21.1c): It uses distributed protocols to determine a reference playback time stamp to which the clients may synchronize their actual media playback. Therefore, timing information is exchanged in a peer-to-peer manner among the clients. This scheme has the highest robustness in terms of overall failure probability. Figure 21.1c depicts a DCS with a media source (green node) and three clients which exchange only timing information in order to agree on a reference playback time stamp. The content provider has only to provide the multimedia content. Nevertheless, the clients have to trust each other in terms of faulty behavior. Furthermore, NAT may cause problems, especially if the clients are behind symmetric NATs. Hesselman et al. [12] presents the *iNEM4U* approach where a DCS is used to achieve IDMS among heterogeneous network infrastructures. Furthermore, it introduces *iSession* for the session management, which provides an XML description of each session including the users and/or clients, content source, and other service-specific data. A very recent DCS for achieving IDMS which uses an extended version of RTCP messages is presented in [13]. The proposed DCS is designed on top of RTP/RTCP, and clients are assigned to a specific cluster. Within a cluster, the clients regularly exchange RTCP RR packets including playback timestamps in order to achieve intra-cluster synchronization. This solution uses multicast for exchanging the RTCP RR packets (following the principles introduced by RFC 7272) between the clients. Another DCS is presented in [14] which uses multicast and provides intra-stream synchronization.

Most solutions which follow one of the listed schemes are built on top of the employed streaming protocol (e.g., RTP/RTCP). Therefore, the actual synchronization on the network level cannot be decoupled from the employed streaming protocol. In contrast to RTP/RTCP which is a purely server-driven (or push-based) streaming protocol, adaptive HTTP streaming such as MPEG-DASH [15] adopts a pure client-centric (or pull-based) approach. Hence, it migrates the needed streaming mechanisms (such as adaptation decisions) to the client. This facilitates simple HTTP servers that provide content in an MPEG-DASH compliant format (e.g., segmented ISO Base Media File Format or MPEG-2 Transport Stream). Furthermore, MPEG-DASH provides a MPD which describes the various spatial, temporal, and quality dimensions of the content as well as different encodings. The MPD also contains information on the location of the content with support for multiple content servers [15]. During the course of this chapter, we introduce *Merge and Forward* a self-organized pull-based IDMS approach adopting MPEG-DASH for session management and constructing a peer-to-peer overlay among the participating clients. Please note that although we adopt MPEG-DASH as the main representation format, the approach presented in this chapter can also be adopted for other formats sharing the same design principles (e.g., Apple HLS or Microsoft Smooth Streaming). In the next section, we will discuss a DCS that can be decoupled from the employed streaming protocol.

Besides the communication between the clients using a centralized or decentralized scheme, another very important decision in IDMS systems, specifically in distributed systems, is the calculation of the reference playback time stamp to which the clients synchronize their media playback. In [16], three different reference selection policies are discussed:

- **Synchronization to the slowest client**, i.e., the client that is displaying the lowest frame number among the group of clients;
- **Synchronization to the fastest client**, i.e., the client that is displaying the highest frame number; and
- **Synchronization to the average**, i.e., to the average frame number among a group of clients.

Each of these policies has its advantages and disadvantages, e.g., synchronizing to the slowest client will cause all clients that are far ahead the reference to pause the playback. The average ($\overline{X}$) is the fairest selection among these three policies from a mathematical point of view because if we aim on minimizing the squared error ε between the reference playback time stamp and all current playback time stamps of the clients expressed by $\varepsilon = \frac{1}{2}\sum_{i=1}^{N}(x_i - \overline{X})^2$, N denotes the number of clients participating in an IDMS session and x_i denotes the current playback time stamp of client i, where $\forall\, x_i,\ 1 \leq i \leq N : x_i \in \mathbb{R}_+$. With $\frac{\partial \varepsilon}{\partial \overline{X}} = \sum_{i=1}^{N}(x_i - \overline{X}) \stackrel{!}{=} 0$, it trivially follows that $\overline{X} = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i$. Therefore, the average as reference point implies the lowest error (with respect to the least square error).

Picking up again the actual playback synchronization, there is ongoing research how the playback has to be altered in order to reach the reference playback time stamp provided by the network-level synchronization. Currently, the common denominator of the discussed IDMS solutions is that compensating the identified asynchronism is done by skipping or pausing the multimedia playback. However, stalling the media playback has a negative impact on the viewing experience/quality of experience of the user shown in [17]. In order to provide a possibility to have a smooth transition instead of abrupt pausing or skipping during the media playback, AMP has been introduced [18]. Here, we increase or decrease the media playback rate of, e.g., video and audio simultaneously until the client's playback reaches the reference playback time stamp. Historically, AMP was introduced to compensate for buffer underflows or overflows by decreasing or increasing the media playback rate to allow the stabilization of the playback buffer in multimedia streaming [19]. We will investigate how AMP can be used to achieve synchronization of the media playback of a group of clients with respect to the QoE by utilizing media content features.

The remainder of the chapter is as follows. Section 21.2 discusses a DCS using MPEG-DASH as streaming technology [20]. This includes the inclusion of session information in MPEG-DASH (Sect. 21.2.1), building and maintaining the peer-to-peer overlay and the actual synchronization (Sect. 21.2.2), and the synchronization on playback level (Sect. 21.2.3). Section 21.3 picks up the principles of GWAP subjectively evaluating IDMS using crowd sourcing.

21.2 Merge and Forward—A Self-Organized IDMS Scheme

Figure 21.2 illustrates our IDMS approach for pull-based streaming and, in particular, for MPEG-DASH. Rather than modifying the server side for IDMS, we introduce **session management** by defining ISO. ISOs are referenced from within the MPD and are stored at the server providing the MPD (i.e., *MPD Server*). We assume that there is a dedicated *MPD Server* that handles MPD requests from clients. The *Application Layer Peer-to-Peer Overlay* is implicitly built by our **distributed synchronization protocol** that utilizes the information contained in an ISO. In order to create the *Application Peer-to-Peer Overlay*, STUN [21] and its relay extensions specified in [22] have to be utilized. These methods allow for a peer-to-peer connection even if clients are behind a NAT. Therefore, our IDMS architecture foresees a separate *STUN Instance* which provides the clients the possibility of detecting the nature of their NAT. The **distributed synchronization protocol** consists of a two-stage protocol. First, it provides a **coarse synchronization** that is introduced on behalf of how multimedia content is segmented using MPEG-DASH. Second, it provides the **fine-grained synchronization** that finally yields a reference playback time stamp for synchronizing the multimedia playback of the clients.

21.2.1 Session Management for MPEG-DASH

Pull-based streaming such as MPEG-DASH represents multimedia content as equally sized, self-contained time units (e.g., 2, 4, 10 s) which are referred to as segments. These segments may be stored as separate files or are indexed by byte ranges in a contiguous file. Additionally, the multimedia content may be provided in different representations—described by the MPD —offering various scalability possibilities (e.g., spatial, temporal, quality) of the multimedia content. The adaptation between representations takes place at segment boundaries. We refer to an IDMS session as a collection of users/clients (family, friends) who want to experience the same mul-

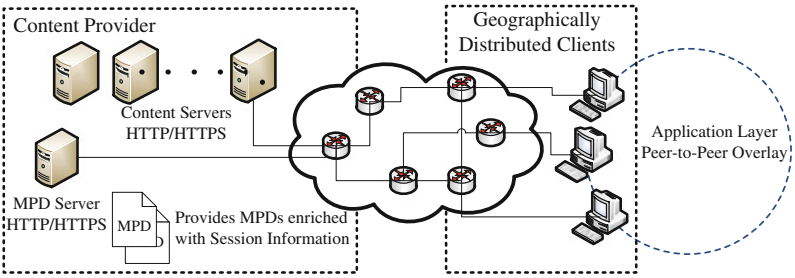


Fig. 21.2 Architecture of IDMS for adaptive media streaming over HTTP adopting MPEG-DASH

timedia together despite being geographically distributed. In order to keep track of how many clients are currently involved in an IDMS session, we utilize the MPD of MPEG-DASH to store session relevant information. Therefore, we extend the MPD of MPEG-DASH [15] with so-called ISOs which are matched against a session key provided by users. Nevertheless, our solution remains compliant to the MPEG-DASH standard because non-IDMS clients will just ignore the additional session description when parsing the MPD.

Definition 1 *(IDMS Session Object)* An ISO is a time-bounded entity to which a set of n clients is assigned to. Let $\mathscr{C}$ be the set of clients, then an ISO $\mathscr{I}$ is $\mathscr{I} \in \mathscr{C}^n \times \mathbb{R}_+ \times \mathbb{N}$. Each ISO shall have an unique identifier $\in \mathbb{N}$ for a certain multimedia content and a time-to-live $\in \mathbb{R}_+$. Furthermore, it shall allow for a unique numbering of the clients that shall be addressable uniquely. Therefore, let $\mathscr{I}_k$ be the k-th component of $\mathscr{I}$, then $0 \leq i,j \leq n$ it holds that $\mathscr{I}_i \neq \mathscr{I}_j$, $i \neq j$.

Invalid ISOs may be deleted without any caution. According to Definition 1, we assume that an ISO is identified by an unique session key which is provided by the application (using a centralized instance or a distributed protocol to agree upon a unique key) or the content provider. The session key identifies an ISO uniquely. The session key is signaled by adding it to the HTTP GET message that requests a MPD from the *MPD Server* along with the public IP (IPv4 or IPv6) address, port number, and type of the client's NAT. The response to such a HTTP GET request includes the MPD extended by the requested session information corresponding to the provided session key. Alternatively, the MPD server generates a temporary MPD that includes the corresponding session information and redirects the client to this temporary MPD. As clients may be behind a NAT, STUN is employed to determine the public IP address and port number to be used during the synchronization. Every client has to follow a certain procedure before it requests a MPD containing session information. We use the same ports for STUN negotiation with the separate *STUN instance* (cf. Fig. 21.2) as we do for our synchronization protocol.

The public IP (IPv4 or IPv6) address, port number, and NAT type are added along with the session key as URL parameters to the initial HTTP GET message that requests the MPD from the MPD server. The initiation of an IDMS session and the provisioning of the session key are out of scope. Let us assume that the user or the application provides the session key. With the initiation of an IDMS session, an ISO with a specific session key is created. The MPD server adds clients that request a certain MPD with a specific session key to the corresponding ISO. When a client requests a MPD, the *MPD server* adds the client to the ISO associated with the session key and returns both. As clients may join the session at different points in time, each client may only have partial information about the actual number of clients in an IDMS session.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.aau.at/
    DASH/Session" targetNamespace="http://www.aau.at/DASH/Session" xmlns:xlink="http
    ://www.w3.org/1999/xlink">
  <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
  <xs:element name="IDMSSessionObject">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PeerList" type="PeerListType" minOccurs="0"
            maxOccurs="unbounded"/>
        <xs:element name="TTL" type="xs:dateTime" minOccurs="1" maxOccurs="
            1"/>
      </xs:sequence>
      <xs:attribute ref="xlink:href"/>
      <xs:attribute ref="xlink:actuate" default="onLoad"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="PeerListType">
    <xs:sequence>
      <xs:element name="Peer" type="PeerType" minOccurs="0" maxOccurs="
          unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PeerType">
    <xs:sequence>
      <xs:element name="Identifier" type="PeerIdentifierType" minOccurs="1"
          maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PeerIdentifierType">
    <xs:sequence>
      <xs:element name="IP" type="xs:string"/>
      <xs:element name="Port" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="nat" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

Listing 21.1 IDMS session object for MPEG-DASH

Listing 21.1 depicts the XML schema for an ISO. As the definition of the ISO demands that an ISO shall allow an unique numbering of the clients in an IDMS session, the XML schema foresees a list of clients (represented by *@PeerListType*). The definition of the ISO further states that an ISO shall be a time-bounded entity. Therefore, we introduce a TTL element (represented by *@TTL*). The maximum TTL for an IDMS session is the duration of the requested multimedia content which may be in case of a live stream an estimated duration of an IDMS session (e.g., the estimated end time of a soccer match). The *@PeerIdentifierType* contains the public IP address, port number, and the NAT type of a specific client. Note that a client may have more than one identifier if it has several network interfaces that are connected to different networks.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:iso="http://www.aau.at/DASH/
      Session" type="static" mediaPresentationDuration="PT3256S" minBufferTime="PT1.2S"
      profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">
<BaseURL>http://www.example.com/</BaseURL>
<Period>
    <AdaptationSet>
        <Representation id="0" mimeType="video/mp4" codecs="avc1.42c01f,mp4a.40.02"
              startWithSAP="1" bandwidth="251674">
          <SegmentList timescale="1000" duration="10000">
    <Initialization sourceURL="init0.mp4"/>
            <SegmentURL media="seg0-1.m4s"/>
            <!-- ... further segments -->
          </SegmentList>
        </Representation>
        <Representation id="1" mimeType="video/mp4" codecs="avc1.42c01f,mp4a.40.02"
              startWithSAP="1" bandwidth="380974">
          <SegmentList timescale="1000" duration="10000">
    <Initialization sourceURL="init1.mp4"/>
            <SegmentURL media="seg1-1.m4s"/>
            <!-- ... further segments -->
          </SegmentList>
        </Representation>
        <Representation id="2" mimeType="video/mp4" codecs="avc1.42c01f,mp4a.40.02"
              startWithSAP="1" bandwidth="666666">
          <SegmentList timescale="1000" duration="10000">
    <Initialization sourceURL="init2.mp4"/>
            <SegmentURL media="seg2-1.m4s"/>
            <!-- ... further segments -->
          </SegmentList>
        </Representation>
            <!-- ... more representations -->
    </AdaptationSet>
</Period>
<iso:IDMSSessionObject>
    <iso:PeerList>
        <iso:Peer>
            <iso:Identifier nat="NoNAT">
                <iso:IP>143.205.122.242</iso:IP>
                <iso:Port>8029</iso:Port>
            </iso:Identifier>
            <iso:Identifier nat="FullCone">
                <iso:IP>143.205.199.149</iso:IP>
                <iso:Port>8030</iso:Port>
            </iso:Identifier>
        </iso:Peer>
        <iso:Peer>
            <iso:Identifier nat="PortRestricted">
                <iso:IP>10.0.0.5</iso:IP>
                <iso:Port>8029</iso:Port>
            </iso:Identifier>
        </iso:Peer>
        <!-- ... more peers -->
     </iso:PeerList>
     <iso:TTL>2014-07-26T21:32:52</iso:TTL>
</iso:IDMSSessionObject>
</MPD>
```

Listing 21.2 Excerpt of an example MPD

Listing 21.2 shows an excerpt of a MPD comprising an ISO. Clients requesting the MPD will be added to the corresponding ISO. The process of adding clients to the ISO induces an implicit ordering of the clients which is used by the subsequent peer-to-peer synchronization algorithms. Every client numbers the clients in the ISO strict monotonically increasing beginning at one. Even if clients leave the peer-to-peer overlay, their entry in the ISO is not deleted, otherwise removing clients that leave would violate the total order on the clients.

21.2.2 Unstructured Peer-to-Peer Overlay Construction and Synchronization

The information contained in the ISO is used to create a peer-to-peer overlay. As mentioned before, clients may be behind NATs which have to be traversed in order to create a peer-to-peer overlay network and to communicate in a peer-to-peer manner. Therefore, each client communicates with a STUN instance in order to determine whether the client is behind a NAT and, if available, the type of the NAT. We differentiate between the following types of NAT: no NAT, symmetric firewall, full-cone NAT, restricted-cone NAT, port-restricted NAT, and symmetric NAT. In the case of a full-cone NAT, where the mapping is done statically by the NAT such that any address is allowed to send packets by using the public IP address and port number to a client, the communication with the STUN instance has already registered the necessary ports at the client's NAT (for a detailed explanation how STUN is used to identify the NAT types, the interested reader is referred to [21]). If the client detects that its NAT is a restricted-cone NAT or port-restricted NAT, the client's NAT allows only incoming packets from an address if the client has already sent a packet to this address. Therefore, we add a relaying function to the STUN instance which is used by clients with a restricted-cone NAT or port-restricted NAT. The procedure for a client with a restricted-cone or port-restricted NAT is as follows (cf. Fig. 21.3):

1. The client sends a UDP packet to the address with which it wants to communicate using the IP and port signaled by the ISO. This tells the NAT that incoming packets from the destination address are allowed.
2. In case the other client has one of the aforementioned NATs, it uses the relaying function of the STUN instance to signal the other client that it shall send a packet to the given public IP and port.
3. The other client uses the signaled information to open its NAT. After this *handshake*, the peer-to-peer synchronization protocol can be carried out.

This allows to have more than a single client behind the same NAT. If there is no NAT but a symmetric firewall or a symmetric NAT, there is no peer-to-peer communication possible. If the router which hides the clients supports UPNP, it is possible to add forwarding rules for the ports used by the synchronization [23].

UDP is used as the transport protocol between the clients because reliable communication is not essential. Using TCP would require the clients to maintain

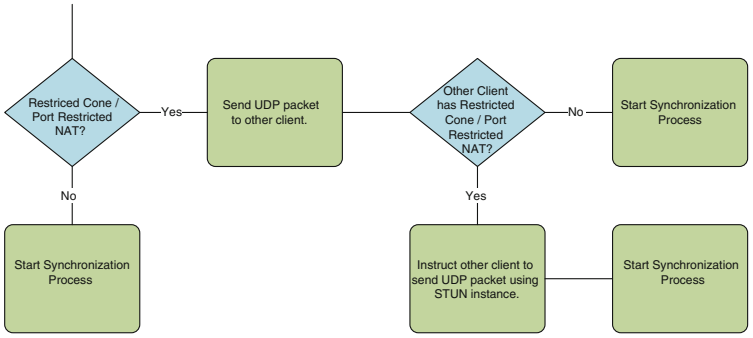


Fig. 21.3 Traversing a restricted-cone or restricted-port NAT between two clients

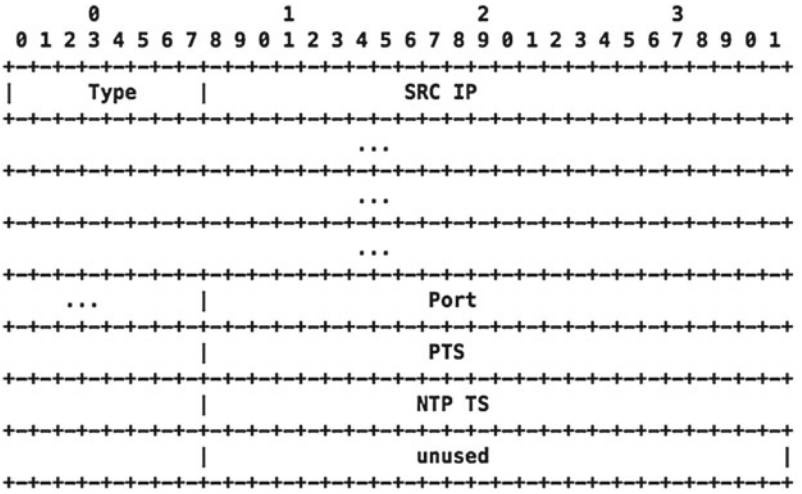


Fig. 21.4 Message structure for the *coarse synchronization*

connections to other clients which implying unnecessary communication overhead. Our algorithms are designed for the use with an unreliable communication (UDP). The only requirement for their correctness is a connected network (no network partitions). Figure 21.4 depicts the message structure for the *coarse synchronization*. This message structure is used for both response and request as specified by the *Type* field. For requests, the fields *IP* and *Port* are set to the public IP address and port number of the requesting client—other fields are empty—which indicates that the receiving client shall respond with its current playback time stamp. The responding client sets its own IP address (field *IP*) and port number (field *Port*) allowing clients to track which clients responded to which requests. The field *PTS* is set to the current playback time stamp, and the field *NTP TS* is the corresponding NTP time stamp. The NTP time stamp is used to align all received time stamps to the same point in time. As the list of clients in the ISO grows over time, clients joining the session early will

only have a subset of available clients. Therefore, if a client is asked for its playback time stamp by an yet unknown client, it adds the associated IP address and port to the list of known clients. Hence, the coarse synchronization has two purposes:

1. perform a coarse synchronization and
2. update the peer-to-peer overlay structure.

Algorithm 1 Coarse Synchronization

```
1:  function request_timestamps
2:      for all p ∈ peers do
3:          sendPacket(Type.Request, p.IP, p.Port, myIP, myPort, null, null)
4:      end for
5:      wait(T_C) ∨ receivedTS.size() = peers.size()
6:      if receivedTS.size() = 0 then
7:          request_timestamps()
8:      else
9:          calculateSegment()
10:     end if
11: end function
12: function receive_request(pt : packet)
13:     if isPeerKnown(pt.srcIP, pt.srcPort) ≠ true then
14:         addPeer(pt.srcIP, pt.srcPort)
15:     end if
16:     sendPacket(Type.Response, pt.srcIP, pt.srcPort, myIP, myPort, PTS, NTPTS)
17: end function
18: function receive_timestamp(pt : packet)
19:     receivedTS.add(pt.PTS, pt.NTP)
20: end function
```

Algorithm 1 implements the *coarse synchronization*. Each client that receives the ISO requests the current playback time stamp from all clients listed in the ISO. In general, we assume that the clocks of the clients are synchronized (e.g., using NTP [3] or PTP [4]). MPEG-DASH provides the possibility to signal the synchronization method within the MPD by using the *@UTCTiming* descriptor. This can be used to signal the clients which protocol and servers they should use to synchronize their clock. The client requesting (i.e., *request_timestamps*) the playback time stamps waits until either all requests have been satisfied or until a given time period T_C has elapsed. If no time stamp arrives during T_C, the client starts over by requesting playback time stamps from the known clients. Each client that receives a request (i.e., *receive_request*) responds with its IP address, port number, playback time stamp, and the corresponding NTP time stamp. The time stamps from the responses may be combined to calculate the start segment using one of four strategies, namely the (i) maximum, (ii) minimum, (iii) average, and (iv) weighted average of the received time stamps. The selection of the strategy depends on the application and may be influenced by QoS parameters like bandwidth, delay, and maximum selectable bit rate of the multimedia content. In our application, we determine the start segment using the average of the received playback time stamps. This procedure provides a

first educated guess on the reference time stamp calculated later on and allows the newly joined client(s) to start streaming multimedia content without waiting until a reference time stamp has been calculated.

Let T_{ref} be the time stamp resulting from such a strategy. We calculate the segment to start with by $\lceil \frac{T_{ref}}{T_s} \rceil$, where T_s is the segment size in seconds (e.g., 1, 2, 3, 4 s, ...). Suppose that M is the theoretical reference time stamp to which all clients will adjust their playback. Therefore, without loss of generality, the asynchronism ξ_j for the j-th client that joins the peer-to-peer overlay after asking the other clients for their playback time stamp and downloading N segments until the playback starts is given by:

$$0 \leq |\xi_j| \leq |M + \sum_{i=1}^{N} \frac{g_j(i + \lceil \frac{T_{ref}}{T_s} \rceil)}{b_{c_j}(i + \lceil \frac{T_{ref}}{T_s} \rceil)} - \lceil \frac{T_{ref}}{T_s} \rceil \cdot T_s|, \tag{21.1}$$

where $b_{c_j}(\cdot)$ is the bit rate (bps) of the transmission channel in bit/s for the $\lceil \frac{T_{ref}}{T_s} \rceil + i$-th segment and $g_j(\cdot)$ is the file size (bits) of the $\lceil \frac{T_{ref}}{T_s} \rceil + i$-th segment of the multimedia content of the j-th client. The fraction $\frac{g_j(i)}{b_{c_j}(i)}$ provides the time needed for downloading the i-th segment. Summing up the time needed for downloading N segments provides the initial start-up time of the j-client. The client will start its playback at $\lceil \frac{T_{ref}}{T_s} \rceil$ but we have to account for the time the client needs to fetch enough initial DASH segments. The *coarse synchronization* ensures that if a client joins an IDMS session, it starts with a segment that is as closest as possible to the segment the other clients are currently playing, especially if the other clients have not yet agreed on a reference playback time stamp.

Algorithm 2 Merge and Forward—Broadcast to Neighbors

$B_i, L_i, BL_i, P_i, NTP_i, I_i^M, I_i^m, S_i, C_i \leftarrow 1$
1: $update(P_i, NTP_i)$
2: **for all** $p \in peers$ **do**
3: $sendPacket(P_i, NTP_i, I_i^M, I_i^m, S_i, C_i, B_i)$
4: **end for**

The fine synchronization phase starts once playback commences at the segment determined by the *coarse synchronization*. Fine synchronization pursues the goal of an agreement on a reference time stamp for a IDMS session. We propose *Merge and Forward*, a flooding-based (periodically sending information to a selected subset of direct neighbors) algorithm calculating the (weighted) average playback time stamp among the clients in a distributed and self-organized manner. The average playback time stamp is utilized because of fairness considerations: The average neither favors the clients already in a IDMS session nor those that have just joined. Selecting the

Algorithm 3 Merge and Forward—Receive Bloom Filter

$B_i, L_i, BL_i, P_i, NTP_i, I_i^M, I_i^m, S_i, C_i \longleftarrow 1$
if $S_j > S_i$ **then**
  $L_i \leftarrow \emptyset, S_i \leftarrow S_j, C_i \leftarrow 1$
  $B_i \leftarrow H(i)$
end if
if $bits(B_j) > bits(B_i)$ **then**
  $increaseSize(B_i, bits(B_j))$
end if
$update(P_i, NTP_i)$
$update(P_j, NTP_j)$
if $B_i \oplus B_j \neq 0 \wedge B_i \cap B_j = \emptyset \wedge B_j \notin L_i$ **then**
  $B_i \leftarrow B_i + B_j, P_i \leftarrow \frac{P_i \cdot C_i + P_j \cdot C_j}{C_i + C_j}$
  $I_i^M \leftarrow \max\{I_i^M, I_j^M\}$
  $I_i^m \leftarrow \min\{I_i^m, I_j^m\}$
  $C_i \leftarrow C_i + C_j$
end if
if $B_i \oplus B_j \neq 0 \wedge B_i \cap B_j \neq \emptyset \wedge B_j \notin L_i$ **then**
  if $C_j \geq C_i \wedge i \in B_i \cap B_j$ **then**
    $B_i \leftarrow B_j, P_i \leftarrow P_j, C_i \leftarrow C_j$
  else if $C_j \geq C_i$ **then**
    $B_i \leftarrow B_j + H(i)$
    $P_i \leftarrow \frac{P_j \cdot C_j + P_j}{C_j + 1}$
    $C_i \leftarrow C_j + 1$
  end if
  $I_i^M \leftarrow \max\{I_i^M, I_j^M\}$
  $I_i^m \leftarrow \min\{I_i^m, I_j^m\}$
end if
if $\#B_i - C_i > 0$ **then**
  $increaseSizeAndTest(B_i, BL_i, C_i, I_i^M, I_i^m)$
  return
end if
$L_i \leftarrow \{B_j\} \cup L_i$
$BL_i \leftarrow \{B_i, C_i, P_i, NTP_i\}$

minimum would privilege clients that recently joined an IDMS session, and it will force all other clients to synchronize to this playback time stamp. Clients that join an IDMS session are likely to start playback at a lower time stamp (due to initially downloading N segments) while the playback time stamps of the other clients increase gradually. The maximum will have the opposite effect. *Merge and Forward* focuses on reducing the overhead introduced when exchanging playback time stamps using unicast between all clients in contrast to other algorithms which use multicast [13] or even broadcast. Furthermore, by avoiding *pure flooding (sending information to all neighbors)*, it maintains media throughput and thus the QoE.

In order to track the clients which have already contributed to the average timestamp, one may simply use a bit field with the bit positions mapped to the (unique) client identifiers. However, if clients leave or (many) new clients join the IDMS

session, bits may be unused or the bit field may be too small. In order to overcome this shortcoming, we suggest to use a Bloom filter. Using a Bloom filter allows us to use packets of fixed length for communication contributing to the scalability of our peer-to-peer approach. Nevertheless, the drawback of Bloom filters is that they are probabilistic data structures. A Bloom filter uses hash functions $h_1(x)$, ..., $h_k(x)$ for calculating which bits have to be set for a client with id x [24]. We suggest to use hash functions with a low collision probability (e.g., Murmur Hash, SHA-1, SHA-2). For our purpose, we used the Secure Hash Algorithm-1 (SHA-1) as hash function. Figure 21.5 depicts the message structure for the peer-to-peer communication once the overlay has been constructed having the following semantics:

- ***ATS***: the current average playback time stamp (of the peers in the Bloom filter) calculated by this client.
- ***NTP TS***: NTP time stamp for aligning the playback time stamp.
- ***Lowest PeerID***: lowest *PeerID* seen by the sending client according to the ISO.
- ***Highest PeerID***: highest *PeerID* seen by the sending client according to the ISO.
- ***Sequence Number***: number of current synchronization round.
- ***Cumulative Count***: number of clients that contributed to the reference playback time stamp.
- ***Bloom filter***: fixed length Bloom filter with a length of m bits.

The *PeerID* fields are used for determining how many clients are in a certain Bloom filter. The *Sequence Number* allows the clients to trigger a re-synchronization by increasing the sequence number (e.g., due to asynchronism or MPD update). Other clients receiving a message with a higher sequence number than their own sequence number will reset to the initial condition and start over. The overall size of such a message is $32 + \frac{m}{8}$ bytes. The peer-to-peer algorithm making use of this message structure is shown in Algorithm 2 and Algorithm 3 and referred to as *Merge and Forward (M&F)*. *Merge and Forward* is named after its operations because it **merges** incoming Bloom filters and **forwards** them to its neighbors. Each client i maintains the following variables and lists (cf. Algorithms 2 and 3):

- B_i denotes the Bloom filter, each client initially inserts itself with its own *PeerID* according to the indices obtained by the use of a common set of k hash functions $h_1(x)$, ..., $h_k(x)$, and the actual index is then determined by $f_n(x) = h_n(x) \ \ MOD \ \ size$, $1 \leq n \leq k$, where *MOD* denotes the modulo operation and size denotes the actual size of the Bloom filter in bit (henceforth we use $h_n(x)$ interchangeably to $f_n(x)$).
- L_i denotes the list of already seen Bloom filters.
- P_i denotes the PTS.
- NTP_i denotes the NTP time stamps.
- S_i denotes the current sequence number ($S_i \in \mathbb{N}$, initially set to zero).
- C_i denotes the cumulative count which is initially set to zero.
- I_i^m denotes the lowest *PeerID* seen by the i-client.
- I_i^M denotes the highest *PeerID* seen by the i-client.
- $H(x)$ denotes the function that generates the bit sequence for the x-client by applying $h_1(x)$, ..., $h_k(x)$.

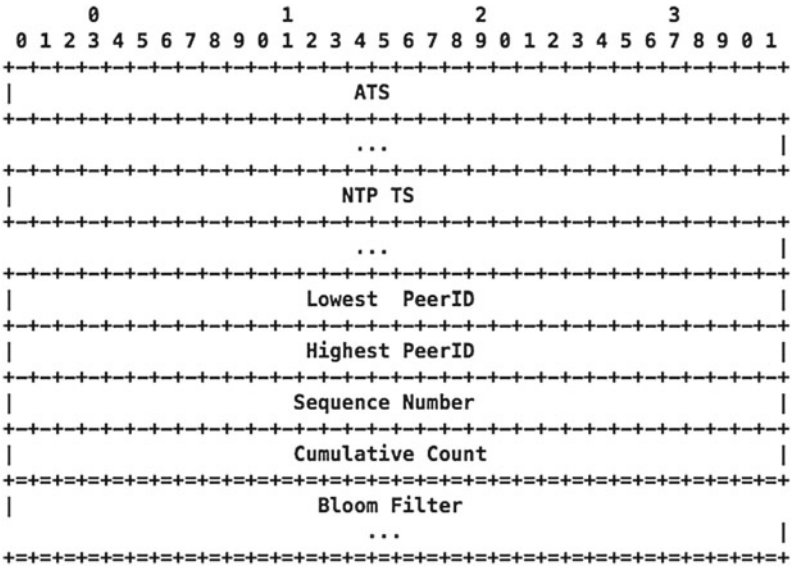


Fig. 21.5 Message structure for the *fine synchronization*

Each client **forwards** B_i, P_i, I_i^M, I_i^m, and C_i periodically to its neighbors depicted by the function *broadcastToNeighbors*, with period τ (cf. Algorithm 2 Line 1 to 4). Please note that the period must not be the same for every client. If client i receives a message from one of its neighbors, it first checks whether the sequence number S_j is greater than S_i. If this is the case, the ith client clears its Bloom filter and re-initializes all variables and sets its sequence number to the received one (cf. Algorithm 3 Line 1 to 4).

If client i receives a Bloom filter from one of its neighbors j, it first compares the size of its Bloom filter and the received Bloom filter and increases the size of its Bloom filter to the size of the received Bloom filter if the received one is bigger (cf. Algorithm 3 Line 5 to 7). Furthermore, client i checks whether it can **merge** the Bloom filters (cf. Algorithm 3 Line 10 to 26). The Bloom filters can only be merged if they are disjoint avoiding introducing a bias to the resulting weighted average. If the Bloom filters B_i and B_j are distinct in terms of $B_i \cap B_j = \emptyset$ (cf. Algorithm 3 Line 10), then the Bloom filters can be merged using the bit-wise *OR* depicted by $+$ and we calculate the weighted average between P_i and P_j (cf. Algorithm 3 Line 11). C_i denotes how many clients already contributed to the average playback time stamp and in the case, if two distinct Bloom filters are merged, the cumulative count is calculated according to Algorithm 3 Line 14. If the Bloom filters are not disjoint and if we have not seen the received Bloom filter yet, we have two further cases (cf. Algorithm 3 Line 10). First, if $C_j \geq C_i$ and if client i is already in both of the Bloom filters B_i and B_j, we store the more recent one (cf. Algorithm 3 Line 17 and Line 18). If $C_i \geq C_j$ but $i \notin B_i \cap B_j$, client i adds itself to B_j and calculates the weighted average (cf. Algorithm 3 Line 19 to Line 22).

The highest *PeerID* I_i^M is set to the maximum of I_i^M and I_j^M, and I_i^m is set to the minimum of I_i^m and I_j^m (cf. Algorithm 3 Line 12 to 13 and Line 24 to 25). After a synchronization round has finished (all clients hold the same reference playback time stamp), a client may trigger a new synchronization round by increasing S_i. The re-synchronization may be triggered if a client has paused the playback of the multimedia content or if it is unable to synchronize its playback to the negotiated reference. In the latter case, the client may increase the importance of its time stamp by introducing a weight (e.g., $PTS = w_i \cdot PTS$, $w_i \geq 1$). C_i depicts the number of clients that have already contributed to P_i. BL_i denotes the backlog where the Bloom filters that have not caused any false positives are stored (cf. Algorithm 3 Line 39).

In order to calculate the overlap or the intersection of two Bloom filters ($B_i \cap B_j$), *Merge and Forward* has to know **which clients** have already been inserted. Due to the nature of Bloom filters, the calculation of the number of clients in a Bloom filter may identify clients that were not inserted (so-called false positives). Consider a test function $test(B, x)$ that returns true if client x was inserted into the Bloom filter. Let us assume we know that client x is not in Bloom filter B and that when we receive the Bloom filter of size m, s bits are set by the use of k hash functions. If we test whether client x was inserted into the Bloom filter and if this test returns true, we have encountered a false positive. Furthermore, we assume that the probability that a bit is set by a hash function is distributed uniformly and independently with $p = \frac{1}{m}$. The probability that an already set bit is set by $h_i(x)$ assuming that s bits are set is $\sum_{t=1}^{s} \frac{1}{m} = \frac{s}{m}$. If we receive a Bloom filter with s bits set and if we test the existence of a specific client, the probability to encounter a false positive is $p_{false} := P(\bigcap_{i=1}^{k} h_i(x)) = \prod_{i=1}^{k} \frac{s}{m} = (\frac{s}{m})^k$ [24]. For reducing the probability of encountering a false positive, we have introduced I_j^M and I_j^m. Especially if a client receives the Bloom filter containing only a single client, then $I_j^M - I_j^m = 0$ and, therefore, only the client with id I_j^M or I_j^m is in the Bloom filter. Figure 21.6 depicts a simplified example of how *Merge and Forward* converges. The arrows only indicate messages

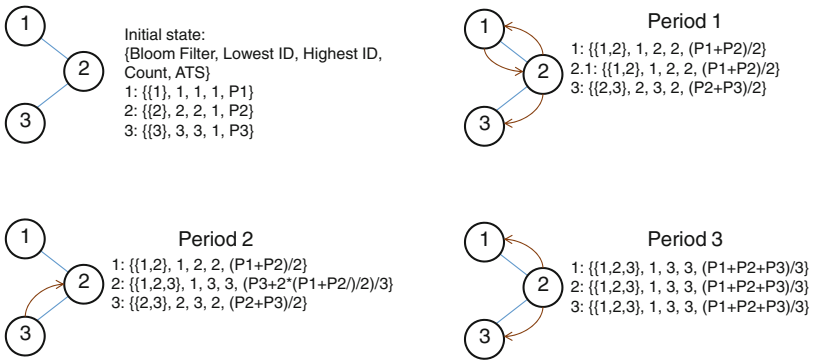


Fig. 21.6 Simplified example on how Merge and Forward converges

that cause an effective change in the clients data structures according to Algorithm 3. Assume that in the first period only clients with id 1 and 2 are sending messages to their neighbors. Since the Bloom filters are disjoint, both nodes add the received Bloom filter to their existing one and they further calculate the average playback time stamp, set the highest and lowest peer ID, and calculate the cumulative count. After the second period, the only message that causes a change is the message sent from client 3 to client 2. Client 2 *merges* the Bloom filters and is able to calculate the average playback time stamp. In the third period, clients 1 and 3 accept the Bloom filter send by client 2 because it contains more clients than their current Bloom filter.

21.2.3 Dynamic Adaptive Media Playout for IDMS

Having the synchronization on the network level in place leaves us with the question how we can carry out the synchronization of the playback. Related work [17] has shown that pausing/skipping has a tremendous negative impact on the QoE. Thus, we want to avoid pausing/skipping multimedia content at all costs with only one important exception. That is when there is no audio and black frames where we could easily skip the frames or pause the playback such that the user would not notice it. We want to find those sections of the multimedia content where we can alter the playback rate with the lowest impact on the QoE. In order to quantify the *distortion* caused by altering the playback rate, we use content features such as the motion intensity for the video domain and the volume of the audio domain. Therefore, we define a metric $d_v : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$ that allows measuring the playback rate variations with respect to the (visual) motion intensity and the (audio) spectral energy. We derive the visual feature (motion intensity) from the average length of the motion vectors between two consecutive frames, whereas the audio feature is extracted from the spectral energy of the audio frames (for each channel). We compare these audiovisual (AV) features from a given content section with a given duration (e.g., 2s) for the increased/decreased playback rate and the nominal playback rate. As the length of the asynchronism is known, both the target playback rate and the duration of the playback rate change can be calculated. Importantly, introducing an increase/decrease in the playback rate will alter the duration of the respective content section which may or may not be perceived by the user. In order to measure the distortion in the video domain, we use the difference of the motion intensity between the altered playback and the unmodified playback defined in Eq. 21.2. The proof that the metrics introduced here follow the definition of a metric is left to the interested reader as exercise.

$$d_v(\mathbf{x}, \mathbf{y}) = \left| \sum_{j=x_1}^{x_1+\lfloor(x_2-x_1)\cdot x_3\rfloor} f_v(j) - \sum_{j=y_1}^{y_1+\lfloor(y_2-y_1)\cdot y_3\rfloor} f_v(j) \right| \tag{21.2}$$

where x_1 denotes the number of the first frame of the content section for which the playback rate should be changed. $x_1 + \lfloor(x_2 - x_1) \cdot \Delta\mu\rfloor$ is the index of the last frame

when using the increased/decreased playback rate $\Delta\mu$, x_2 denotes the last frame of the selected content section, and f_v denotes the average motion intensity. For our purpose we set $\mathbf{y} = (x_1, x_2, 1)^T$. This *compares* the standard playback to the altered playback (increased/decreased playback rate) in the video domain. The audio metric $d_a : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$ is defined in Eq. 21.3.

$$d_a(\mathbf{x}, \mathbf{y}) = |\sum_{c=1}^{C}((\sum_{j=x_1}^{x_1+\lfloor(x_2-x_1)\cdot x_3\rfloor}\sum_{k=0}^{S_f}|\widehat{a}_{c_u}(k)| - \sum_{j=y_1}^{y_1+\lfloor(y_2-y_1)\cdot y_3\rfloor}\sum_{k=0}^{S_f}|\widehat{a}_{c_u}(k)|))| \tag{21.3}$$

where C denotes the number of available audio channels and S_f denotes the highest frequency. x_1, x_2, and x_3 are defined in the same way as for the video metric. We set again $\mathbf{y} = (x_1, x_2, 1)^T$. The Fourier-transformed audio frames are denoted by $\widehat{a}_c$ (we use a half overlapping Hamming window) for a given audio channel c. We further define the two transform functions $h_v, h_a : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ which transform time stamps into frame numbers for the video (h_v) and the audio (h_a) domains as follows. For the video domain, we define $h_v(\mathbf{x}) = (x_1 \cdot fps_v, x_2 \cdot fps_v, x_3)^T$, where fps_v denotes the frames per second of the video content. For the audio domain, we define $h_a(\mathbf{x}) = (x_1 \cdot fps_a, x_2 \cdot fps_a, x_3)^T$, where fps_a denotes the frames/samples per second of the audio content. The combined AV metric is defined in Eq. 21.4.

$$g(\mathbf{x}, \mathbf{y}) = \sqrt{d_v(h_v(\mathbf{x}), h_v(\mathbf{y}))^2 + d_a(h_a(\mathbf{x}), h_a(\mathbf{y}))^2} \tag{21.4}$$

$g(\mathbf{x}, \mathbf{y})$ is neither convex nor continuous because it depends on the features which are derived from the actual content.

Since we have now a possibility to quantify *distortion* in the perceptual domain using content features, the question arises how do we find appropriate content sections using the given metric and adding some constraints. Finding only the content section that provides the lowest *distortion* is not constructive because the content section could be located at the end of the movie while the users just started watching it. Our goal is it to synchronize playback as soon as possible with the lowest distortion as possible. Therefore, we present at first a generalized problem formulation:

$$\arg\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{21.5a}$$

$$x_2 \cdot (x_3^{sign(\xi)} - 1) \cdot sign(\xi) = |\xi| \tag{21.5b}$$

$$L \leq B - x_2 \cdot x_3 + x_2 \cdot \frac{b_c}{b_r} \tag{21.5c}$$

$$x_1 \leq T \tag{21.5d}$$

$$x_2 \leq t_{max} \tag{21.5e}$$

where $\mathbf{x} \in \mathbb{R}^3$ is our vector with $(x_1, x_2, x_3)^T$, x_1 denotes the starting time of the playback rate change relative to the current buffer, x_2 denotes the duration of the

playback rate change, and x_3 denotes the target playback rate. Our aim is to find values for $\mathbf{x}$ such that $\mathbf{x}^*$ is a minimizer for $f(\mathbf{x})$ ($\forall \mathbf{x} \in \mathbb{R}^3 : f(\mathbf{x}^*) \leq f(\mathbf{x})$). $f(\mathbf{x})$ can be any function that models the impact of changing the playback rate for the given duration on the QoE. Furthermore, we define constraints that reduce the set of feasible points. First, we introduce the constraint as depicted by Eq. 21.5b which states that the given asynchronism should be compensated for by selecting appropriate values for x_2 and x_3, respectively. ξ denotes the asynchronism identified by comparing the current playback time stamp to the reference time stamp. If $\xi < 0$ the playback rate is reduced and if $\xi > 0$ the playback rate is increased. Equation 21.5c avoids buffer underflows and, thus, stalls in the multimedia playback. This constraint only applies if the playback rate is increased. In particular, L denotes the lower buffer threshold in seconds, B the current buffer fill state in seconds, b_c the client's bandwidth, and b_r the bit rate of the selected representation. Furthermore, we constrain the starting time (T) of the playback rate variation by bounding x_1 (cf. Eq. 21.5d). Equation 21.5e limits the duration (t_{max}) of the playback variation.

The highest asynchronism in our IDMS system is given by Eq. 21.1. The initial asynchronism of a newly joined client depends on the time the client requires for downloading sufficient segments such that it can start the playback. If the asynchronism is greater than the current buffer fill state, the client will not be able to compensate the asynchronism. In such cases, the synchronization process must be partitioned into a number of smaller synchronization processes. Therefore, we define the asynchronism for each synchronization process as $\delta_{k+1} = \min(\xi - \delta_k, B - L)$, starting with $\delta_0 = \min(\xi, B - L)$. This has no effect if ξ is lower than zero because reducing the playback rate does not affect the buffer fill state. Using the sequential unrestricted minimization technique, we transform the general optimization problem given in Eq. 21.5 into the following optimization problem:

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \begin{cases} g(\mathbf{x}, (x_1, x_2, 1)^T) + \gamma \cdot \sum_{i=1}^{3} p_i(\mathbf{x}) \text{ if } \xi \geq 0 \\ g(\mathbf{x}, (x_1, x_2, 1)^T) + \gamma \cdot \sum_{i=2}^{3} p_i(\mathbf{x}) \text{ if } \xi < 0 \end{cases} \tag{21.6a}$$

$$p_1(\mathbf{x}) = \min\{0, B - x_2 \cdot x_3 + x_2 \cdot \frac{b_c}{b_r} - L\}^2 \tag{21.6b}$$

$$p_2(\mathbf{x}) = \min\{0, T - x_1\}^2 \tag{21.6c}$$

$$p_3(\mathbf{x}) = \min\{0, t_{max} - x_2\}^2 \tag{21.6d}$$

Equation 21.6a shows the transformed cost function. To reduce the set of feasible points, we use the constraint given in Eq. 21.5b and apply the implicit function theorem, reducing x_2 to a function of x_3 by $u(x_3) = sign(\delta_k) \cdot \frac{|\delta_k|}{x_3^{sign(\delta_k)} - 1}$ for $x_3 \neq 1$ [*Proof*: by application of the implicit function theorem on $f(x_3, x_2) = x_2 \cdot (x_3^{sign(\delta_k)} - 1) \cdot sign(\delta_k) - |\delta_k|$]. Thus, we try to find values x_1 and x_3 that minimize $f(\mathbf{x})$. The penalty function $p_1(\mathbf{x})$ (cf. Eq. 21.6b) states that the buffer fill state shall not be drained below the threshold L when increasing the playback rate. $p_2(\mathbf{x})$ (cf. Eq. 21.6c) depicts the costs that depend on the starting point of the playback rate variation. $p_3(\mathbf{x})$ (cf. Eq. 21.6d) states that a client should be synchronized within

t_{max} seconds. γ denotes the penalty factor for the transformed constraints ($\gamma > 0$). For $\gamma \to \infty$ the solution found will be exact. Since we run into numerical problems when setting $\gamma \to \infty$, we have to find a trade-off between exactness and fulfilling the constraints. For solving the optimization problem during the multimedia playback, we use multiple instances of the Nelder–Mead algorithm with different starting points (equally distributed) which yields a set of local minima E [25]. There may be more than a single optimal solution that minimizes the impact on the QoE because $f(\mathbf{x})$ is not convex since different values for $\mathbf{x}$ may yield the same values for $d_a(\mathbf{x})$ and $d_v(\mathbf{x})$ depending on the multimedia content and it is not continuous, and this can be easily verified because the features used are not continuous in time. We then select $\mathbf{x}^* = \arg\min_{\mathbf{x}}\{E\}$ as a minimizer of $f(\mathbf{x})$. Our dynamic AMP approach overcomes asynchronism by searching for content sections where the playback rate may be increased or decreased having the least impacting on the QoE.

21.2.4 Evaluation of Merge and Forward

To evaluate the performance of *Merge and Forward*, we compare our algorithm to an approach that is similar to the one described in [13]. As this approach uses multicast, we modify it such that each client aggregates all received playback time stamps into a single message and sends them periodically to its neighbors using unicast because we do not assume that multicast is in place. We call this approach *Aggregate*. We compare the two algorithms based on the overhead produced by clients agreeing on the average playback time stamp and the duration of the process. The evaluation scenario foresees that a certain number of clients will join the peer-to-peer overlay over the lifetime of the session. If the clients join one by one after the existing peers have already synchronized, the time required by *Merge and Forward* to synchronize is the same as *Aggregate*. Furthermore, clients may leave the peer-to-peer overlay during the negotiation process as any reference time stamp they have contributed persists until a re-synchronization is triggered. We simulated the algorithms on Erdös Rényi random networks [26] implemented using OMNeT++ [27]. A period τ of 250 ms was used for both algorithms, with the round trip time set to 300 ms and a maximum clock skew of 30 ms randomly (uniformly) chosen from an interval of $[-15, 15]$. For *Merge and Forward*, we set the size of the Bloom filter to 512 bits. We generate random networks for the following number of clients: 40, 60, and 80 with following probabilities for creating a connection between two clients: 0.1–0.9 increased in 0.1 steps. We use these numbers of clients and connectivities in order to show how the algorithms scale when increasing the number of clients and their connectivity. For each of the parameter settings, we conducted 30 simulation runs and take the average of the results.

Figure 21.7 illustrates the average network traffic (in kbit) generated at each client by both algorithms during the agreement phase with respect to the overall connectivity of the network. The connectivity is given by the ratio of the average node degree

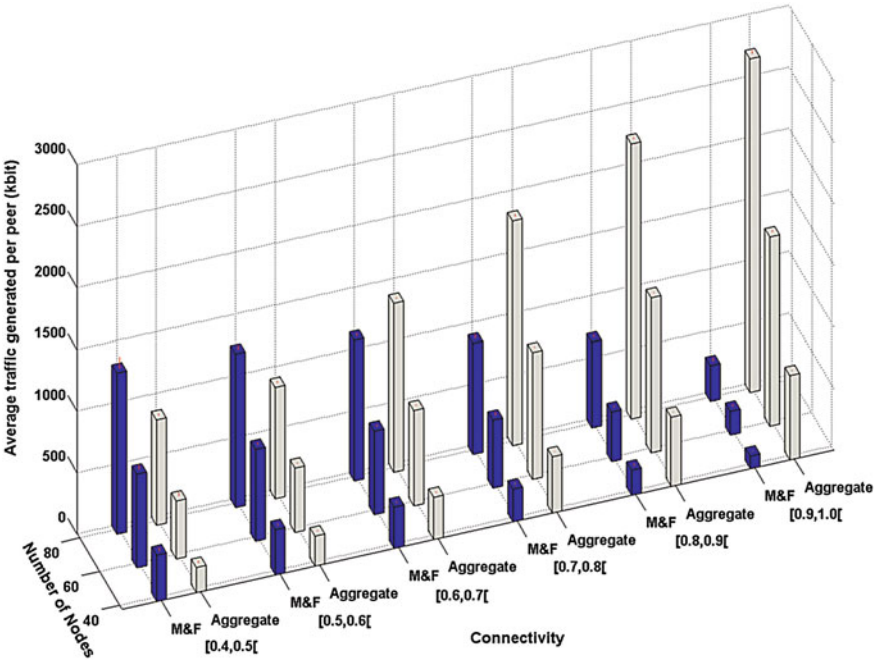


Fig. 21.7 Traffic overhead per client until all peers have the same reference playback time stamp using *Merge and Forward* (M&F) and *Aggregate* (95% CI)

and the number of clients in the network ($c = \frac{\frac{1}{|V|}\sum_{i=1}^{|V|} d_G(v)}{|V|-1}$, V is the set of vertices and $d_G(v)$ is the degree of node $v \in V$). The x-axis represents connectivity in intervals which increase in 0.1 steps. With low connectivity, *Merge and Forward* generates more traffic than *Aggregate*, but this is due to the time required to compute the average playback time stamp among all clients. If the connectivity increases, the fixed length messages of *Merge and Forward* start to pay off and it subsequently outperforms the *Aggregate* algorithm.

Figure 21.8 depicts the time for the distributed calculation of the average playback time stamp. Here, *Aggregate* represents the optimum case because lists of time stamps can be merged even if they overlap because the contribution of each client can be clearly identified. Using Bloom filters and only a single field for the average playback time stamp, the contribution of a single client cannot be uniquely identified anymore. If a peer receives a Bloom filter where a subset of the clients contributed to the received one and to one that the clients holds, calculating the weighted average would yield a skewed weighted averaged compared to the real weighted average without the overlap. However, with an increase in the connectivity, the time needed by *Merge and Forward* converges to the time needed by *Aggregate*. This shows that the negotiation time required by *Merge and Forward* does not solely depend on the number of clients in the overlay.

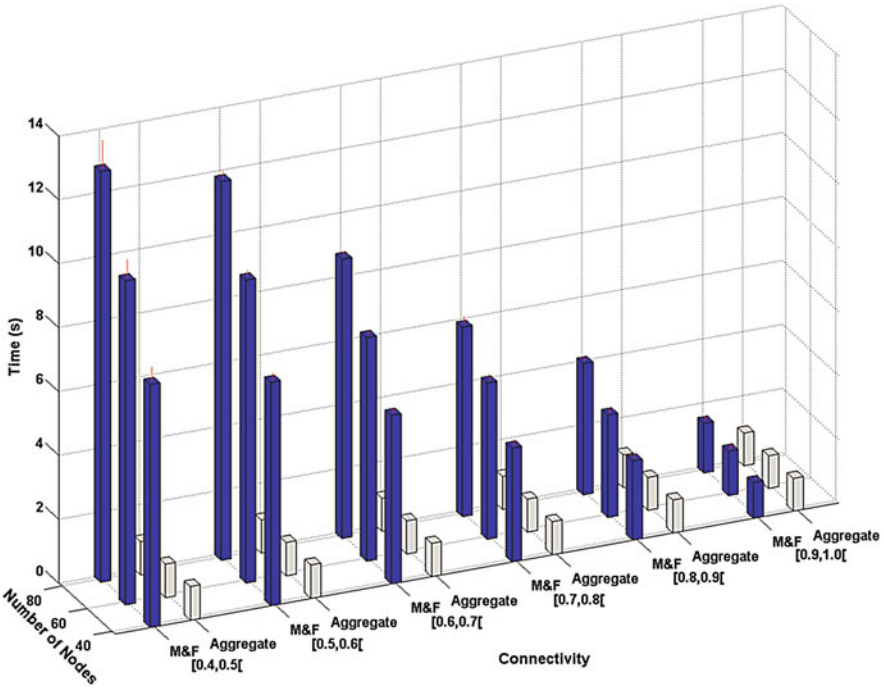


Fig. 21.8 Time needed for the distributed calculation of the average playback time stamp using *Merge and Forward* (M&F) and *Aggregate* (95% CI)

Combining Figs. 21.7 and 21.8 yields the average traffic in kbit/s generated at each client by both algorithms during the agreement phase, *Merge and Forward* always outperforms *Aggregate*. If a client does join an IDMS session, the whole process of calculating the reference playback time stamp is restarted. Compared to *Merge and Forward*, using *Aggregate* or any other pure flooding algorithm will generate more overhead as the number of clients or connectivity increases. The assumption that *Merge and Forward* has to calculate the average among all playback time stamps and the low overhead results in an increase in the time until all clients hold the same time stamp.

21.2.5 Evaluation of the Dynamic Adaptive Media Playout

In order to validate whether our approach introduced in Sect. 21.2.3 reflects the QoE, we evaluate our dynamic AMP approach by conducting a subjective quality assessment using crowd sourcing [28]. The aim is to determine how the AV metrics introduced in Eqs. 21.2, 21.3, and 21.4 reflect the actual impact on the QoE when increasing or decreasing the playback rate.

In order to conduct the user study, we used the Microworkers [29] crowd sourcing platform which hosts so-called campaigns to which users (microworkers) can subscribe. These campaigns include a detailed description of the task and require participants to submit proof that they participated in the campaign. The user study was designed with a duration of 15 min and offered US$0.25 as a reward, slightly more than the typical US$0.20 paid for such a campaign. The design of our study is as follows: (1) We present an introduction where we explain the task and the test procedure in detail. (2) A pre-questionnaire is presented to the subject which asks for demographic data. (3) A short training phase is conducted in order to mitigate the surprise effect. (4) The main evaluation takes place using a single stimulus method as defined in [30]. (5) At the end, a post-questionnaire is presented which allows for comments. (6) A unique token is provided as proof for a successful participation.

For the training phase, we selected a short video sequence from Babylon A. D. The training sequence is presented with three different media playback rates $\mu \in \{1, 0.5, 2\}$ in order to introduce AMP to the subject and the corresponding effect on the playback. A playback rate of one depicts the nominal playback rate of the video sequence (e.g., 25 fps). A playback rate of 2 is twice as fast as the nominal playback rate, and 0.5 is half the nominal playback rate. The stimulus for the main evaluation is the first 51 s of Big Buck Bunny (http://www.bigbuckbunny.org). This content provides high and low motion scenes as well as scenes with high and low audio volume. We introduce four content sections for which the playback rates are increase and decreased. The playback time stamps for the selected content sections in seconds are as follows (start-end): 6.4–7.2, 9–10, 16–18, 35–38, and 46–49. These content sections are presented with following playback rates: 0.5, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2. The multimedia player that is employed uses the Waveform Similarity-based Overlap-Add algorithm [31] that tries to preserve pitch for the audio domain while increasing or decreasing the playback rate. For rating the QoE, we use a continuous rating scale $[0, 100]$ displayed as a slider. Furthermore, we randomly insert a control question after a stimulus which asks the participants what they have seen in the previous video sequence.

For the statistical analysis of the results, we screened the subjects according to [30] and specifically removed participants who failed to correctly answer to the control question. This yielded 55 (48 males and 7 females) subjects for the statistical analysis out of a total of 80 participants. The QoE ratings for each stimulus were subject to a Shapiro–Wilk test to assess whether the ratings were normally distributed. The null hypothesis ($H0$), stating that no normal distribution is present, was rejected for each configuration of playback rates. Furthermore, we calculate the average distortion $\overline{g(\mathbf{x}, (x_1, x_2, 1)^T)}$ for each stimulus. Figure 21.9 depicts the assessed Mean Opinion Score (MOS) for each tuple $(\overline{g(\mathbf{x}, (x_1, x_2, 1)^T)}, \mu)$. The hidden reference condition is given by $\mu = 1$. The results state that playback rates below (i.e., $\mu = 0.8$) and above (i.e., $\mu \in \{1.2, 1.4, 1.6, 1.8\}$) the nominal playback rate have no significant impact on the QoE. This confirmed a Student's t-test for equal sample variances which showed no significant difference in MOS between the reference

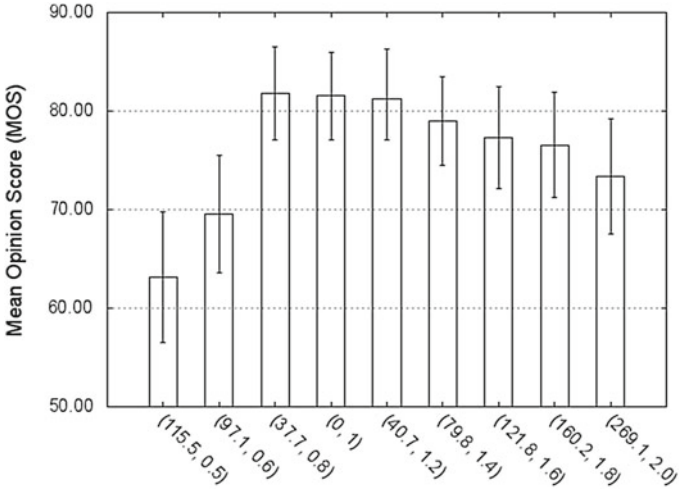


Fig. 21.9 MOS and 95% CI for $\overline{(g(\mathbf{x}, (x_1, x_2, 1)^T), \mu)}$

condition and the playback rates.[1] These results indicate that the subjects could not notice a significant difference for playback rates within the range of [0.8, 1.8]. For the other playback rates, the QoE significantly degrades. A Student's t-test revealed that there is a statistical significant difference between the MOS of the reference condition and the playback rates.[2]

Finally, we investigated how well our distortion metric defined in Eq. 21.4 correlates with the assessed MOS. For playback rates greater than the nominal playback rate, the Pearson correlation coefficient is $\rho = 0.975$ with the probability of encountering a false positive being $p = 0.0009$. Playback rates lower than the nominal playback rate exhibit a high negative correlation with $\rho = -0.995$ and $p = 0.0047$. This indicates a strong linear correlation between our distortion metric and the MOS assessed by the subjective quality assessment, supporting our approach to the optimization problem as stated in Sect. 21.2.3. Furthermore, such an optimizer will always perform equal or better than skipping or pausing the multimedia playback in terms of QoE.

[1] $\mu = 0.8 : p = 0.93, t = -0.083; \mu = 1.2 : p = 0.92, t = 0.096; \mu = 1.4 : p = 0.81, t = 0.42; \mu = 1.6 : p = 0.22, t = 1.23; \mu = 1.8 : p = 0.16, t = 1.41$ for $\alpha = 5\%$.

[2] $\mu = 0.5 : p = 0.00, t = 4.5217; \mu = 0.6 : p = 0.002, t = 3.2; \mu = 2 : p = 0.03, t = 2.19$ for $\alpha = 5\%$.

21.3 Crowd Sourcing IDMS User Studies

Evaluating the influence of asynchronism on the QoE can be a tough task as pointed out in [1]. This is especially the case for in-lab subjective evaluations. The test setup needs careful planning, and minor deviations may result in huge influence factors. Therefore, we introduce and discuss a methodology that employs the concepts of *games with a purpose* for conducting subjective quality assessments in the field of IDMS using crowd sourcing. We will discuss how such a *game with a purpose* for IDMS could look like and to which design principles one should adhere. We will validate this new methodology by conducting a subjective quality evaluation and investigating the following research question:

What is the lower threshold on asynchronism for IDMS?
In order to answer this research question, we adopt concepts from *human computation* to develop a reaction game which allows us to conduct a *crowd-sourced SQA*. We will introduce and describe the design of the reaction game that provides the possibility to mimic the scenario of online quiz shows. Using this game, we evaluate the impact of asynchronism on the overall *QoE*, *togetherness*, *fairness*, and *annoyance*. Quality of Experience denotes the overall delight of the users with the game. Togetherness refers to the perceived quality of being together. Fairness aims at measuring the users' perception whether they are treated equally. Annoyance refers to the users' state of being annoyed [1]. These are subjective measures and shall reflect the opinion of the users, and they are measured on a scale from 0 to 100. The results of the SQA indicate the lower asynchronism threshold for which the QoE decreases significantly. The novelty in this approach is the adoption of human computation—specifically, GWAP—for evaluating the QoE of IDMS use cases.

21.3.1 Assessment Methodology, Stimuli, and Reaction Game

For the assessment of the asynchronism threshold for IDMS, we introduce a reaction game following the GWAP principle. The game design was refined based on an in-lab study. This section describes the game design, SQA methodology (incl. stimuli), and the platform used for the SQA. The reaction game which has been designed for conducting the SQA follows a simple principle such as quiz shows do. In our game, two players have to collaborate in order to achieve the highest possible score. Therefore, we introduce time-bounded events during which both players have to *click* onto the canvas that is displaying the video sequence. Figure 21.10 depicts the time sequence beginning by the start of a video sequence, occurrence of an event, and the different possibilities to click during a given time window. An event is signaled by displaying an attention symbol in the upper left corner of the video sequence (cf. Fig. 21.11). If the multimedia playback of both players is synchronous, the time window for being awarded with additional (bonus points) score equals to the total duration of the occurred event. With an increase in asynchronism, the time window

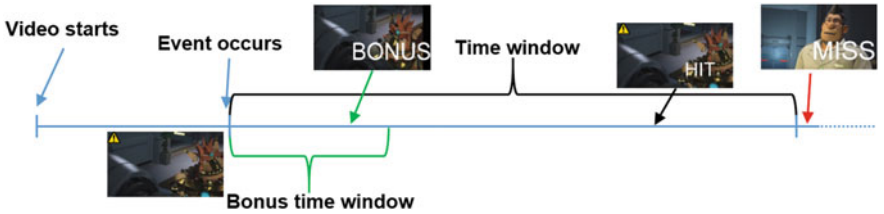


Fig. 21.10 Temporal sequence of the game events

Fig. 21.11 **a** A game event occurs and is indicated in the upper left corner. **b** Provide feedback if the player has not passed the event but did miss the bonus time window. **c** Indicate that both players clicked within the bonus time window. **d** The player missed an event or clicked even if no event has occurred

for the bonus score decreases and introduces a certain level of difficulty as it would be the case in online quiz shows. In the case that the multimedia playback of both players is not synchronous and, therefore, the occurrence of the events differs in time, the bonus time window shrinks according to the amount of introduced asynchronism (total time window—asynchronism). If a player clicks too early or too late (depending on the player and the sign of the asynchronism) but still in the time window of the event, the player is awarded 100 points. If both players manage to click during the bonus window, they are awarded double score. The game provides visual feedback for achieving bonus score (cf. Fig. 21.11c), achieve score (cf. Fig. 21.11b), and for no reaction during the time window (cf. Fig. 21.11d). If players miss (reaction too soon or too late) or react even if no event has occurred, penalty scores are given.

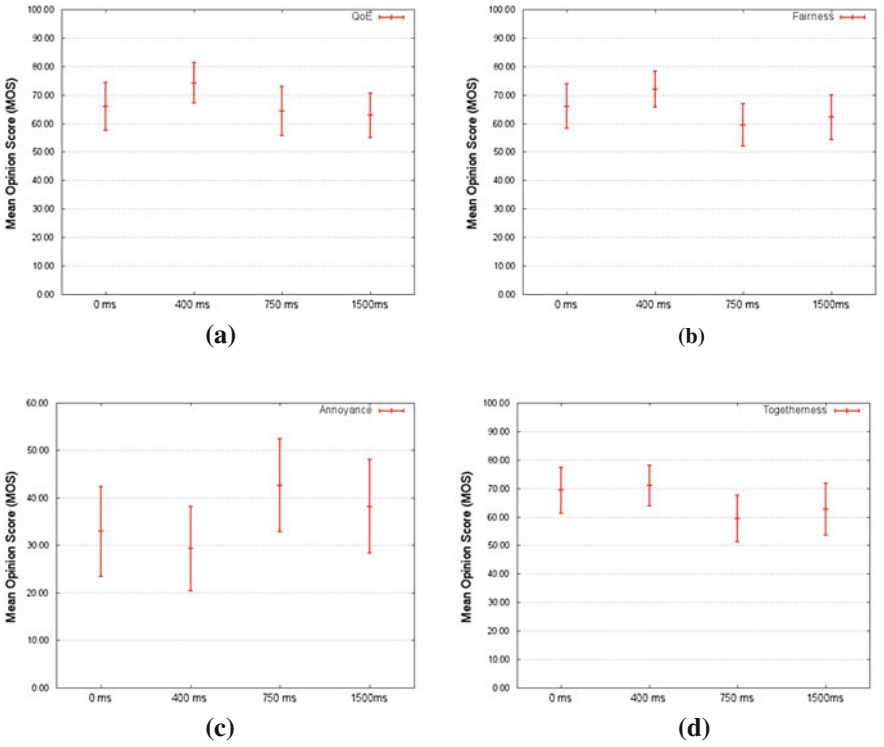


Fig. 21.12 **a** MOS and 95% CI for the overall QoE (higher is better). **b** MOS and 95% CI for fairness (higher is better). **c** MOS and 95% CI for annoyance (lower is better). **d** MOS and 95% CI for togetherness (higher is better)

Since pairing participants online may introduce unpleasant waiting times, we decided to introduce an AI as the second player. In order to provide an AI that mimics the behavior of a real player, we invited students to take part in an in-lab SQA where they played together in pairs. With the gathered statistics from this SQA, we simulate the other player in the SQA using crowd sourcing. Since we have gathered real user data, we were able to use a simple algorithm for the simulated player. The underlying assumption was that the opposite player always clicks if an event occurs. We parsed the reaction times of the successful clicks and built a knowledge base of reaction times. At every event, one of these reaction times is picked at random (uniformly) and acts as second input for the original two-player game. In order to assess the impact of asynchronism on *QoE*, *fairness*, *togetherness*, and *annoyance*, we selected a single stimulus with hidden reference as recommended by the ITU [30, 32]. We further used Microworkers [33] to hire participants for the SQA from the USA, Canada, Australia, New Zealand, West Europe, and East Europe. The duration of the whole SQA is about 15 min. For each successful participation, we pay $0.50.

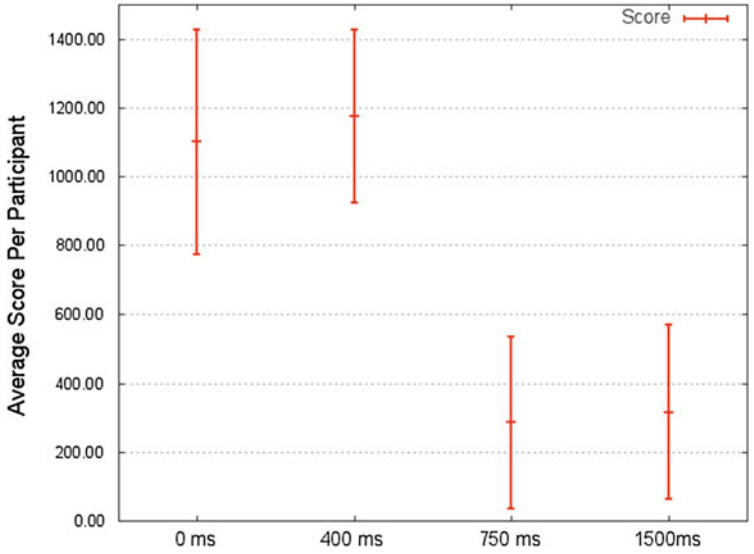


Fig. 21.13 Average score achieved by a participant with 95% confidence interval

Table 21.1 Videos used for evaluation

Video name	Length (mm:ss)	#events
Training	00:54	3
Knack	01:50	4
Famson 1	01:46	6
Famson 2	01:58	8

In order to gather appropriate multimedia content that allows for assessing IDMS using a reaction game, the multimedia content should provide enough possibilities to introduce game events. Therefore, we recorded in-game scenes of video games. We selected non-violent in-game scenes from the two games *inFAMOUS Second Son* [34] (henceforth denoted by Famson) and Knack [35] (henceforth denoted by Knack). Table 21.1 depicts the length and the number of events for the selected video sequences. All the video sequences comprise audio. The SQA is split into the following five parts: **Introduction**. In the beginning, an introduction is presented to each participant which explains in detail the experiment and the actual task. In particular the introduction explicitly states that the audio devices should be turned on and the audio volume should be adjusted accordingly. We further instruct the participants to switch off their mobile devices. The reaction game and the rating possibility are explained in detail such that no questions are left open, including figures that show how events are indicated. At the end of the introduction and before the actual SQA starts, each participant has to agree to a disclaimer.

Pre-Questionnaire. After agreeing to the disclaimer, the participants have to fill out a pre-questionnaire. The pre-questionnaire is used to gather demographic information about the participants, i.e., age, gender, nationality, and country of residence. The data is used to cross-check the preferred regions from which participants are hired for the SQA using crowd sourcing. During the pre-questionnaire, the video sequence is cached. Only if the video sequences are cached successfully, the participants are allowed to pass on to the training and main evaluation.

Training. In order to provide the participants the possibility to become familiar with the reaction game, we introduce a training phase to the SQA. For the training phase, we use an in-game scene from Famson with a duration of approximately 54 s (cf. Table 21.1). We further use the opportunity of a training phase to present the voting possibilities to the participants which are used to assess the following variables: overall QoE, fairness, togetherness, and annoyance. The voting is done by adjusting a slider on a numerical scale from 0 to 100 for each of the variables, where 0 indicates a very low QoE, togetherness, fairness, or annoyance and 100 indicates a very high QoE, togetherness, fairness, or annoyance. We do not restrict the duration of the voting phase. We asks the participants for the same variables as in [1].

Main Evaluation. As already mentioned, the main evaluation adopts a single stimulus with hidden reference as recommended by the ITU [30, 32]. We selected a single stimulus with hidden reference because the participants shall not know which one is the reference condition during the SQA. This allows to investigate whether there is a significant difference between the reference (in our case both players are synchronous) and the cases where we introduce asynchronism. For the main evaluation, we present the video sequences randomly (uniform) assigned to the test cases and each participant has to play the reaction game using the test cases depicted in Table 21.2. The main evaluation comprises only test cases one to four. The first test case (cf. Table 21.2 case 1) depicts the hidden reference. The second test case (cf. Table 21.2 case 2) introduces an asynchronism of 400 ms which results in a bonus time window for each event of 1600 ms. The third test case (cf. Table 21.2 case 3) introduces an asynchronism of 750 ms which corresponds to a bonus window of 1250 ms. The fourth test case (cf. Table 21.2 case 4) introduces an asynchronism of 1500 ms which corresponds to a bonus window of 500 ms. We selected these test

Table 21.2 Test cases for the crowd-sourced evaluation

Case	Description	Asynchronism (ms)	Window length (ms)	Bonus window length (ms)
0	Training video	0	2000	2000
1	Synchronous	0	2000	2000
2	Small asynchronism	400	2000	1600
3	Medium asynchronism	750	2000	1250
4	Big asynchronism	1500	2000	500

cases in order to see whether the assumptions deduced in [1] hold and to assess if the threshold lies below one second. The test cases and the assigned video sequences are randomly presented to the participants during the main evaluation. After each stimulus presentation, the participants are asked to vote the overall QoE, togetherness, fairness, and annoyance as discussed before.

Post-Questionnaire. Finally, in the end of the SQA, the participants are asked to fill out a post-questionnaire. This provides the participants the possibility of providing general feedback. We further ask if the participants have already participated in a similar SQA.

For conducting the SQA using crowd souring, we use a Web-based quality assessment platform provided by [36]. Besides the possibility of selecting the assessment methodology, the platform allows to easily extend the stimulus presentation. In addition to the stimulus presentation time, we measure some more variable regarding the reaction game in order to investigate the behavior of every participant. This is done to identify participants that do not *honestly* take part in the SQA, to identify participants that did not understand the actual task, or to identify participants that try to cheat by reducing the stimulus presentation time [37]. Therefore, we measure the following variables regarding the reaction game: reaction time (time between the occurrence of an event and the first click after it occurred), number of stalls or pauses of the multimedia playback, number of browser window focus changes, the audio volume, the number of clicks during time window, and the total amount of clicks during a video sequence. By the use of these statistics, we filter the participants. How many participants are filtered is given in Sect. 21.3.2.

21.3.2 Statistical Analysis of the Results

In total 89 persons participated in the SQA. After the screening, 44 participants were accepted for the final statistical analysis. The participants have been screened accordingly to the following rules: (i) **Browser focus change**, 27 participants were screened because they changed the Browser focus during the stimulus; presentation. (ii) **Total number of clicks**, 16 participants were screened because they did not click a single time during a stimulus; presentation. (iii) **Number of clicks during an event**, two participants were screened because they never clicked during an event. Figure 21.12 depicts the results for the overall QoE (cf. Fig. 21.12a), fairness (cf. Fig. 21.12b), annoyance (cf. Fig. 21.12c), and togetherness (cf. Fig. 21.12d) for each test case (cf. Table 21.2). For all the test cases, we assume that the samples follow a normal distribution according to Shapiro–Wilk tests [38] and by investigating Q-Q plots. Before conducting a Student's t-test, we verified homogeneity of the variance by conducting F-tests.

Figure 21.12a depicts the MOS for the overall QoE for an asynchronism of 0, 400, 750, and 1500 ms. The test pairs (400, 750 ms)[3] and (400, 1500 ms)[4] show a significant difference. This indicates that the threshold for asynchronism lies in between 400 and 750 ms.

Figure 21.12b depicts the MOS for fairness. The participants had to vote how fair (in their opinion) each test case was with respect to the reaction game. Again the pairs (400, 750 ms)[5] and (400, 1500 ms)[6] showed a significant difference.

Figure 21.12c depicts the MOS for annoyance. The annoyance states how annoyed the participants were after the test cases. As the figure indicates, there is a clear increasing tendency in annoyance of the subjects with an increase in asynchronism. For the test case (400, 750 ms)[7], we found a statistical significant difference in annoyance.

The last variable we measured is the togetherness. Figure 21.12d depicts the MOS for togetherness. The results for this variable follow the same principle as the other three. A Student's t-test revealed the following significant difference between the means of the following test cases: (0, 750 ms)[8] and (400, 750 ms).[9]

Finally, we report the average score a participant has achieved for each of the test cases. Figure 21.13 depicts the average score with 95% confidence interval. Again, the same tendencies can be observed. For the test cases below an asynchronism of 750 ms, the participants are able to achieve high scores. If the asynchronism increases to 750 ms, the scores suddenly drop below 400 points on average.

21.4 Conclusion

In this chapter, we introduced IDMS for pull-based streaming by using a Distributed Control scheme to negotiate a reference playback time stamp among the clients participating in an IDMS session. Specifically, we introduced the notion of an IDMS session for pull-based streaming and showed how MPEG-DASH can be adopted to incorporate these IDMS sessions in the MPD. Following this, we described a Distributed Control scheme for negotiating a reference playback time stamp among the clients in an IDMS Session. The proposed algorithm was evaluated with respect to scalability and the time required to synchronize a certain number of peers. The results show that our proposed approach *Merge and Forward* scales very well with the number of clients. Furthermore, the overhead saved may allow the clients to request higher quality streams which can improve the overall QoE of the IDMS sys-

[3] $t = 1.73$ *p-value* $= 0.087$ $\alpha = 0.1$.
[4] $t = 2.1$ *p-value* $= 0.039$ $\alpha = 0.05$.
[5] $t = 2.51$ *p-value* $= 0.014$ $\alpha = 0.05$.
[6] $t = 1.93$ *p-value* $= 0.057$ $\alpha = 0.1$.
[7] $t = -1.31$ *p-value* $= 0.049$ $\alpha = 0.05$.
[8] $t = 1.68$ *p-value* $= 0.096$ $\alpha = 0.1$.
[9] $t = 2.08$ *p-value* $= 0.03988$ $\alpha = 0.05$.

tem. The selection of the average playback time stamp as the reference has the potential drawback that certain peers may be unable to synchronize to it due to a shortage of bandwidth. Therefore, we introduce a re-synchronization method that allows a peer to influence the calculation of the reference playback time stamp, such that all peers are able to synchronize their multimedia playback.

The synchronization of an IDMS system is crucial, because it directly impacts the QoE as skipping and pausing leading to an undesirable degradation. As a result, we adopt adaptive media playout and formulate an optimization problem selecting appropriate parameters, such that the impact of the resulting playback rate adjustment on the QoE is minimized while avoiding asynchronism. We use a distortion metric as the objective function whose validity for modeling the impact of playback rate adjustments on the QoE is supported by the results of a subjective quality assessment.

We further investigated and proposed a methodology on how subjective quality assessments in the field of IDMS can be conducted using crowd sourcing. The proposed methodology using *games for a purpose* was exemplified by investigating the lower asynchronism threshold for IDMS. [1] assume the lower asynchronism threshold for active voice chatters in social TV scenarios to be about one to two seconds. The presented results of the subjective quality assessment clearly show that one second as the lower asynchronism threshold is not enough in more complex scenarios such as collaborative games or online quiz shows. The results show that all the measured variables significantly decrease (QoE, togetherness, fairness) or increase (annoyance) if the asynchronism reaches 750 ms. Thus, we may expect the lower threshold on asynchronism between 400 and 750 ms. The results provide even more insights. By taking a look at the subfigures of Fig. 21.12, it can be observed that there is a strong relationship between fairness, togetherness, annoyance, and the overall QoE. Indeed, the QoE may be modeled by the other three variables.

References

1. Geerts, D., Vaishnavi, I., Mekuria, R., van Deventer, O., Cesar, P.: Are we in Sync?: Synchronization requirements for watching online video together. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 311–314. ACM (2011)
2. Montagud, M., Boronat, F., Stokking, H., Brandenburg, R.: Inter-destination multimedia synchronization: schemes, use cases and standardization. Multimedia Syst. **18**, 459–482 (2012)
3. Mills, D.L.: Internet time synchronization: the network time protocol. IEEE Trans. Commun. **39**, 1482–1493 (1991)
4. IEEE 1588 Precision Time Protocol on Wireless LAN Software and Hardware Prototypes (2005)
5. Ishibashi, Y., Tsuji, A., Tasaka, S.: A group synchronization mechanism for stored media in multicast communications. In: Proceedings of 16th Annual Joint Conference of the Computer and Communications Societies. Driving the Information Revolution (INFOCOM)., vol. 2, pp. 692–700. IEEE (1997). https://doi.org/10.1109/INFCOM.1997.644522
6. Stokking, H., Van Deventer, M., Niamut, O., Walraven, F., Mekuria, R.: IPTV Inter-destination synchronization: a network-based approach. In: Proceedings of the 14th Intelligence in Next Generation Networks (ICIN), pp. 1–6. IEEE (2010)

7. Mauve, M., Vogel, J., Hilt, V., Effelsberg, W.: Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications (2002)
8. Boronat Seguí, F., Guerri Cebollada, J., Lloret Mauri, J.: An RTP/RTCP based approach for multimedia group and inter-stream synchronization. Multimedia Tools Appl. **40**, 285–319 (2008)
9. Montagud, M.: Design, development and evaluation of an adaptive and standardized RTP/RTCP-based IDMS solution. In: Proceedings of the 21st International Conference on Multimedia, pp. 1071–1074. ACM (2013)
10. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RFC 3550: RTP: A transport protocol for real-time applications. Tech. Rep., IETF (2003). www.ietf.org/rfc/rfc3550.txt
11. Brandenburg, R., Stokking, H., Deventer, O., Boronat, F., Montagud, M., Gross, K.: RFC 7272: Inter-destination media synchronization (IDMS) using the RTP control protocol (RTCP). Tech. Rep., IETF (2014). www.ietf.org/rfc/rfc7272.txt
12. Hesselman, C., Abbadessa, D., Van Der Beek, W., Gorgen, D., Shepherd, K., Smit, S., Gulbahar, M., Vaishnavi, I., Zoric, J., Lowet, D., De Groote, R., O'Connell, J., Friedrich, O.: Sharing enriched multimedia experiences across heterogeneous network infrastructures. IEEE Commun. Mag. **48**(6), 54–65 (2010)
13. Montagud, M., Boronat, F., Stokking, H.: Design and simulation of a distributed control scheme for inter-destination media synchronization. In: Proceedings of 27th International Conference on Advanced Information Networking and Applications, pp. 937–944. IEEE (2013)
14. Ishibashi, Y., Tasaka, S.: A Distributed Control Scheme for Causality and Media Synchronization in Networked Multimedia Games. In: Proceedings of the 11th International Conference on Computer Communications and Networks, pp. 144–149. IEEE (2002). https://doi.org/10.1145/963900.963904
15. Sodagar, I.: The MPEG-DASH standard for multimedia streaming over the internet. IEEE MultiMedia **18**(4), 62–67 (2011). https://doi.org/10.1109/MMUL.2011.71
16. Boronat, F., Montagud, M., Vidal, V.: Master selection policies for inter-destination multimedia synchronization in distributed applications. In: 19th International Symposium on Modeling Analysis and Simulation of Computer and Telecommunication Systems, pp. 269–277. IEEE (2011). https://doi.org/10.1109/MASCOTS.2011.43
17. Hossfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., Schatz, R.: Quantification of YouTube QoE via Crowdsourcing. In: International Symposium on Multimedia, pp. 494–499. IEEE (2011)
18. Montagud, M., Boronat, F.: On the use of adaptive media playout for inter-destination synchronization. IEEE Commun. Lett. **15**(8), 863–865 (2011)
19. Yuang, M., T.Liang, S., Chen, Y., Shen, C.: Dynamic video playout smoothing method for multimedia applications. In: Proceedings of the International Conference on Converging Technologies for Tomorrow's Applications, vol. 3, pp. 1365–1369. IEEE (1996). https://doi.org/10.1109/ICC.1996.533632
20. Rainer, B., Timmerer, C.: Self-organized inter-destination multimedia synchronization for adaptive media streaming. In: Proceedings of the 22st ACM International Conference on Multimedia. ACM, New York, NY, USA (2014)
21. Network Working Group: RFC 5389 - Session Traversal Utilities for NAT (STUN). Tech. rep., IETF (2008) http://tools.ietf.org/html/rfc5389
22. RFC 5766–Traversal Using Relays around NAT (TURN). Technical Report, IETF (2010)
23. Kuramochi, K., Kawamura, T., Sugahara, K.: NAT traversal for pure P2P e-learning system. In: Proceedings of the 3rd International Conference on Internet and Web Applications and Services, pp. 358–363. IARIA (2008). https://doi.org/10.1109/ICIW.2008.74
24. Christensen, K., Roginsky, A., Jimeno, M.: A new analysis of the false positive rate of a bloom filter. Informat. Process. Lett. Elsevier North-Holland **110**(21), 944–949 (2010)
25. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
26. Erdos, P., Renyi, A.: On Random Graphs. Publicationes Mathematicae (Debrecen) **6**, 290–297 (1959). http://www.renyi.hu/erdos/Erdos.html#1959-11

27. OMNET++ 4.3.1: http://www.omnetpp.org/ (Last accessed Dec 2013)
28. Rainer, B., Timmerer, C.: A quality of experience model for adaptive media playout. In: Sixth International Workshop on Quality of Multimedia Experience, pp. 177–182. IEEE (2014)
29. Microworkers: http://www.microworkers.com
30. Rec. ITU-R BT.500-11. Technical report https://www.itu.int/dms/pubrec/itu-r/rec/bt/R-REC-BT.500-11-200206-S!!PDF-E.pdf
31. Verhelst, W., Roelands, M.: An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 554–557. IEEE (1993). https://doi.org/10.1109/ICASSP.1993.319366
32. ITU-T Recommendation P.910: Subjective video quality assessment methods for multimedia applications. Tech. Rep., International Telecommunication Union, Geneva, Switzerland (2008)
33. Hirth, M., Hossfeld, T., Tran-Gia, P.: Anatomy of a Crowdsourcing Platform–Using the Example of Microworkers.com. In: Proceedings of the 5th Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 322–329. IEEE (2011)
34. inFAMOUS Second Son–Sukker Punch: http://infamous-second-son.com/
35. Knack–SCE Japan Studio: http://us.playstation.com/ps4/games/knack-ps4.html
36. Rainer, B., Waltl, M., Timmerer, C.: A web based subjective evaluation platform. In: Proceedings of the 5th International Workshop on Quality of Multimedia Experience, pp. 24–25. IEEE, Los Alamitos, CA, USA (2013). http://www.qomex2013.org
37. Hossfeld, T., Keimel, C., Hirth, M., Gardlo, B., Habigt, J., Diepold, K., Tran-Gia, P.: Best practices for QoE crowdtesting: QoE assessment with crowdsourcing. IEEE Trans. Multimedia **PP**(99), 1–1 (2013). https://doi.org/10.1109/TMM.2013.2291663
38. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality. Biometrika **52**(3/4), 591–611 (1965)

Chapter 22
Watermarking and Fingerprinting

Rolf Bardeli

Abstract An important task in media synchronisation is to find out the playback position of a running media stream. Only based on this information, it is possible to provide additional information or additional streams synchronised to that running stream. This chapter gives an overview of two techniques for solving this basic task: watermarking and fingerprinting. In the former, synchronisation information is embedded imperceptibly in the media stream. In the latter, an excerpt of the media stream is identified based on an index of compact representations of known media content. In both cases, there are specific approaches for audio, image, and video signals. In all cases, the robustness of the methods may be increased by using error correcting codes.

Keywords Fingerprinting · Watermarking · Error correcting codes · Second screen · Content-based indexing

22.1 Introduction

A basic task in media synchronisation is to identify multimedia content and find out the current playback position. This is necessary to make the connection to other material related to the given playback position. In this chapter, we explain two widely used techniques for deducing this information. The first technique, watermarking, aims at embedding information about the identity and time position of a multimedia document into that document without interfering with its contents. The second technique, fingerprinting, leaves the original material untouched but creates a compact representation of the primary material that allows fast and robust identification.

Figure 22.1 gives an overview of how these two techniques are applied in media synchronisation. Both techniques allow finding both the identity and the playback position of the media stream a user is currently watching. This information can be used to present secondary content that is tailored to a specific media context.

R. Bardeli (✉)
Vodafone GmbH, Ferdinand-Braun-Platz 1, 40549 Düsseldorf, Germany
e-mail: rolf.bardeli1@vodafone.com

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_22

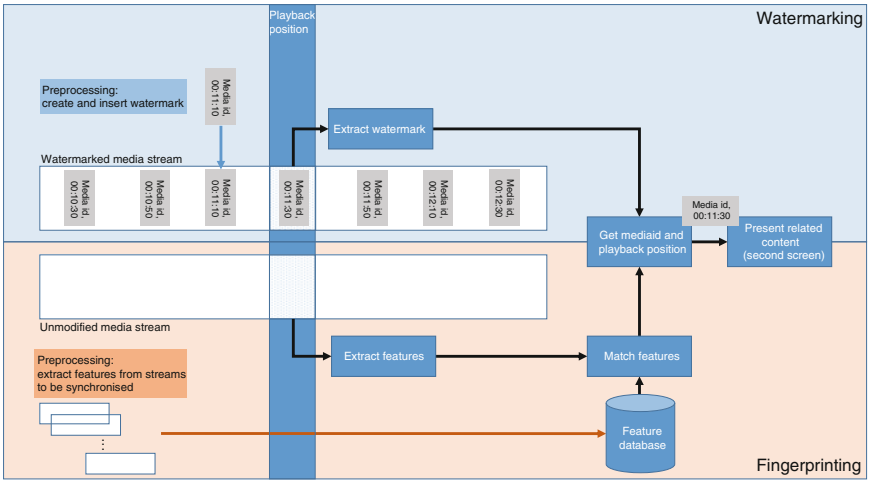


Fig. 22.1 The role of watermarking and fingerprinting in media synchronisation. In watermarking, information about the identity and playback position of a media stream is embedded in the stream in a way that it can be extracted again. Fingerprinting leaves the media stream untouched and uses a database of features from streams to be synchronised in order to find identity and playback position based on a feature matching process. In both cases, the identity of the media stream and its playback position are used to present secondary content tailored to the current media context

The two techniques differ, however, in the way they achieve to obtain this information. By watermarking, it is embedded directly into the media stream and can be inferred without an external source of information. Fingerprinting leaves the media stream unmodified. Rather, a database of features and their positions in the media streams to be synchronised is prepared and at playback time, the identification of media and playback position is obtained as the result of a feature matching process.

As both techniques can be used for synchronisation, the question is which method to use for which application. For some users in the broadcasting industry, the answer is clear: do not tamper with the broadcasting signal. In such a case, using watermarking is not applicable and therefore one has to resort to fingerprinting. In general, watermarking is the more local technique in the sense that the signal to be synchronised is all that is needed to find basic synchronisation information. For fingerprinting, there is always a lot of additional infrastructure that is necessary to maintain and update fingerprint databases. Moreover, when the amount of material the user could synchronise with is large or when the number of users is high, the use of watermarking makes it easier to use the computing devices of end-users to allow scaling with the number of users.

In general, both watermarking and fingerprinting have a wide field of applications which lead to possibly quite different requirements on the algorithms. Classical applications, in addition to synchronisation, include copy protection, digital rights management, and duplicate detection.

Watermarking and fingerprinting approaches both need to be robust. Which kind of robustness is needed, however, depends very much on the application. Where watermarks need to be highly robust to tampering in copy protection applications, this plays almost no role for media synchronisation.

In the next sections, we will explain techniques for both watermarking (Sect. 22.2) and fingerprinting (Sect. 22.3). We will concentrate on those approaches which are most suitable for media synchronisation and discuss the specific requirements and how they are met as we go along.

Finally, Sect. 22.4 will give an overview of the role of error correcting codes in watermarking and in fingerprinting.

22.2 Watermarking

Fewer and fewer people today encounter paper with a watermark. Therefore, we have to allude to banknotes in order to have a picture of what a watermark means classically. Now, fetch a bank note and have a look at it; first against a background and then against the light. You will notice (depending on the note and currency) a translucent image appearing when checked against the light. This can be interpreted as extra information embedded in the banknote which is not noticed during typical use.

For multimedia content, the goal of watermarking is similar to that of watermarking paper: embed information into a multimedia document such that this information can be easily extracted by the right means. In particular, when this information describes the identity of the content and a timestamp, it gives all the information that is necessary for media synchronisation.

When watermarking is applied for media synchronisation, there are four central requirements:

1. **Robustness**: The information embedded in a signal must remain intact when typical distortions such as cropping, noise, or lossy compression occur.
2. **Imperceptibility**: Users should not notice the difference between the original signal and watermarked versions.
3. **Extractability**: The information embedded in a signal must be easy to extract.
4. **Data sufficiency**: The watermark must provide a data rate that is high enough to enable embedding synchronisation information such as unique identifiers and timestamps.

The basis for watermarking is psychoacoustic and psychovisual effects which allow modifying parts of the primary material such that the modified (watermarked) version contains additional information that is either inaudible (in the case of audio) or invisible (in the case of images and videos) to the human ear or eye. Therefore, they are specific to these different modalities and we will discuss techniques per modality.

22.2.1 Audio Watermarking

In many domains of audio signal processing, a first categorisation of methods can be based on whether they are working with the time-domain signal or with its time–frequency representation given by the windowed Fourier transform or similar transforms. This is also true for audio watermarking. In this section, we roughly follow the presentation in the more extensive survey [19]. However, we will leave out the topic of attacks and countermeasures—the questions of how adversaries might try to remove a watermark or make it unrecoverable and how to make this hard or impossible—because they relate mainly to the use-case of watermarking for digital rights management. The only form of attack to a watermarking scheme that is relevant in the context of media synchronisation is given by what may be called unintentional (yet unavoidable) attacks like the introduction of noise or low bit-rate lossy compression. Moreover, we concentrate on a specific time-domain approach which will allow us to introduce most of the central aspects in audio watermarking.

One important aspect of watermarking for incorporating information into an audio signal is that such systems should allow blind watermark detection. This means that it must be possible to extract the watermarks without knowing the exact watermarking signal. This is in contrast to digital rights management applications where it is only import to robustly decide whether or not a known watermark is embedded in a given signal.

As an example for a time-domain approach along these lines, we will discuss the algorithm proposed in [3]. It will also allow illustrating the use of a psychoacoustic model to achieve inaudibility of the watermark.

We start with a discrete-time audio signal $x(n)$ where n denotes the discrete-time variable. Assume that the information to be embedded is given by a sequence of L characters $c_0, c_1, \ldots, c_{L-1}$ from a fixed-size alphabet $\{0, 1, \ldots, M-1\}$. The first step is to translate this character sequence into a watermark signal. The basis for this step is the choice of M audio signals $S = (s_0, s_1, \ldots, s_{M-1})$ of a fixed length N representing the M available characters. This is known as the embedding codebook. The watermark signal is then simply the concatenation v of the signals s_{c_ℓ} for the characters c_ℓ given by

$$v(n) = \sum_{\ell=0}^{L-1} s_{c_\ell}(n - \ell N).$$

This watermark v is embedded into the audio signal x to obtain the watermarked signal y as follows:

$$y(n) = x(n) + \alpha \cdot h(n) * v(n) \tag{22.1}$$

Here, $h(n)$ is a shaping filter determining in which frequency bands the watermark signal should be embedded, α determines the embedding strength, and $*$ denotes convolution. The choice of h and α is based on a psychoacoustic model and needs to be made in a way that ensures that the watermark has both enough power that it can be extracted and that it is inaudible.

Psychoacoustic models [13] are based on the fact that in certain conditions, sounds are not perceived even though they are clearly present. For example, in the presence of a loud tone, a tone in a related frequency band cannot be heard unless its power is greater than a threshold depending on the first tone. This effect is called masking. Such effects do not only apply to simultaneous tones but also to ones shortly after the first one (forward masking) or even shortly before it (backward masking). The task of a psychoacoustic model is to estimate the so-called masking threshold which is both time and frequency dependent. This means that at each time and for each frequency band, the model gives a power threshold below which a tone at that time and in that frequency band is inaudible. Watermark information can be embedded in a given time–frequency bin with a power that is below the threshold given by the psychoacoustic model.

The shaping filter h in Eq. 22.1 is designed adaptively such that it distributes the watermarking signal v over the whole frequency range in a way that ensures that the embedded watermark is of sufficient power to be detectable yet in each frequency band is of low enough power to be below the masking threshold and therefore inaudible.

The data rate R that can be achieved by this approach is given by $R = \log_2(M)\frac{F_x}{N}$ when the sample rate of the original signal x is F_x. If a typical sample rate $F_x = 44.1$ kHz is assumed, as well as an alphabet size of $M = 4$ and a watermark signal length $N = 441$, this results in a data rate of 200 bits per second.

The watermark can be extracted from a disturbed watermarked signal using a maximum likelihood approach based on a discrete-time random process model for the original signal x (see [3] for details).

Transform domain methods, as their name suggests, work by embedding a watermark into a transformed version of the signal like its windowed Fourier transformation. This may allow a better control of inaudibility by avoiding to embed the watermark in parts of the signal (e.g. parts of its spectrum) which are either silent or contain rich sound [21].

22.2.2 Image Watermarking

Image watermarking is usually performed in a transform domain. In this section, we showcase two of the most popular transforms in this context: the discrete cosine transform (DCT) and the discrete wavelet transform (DWT). The former is well-known from early image compression algorithms and is therefore an immediate candidate for other image processing tasks as well. The latter has many intrinsic advantages such as its space–frequency localisation, its inbuilt multi-resolution representation, and its low computational complexity. It also makes it comparatively easy to design algorithms adapted to the human visual system which is important where invisible watermarks are concerned.

The discrete cosine transform expresses an image as a weighted sum of cosine functions of different frequencies. This implies a frequency interpretation of the

coefficients of the transform which is exploited in image compression where fewer bits are reserved to encode small high-frequency coefficients. Similarly, the distribution of watermark energy over different DCT coefficients can be designed in this fashion.

Often, watermarks are encrypted before embedding them in an image. This is important in digital rights management applications where only the watermark owner is supposed to be able to reconstruct it. For synchronisation purposes, this is not necessary.

The colour channels for red, green, and blue in RGB images are highly correlated. This can be clearly seen when comparing the individual channels as greyscale images. Therefore, watermarking is either performed on greyscale images or in a different colour space such as the YUV colour space. This is also similar to the case of image compression.

The DCT is usually applied to 8×8-pixel blocks of an image (again as in JPEG compression). After that, there are two aspects of deciding how to embed the watermark. First, the blocks in which the watermark is embedded are selected. This is done such that the invisibility of the watermark is optimised. This means that the most complex blocks of an image are selected. Then, on a similar basis, the DCT coefficients in each block are selected in which the watermark information will be embedded. A simple strategy for these choices is to select blocks in the descending order of the number of nonzero DCT coefficients in the block. Then, the watermark is embedded in the low-frequency coefficients of the selected blocks [25]. The number of blocks to be selected depends on the number of watermark bits embedded in each block and on the length of the watermark. To further reduce visibility, the luminance (Y channel in the YUV colour space) may be adjusted towards the original luminance. A quality factor can be introduced to govern how much weight is put on invisibility as compared to easy extraction of the watermark.

In the embedding process, DCT coefficients can be modified such that their modulus with respect to a fixed divisor is more than half the divisor if the watermark bit 1 is embedded and less than half the divisor otherwise. Hence, the watermark can be extracted by first computing the block-wise DCT of the luminance channel of an image. Then, the blocks in which the watermark is embedded are found in the same way as during watermark embedding. Finally, the watermark bits can be found by computing the remainders with respect to the fixed divisor and thresholding the result.

The discrete wavelet transform can be understood as a cascading sequence of low-pass and high-pass filters in which the low-pass signal is understood as a coarser level of resolution. The filter outputs are down-sampled and in turn analysed by low-pass and high-pass filters. Wavelet coefficients at the different levels of the transform correspond to different scales and detail levels.

In the DWT of an image, energy is not distributed uniformly in those coefficients corresponding to the high-pass filter outputs. Image information from edges and texture is concentrated in a few coefficients. Hence, in DWT-based watermarking approaches [27], a similar strategy as in audio watermarking can be pursued by distributing the watermark power adaptively depending on the size of the DWT

coefficients. This leads to an implicit exploitation of psychovisual effects because it leads to placing more watermark energy in image regions to which human vision is less sensitive. It is however important to keep in mind that those parts of an image to which the visual system is least sensitive may well be distorted considerably by lossy image compression algorithms exploiting exactly this fact.

22.2.3 Video Watermarking

A straightforward way of implementing video watermarking is to apply the techniques of the previous sections to the images of a video, to its audio tracks, or both. However, videos are usually stored and transmitted in compressed form. In particular, only some of the frames are stored as compressed images (I-frames). The rest are stored in differential form, being predicted from previous (P-frames) and/or following frames (B-frames) based on block motion. It is therefore adequate to devise special techniques adapted to this situation.

Most video watermarking approaches target I-frames and are quite similar to image watermarking approaches. However, I-frames may be relatively sparse within a video stream and hence, their applicability for synchronisation purposes is limited.

P-frames in videos are highly compressed and therefore do not carry enough bandwidth to allow applying psychovisual masking techniques which would result in selecting a lower number of DCT coefficients in which to embed the watermark. Moreover, changing the DCT coefficients corresponding to frequency zero (dc coefficients) leads to visible effects and is therefore avoided. Consequently, the watermark is distributed over all the DCT coefficients except for the dc coefficients.

An approach that is based fully on spatio-temporal information is given by using the three-dimensional Fourier transform of a sequence of video frames to embed a watermark [12]. Here, videos are divided into consecutive chunks of a fixed number of frames resulting in a three-dimensional signal with two spatial and one temporal dimension. The three-dimensional Fourier transform has similar favourable invariance properties as the lower dimensional versions with respect to linearity, scaling and translation in all three variables, and rotation in the spatial variables. Also in this approach, the watermark is encoded for two reasons. First, for cryptological reasons because this is making the watermark robust to attacks. Second, because of the error correction properties (see Sect. 22.4).

Watermark embedding is similar to embedding in the two-dimensional case. As the Fourier transform is separable—it can be computed one dimension after the other—it makes sense to consider spatial frequencies. The watermark information is then embedded in the magnitudes of Fourier coefficients from the medium frequency band. This leads to a compromise between invisibility and robustness to lossy encoding. Some care has to be taken to modify the Fourier coefficients without destroying the symmetry of the coefficients characteristic for real-valued input signals.

The shift invariance of the Fourier transform makes it possible to ignore synchronisation issues when extracting the watermark if the same watermark is used in

several consecutive chunks. Each bit of the watermark is encoded in a pair of Fourier coefficients using opposite signs. In this way, the watermark can be extracted by computing differences of coefficient pairs.

22.3 Fingerprinting

The idea of fingerprinting is derived from the following property of the human fingerprint: it is a compact unique identifier for a human being. To stretch the analogy a little further, it is often possible to identify a person from only a partial fingerprint.

For multimedia data, fingerprints are designed to be compact bit strings which allow the unique identification of excerpts of documents. The identification should still be possible if the query document is distorted by transformations typical for the application domain. For example, fingerprints should be robust with respect to lossy encoding, noise, and cropping. For synchronisation purposes, it is important to use fingerprinting techniques which not only allow identifying a partial document but also find its position within the complete document.

Recognition of media based on fingerprinting is often reduced to point cloud matching. This is because fingerprints are often derived from points of interest, e.g. in a spectrogram or in an image. To get an idea of the matching task behind many fingerprinting approaches, we make use of the following analogy. Let us try finding a specific star constellation in the sky. Figure 22.2 shows some of the stars of the constellation of the Plough. It also shows stars from the star field surrounding this constellation. Try finding the constellation within the star field before looking at the solution in Fig. 22.3. This is more or less the same problem that has to be solved to match a fingerprint to all the fingerprints in a database. Several approaches to fingerprinting make use of constellations of interest points to speed up the matching process. This is illustrated by adding constellation lines as shown in Fig. 22.3.

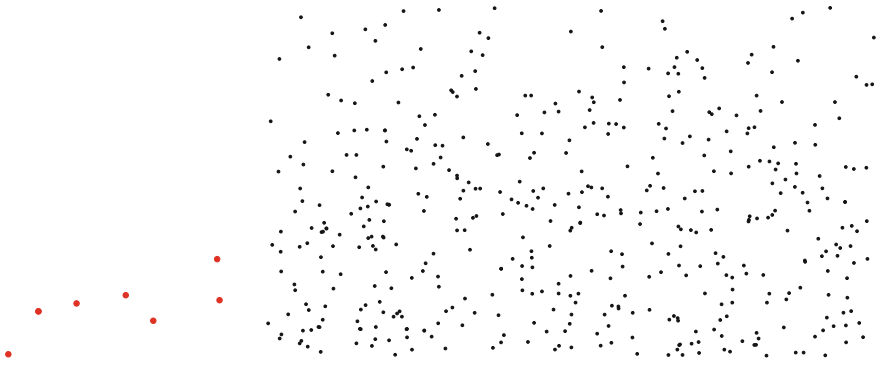


Fig. 22.2 Finding a constellation in the sky. Try finding the stars of the Plough given as a query (in red) within the star field on the right-hand side (black) before looking at the solution in Fig. 22.3. This task gives an analogy to fingerprint matching

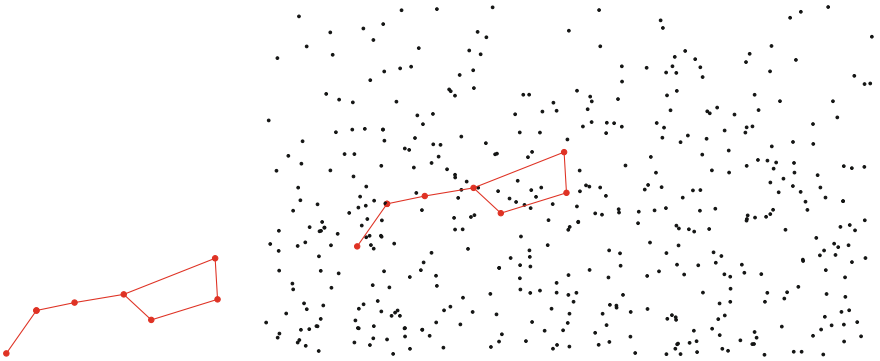


Fig. 22.3 Solution to the matching task from Fig. 22.2. Adding constellation lines can greatly speed up the matching task

In this section, we will first give an introduction to fingerprinting for audio, images, and videos (Sects. 22.3.1–22.3.3). Then, we will discuss how audio fingerprinting has been adapted to music fingerprinting in an application synchronising live piano performances to sheet music (Sect. 22.3.4). We will close this section with a look at different requirements for efficiency in fingerprinting applications for media synchronisation (Sect. 22.3.5).

22.3.1 Audio Fingerprinting

Audio fingerprinting is by now a well-researched topic and a fully commercialised tool that has been used for well over a decade. With the advent of a number of, by now, classical techniques, the problem of efficient audio identification based on a recording of an excerpt of the same piece of audio was regarded as solved. An early review of such techniques can be found in [8].

In the AudioID system [1], clustering is at the centre of its indexing and search algorithms. Feature vectors calculated from each audio signal are clustered, and the results are employed to represent the audio file as a codebook based on the cluster centres. Thus, a search index is given by a sequence of codebooks. Queries are performed by finding the codebook in the index which produces the smallest error when approximating the query features.

The audio hashing approach [16] is based on mapping spectral feature vectors to 32-bit feature vectors. In the indexing process, occurrences of feature vectors are stored in a hash table. Queries are resolved by using this hash table to retrieve all occurrences of consecutive feature vectors in the database signals. For each such occurrence, a post-processing step is required to compare successive 32-bit vectors of the query and the candidate match, to compute a final match probability.

In contrast to these approaches, Cano et al. [7] explicitly design their algorithm with robustness towards timescale changes in mind. Pre-trained hidden Markov models (HMMs) are used to assign labels to the segments of an audio signal. The resulting label sequences are referred to as AudioGenes. A query is then given as a sequence of AudioGenes. It is resolved by a fault tolerant string matching algorithm. In a latter implementation, string matching is replaced by modelling the search index as a single composite HMM and queries are resolved using Viterbi decoding.

A beautiful general approach to multimedia retrieval is used in [9] to derive an audio fingerprinting algorithm. The general approach is based on group theory to model the invariance properties desired for matching. For example, the group of translations along the time axis can be used to model the invariance property needed to identify excerpts of audio. This approach automatically leads to a fast matching algorithm. It has to be tailored to a specific application like audio fingerprinting by specifying the invariance group and by designing a feature extractor which is at least approximately a group homomorphism (in particular, it should commute with the action of the invariance group).

From the classical techniques, the original Shazam audio fingerprinting approach [38] stands out because of its commercial success and wide publicity. It is also easy to understand and illustrates some general principles in audio fingerprinting. We will therefore use it as a showcase to explain audio fingerprinting.

Figure 22.4 gives an overview of this approach. The upper half of the figure illustrates the indexing process. All audio files which are meant to be recognised by the fingerprinting system are processed during indexing. First, fingerprints are extracted, leading to a sequence of hash-values for each file. They are stored in a fingerprint database. In parallel, metadata for each file is stored in a metadata database and of course, these data are connected via unique identifiers. When a query is posed in the form of an audio file, it first undergoes the same process of fingerprint extraction. The resulting query fingerprints are then matched against the fingerprints in the database giving a ranked list of matching audio files including the exact playback position at which the query matches. In the following, we look in more detail at both the fingerprint extraction process and at an efficient matching algorithm.

A central idea, leading to fast matching, is that of creating constellations as fingerprints (Fig. 22.5). This is similar to what has been discussed in the introduction. A first step is to find points of interest in the spectrogram. These points of interest are simply found as local maxima in the spectrogram. More precisely, for each time–frequency point in the spectrogram, this point is marked as a point of interest if the value of the spectrogram at that point is larger than any value in a fixed neighbourhood (e.g. a 5×5 array around the point of interest).

Constellations are then formed using the points of interest. For each such point p at time t_p and frequency f_p, all points of interest in a so-called *target zone* are considered. This zone consists of all time–frequency positions (t,f) with $t \geq t_p + \tau_0$, $t \leq t_p + \tau_1, f_p \geq f + \omega_0$, and $f_p \leq f + \omega_1$. Hence, it has the shape of a box of width $\tau_1 - \tau_0$ and height $\omega_1 - \omega_0$ placed relative to the each point of interest. The values $\tau_0, \tau_1, \omega_0, \omega_1$ are fixed design parameters influencing the number of constellations and thus the size of the fingerprint. Given a point of interest at position (t_1,f_1) and

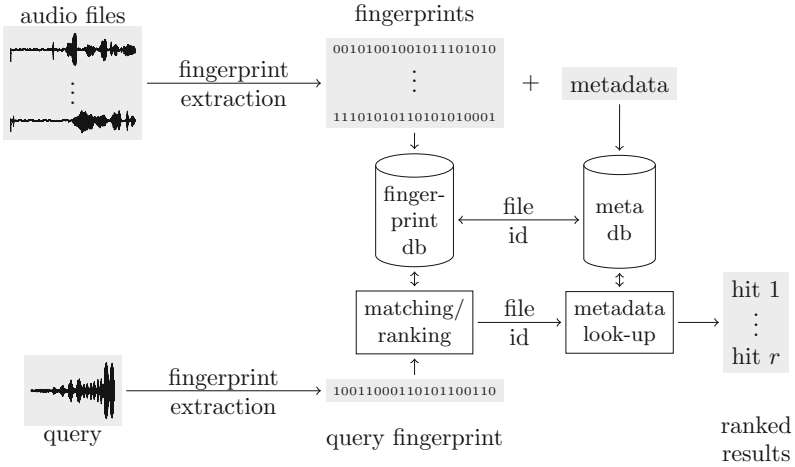


Fig. 22.4 An overview of the Shazam audio fingerprinting approach. The upper half of the figure illustrates the indexing process for audio files that are meant to be recognised. The lower half reflects query processing including efficient matching of fingerprints

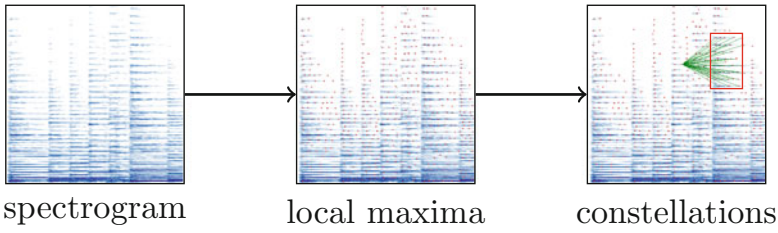


Fig. 22.5 An overview of fingerprint extraction in the Shazam audio fingerprinting approach. Fingerprints are extracted from the spectrogram, a representation of audio given time on the horizontal axis and frequency on the vertical axis. Intensity is colour-coded, and deeper shades of blue indicate stronger contribution of the given frequency at the given time to the audio signal. Local maxima are extracted and marked as points of interest (red dots). Each point of interest is then paired with each point in a so-called *target zone* (red box) positioned relative to the point of interest. This leads to constellations (green lines) to be indexed

a point (t_2, f_2) in its target zone, a fingerprint is created from the values f_1, f_2, and $t_2 - t_1$. This can be encoded as a 32-bit integer. The integer value is called a hash-value and is stored in the fingerprint database. At the same time, side information is stored for this hash-value. It is given by the time t_1 and by the unique identifier of the audio file under consideration. The side information will be used in the matching process to identify both the file in which a match was found and the time position at which the match was found.

Figure 22.6 gives an example of the matching process. Once fingerprints have been extracted from a query, these can be matched to the database as follows. (For illustration, we ignore the fact that there are several files in the database. We can

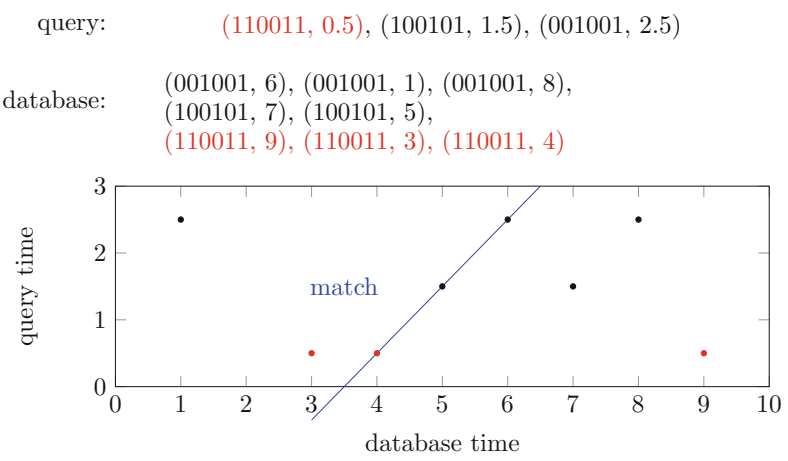


Fig. 22.6 Fingerprint matching in the Shazam audio fingerprinting approach. This example gives three hashes extracted from a query at time 0.5, 1.5, and 2.5 and hashes for only one audio file in the database. These are matched to hashes in the fingerprint database. Whenever a query hash at time t_q matches a hash-value at time t_d in the database, the point (t_d, t_q) is added to a so-called *matching table*. This is illustrated in red for the first query element (and in black for all others). A match can be found as a diagonal in the matching table (blue)

either assume that the database contains only one very long file: the concatenation of all files in the database or that the following process is performed once per file in the database.) Assume that the query contains q elements $(h_1, t_1), \ldots, (h_q, t_q)$, each consisting of a hash-value h_i and a time position t_i. For each such element, all matching hashes (h_i^d, t_i^d) with $h_i^d = h_i$ in the database are found. For each such match, the pair of times (t_i^d, t_i) is stored in a so-called *matching table*. Once the matching table is filled, actual matches can be found as diagonals in the table. A diagonal indicates that the matches in the database show the same time progression as the query elements. Finding the diagonal can be reduced to finding peaks by accumulating histograms of the values $t_i^d - t_i$. Geometrically, this means stacking all points along a diagonal on top of each other.

Audio fingerprinting has many applications. One which has received a lot of attention recently is second screen synchronisation. For example, the fingerprinting approach described in [5] was used to provide audio-based synchronisation of first and second screen devices. A parallelisation based on Amazon Web services was used to scale the retrieval engine to varying numbers of users.

If audio fingerprinting has been seen as a solved problem for several years, then what do recent developments target as open points still meriting further research? Audio fingerprinting seems to be driven by the same technological foci as other areas: these include devising approaches which are particularly suited for mobile devices [20] and speeding up retrieval by making use of the vast parallel processing capacity of GPUs [30].

22.3.2 Image Fingerprinting

An obvious application of finding an excerpt of an image in a large database is in biometrics, in particular in fingerprint matching [14, 15], the godfather of our topic. However, this field is both vast and very far from the topic of synchronisation. Let us therefore turn to modern children's books for motivation.

In recent years, many books are published as augmented reality editions. These come in two versions. One is closer to the watermarking domain: specialised hardware is used to recognise minute patterns printed on a book's pages and can then play back audio related to the position in the book. A second version is a direct application of image fingerprinting: the camera of a mobile device is used to take an image of the book at hand which is in turn recognised. In this case, the material can be enhanced both acoustically and visually. Here, we are in a typical media synchronisation scenario where media is recognised in order to deliver context-based additional content.

We will see in Sect. 22.3.3 that the techniques discussed in this section also apply to the frames of videos.

A simple way to design an image fingerprint is a representation that retains a rough description of the contents of an image. For example, colour histograms in various colour systems provide such representations.

Many applications need image fingerprints that are robust with respect to typical image transformations. A basic idea to incorporate invariance towards such transformations is to apply a transformation with the respective invariance properties to the image.

A good illustration for this approach is the Radon transform which has been applied to image fingerprinting in [34]. For a given image $f(x, y)$, the Radon transform $R[f](s, \theta)$ is given by a line integral along a line at an angle θ to the y-axis and at a distance s from the origin:

$$R[f](s,\theta) = \int_{-\infty}^{\infty} f(s\cos\theta + t\sin\theta, t\sin\theta - s\cos\theta)dt$$

Translation, scaling, and rotation of the image f have the following effects on the transform g:

- **Translation**: The Radon transform of a translated function is given by

$$R[f(\cdot - x_0, \cdot - y_0)](s,\theta) = R[f](s - x_0\cos\theta - y_0\sin\theta, \theta),$$

  showing that the angle variable is unaffected by translation.
- **Scaling**: For a scaled function, we get

$$R[f(\frac{\cdot}{s_x}, \frac{\cdot}{s_y})](s,\theta) = |s_x| R[f](s\frac{s_x}{s_y}, \frac{\theta}{s_y}).$$

In particular, if $s_x = s_y =: \sigma$, then

$$R[f(\frac{\cdot}{\sigma}, \frac{\cdot}{\sigma})](s, \theta) = |\sigma| R[f](s, \frac{\theta}{\sigma}).$$

- **Rotation**: Finally, for a rotated function

$$R[f(\rho_{\theta_0})(\cdot, \cdot)] = R[f](s, \theta_0 - \theta).$$

Here, ρ_{θ_0} denotes rotation around the origin by the angle θ_0. This shows that rotation in the image domain leads to a translation of the angle variable.

These properties can be exploited to obtain a fully invariant transform with respect to these three effects by using the autocorrelation function of the Radon transform (translation invariance), followed by the logarithm mapping and the 2d Fourier transform (scaling and rotation invariance).

Another way to think about such invariance properties is to consider key points in images. The idea is to find a way of extracting such interest points which have a high likelihood of also being extracted from the same image after applying a transform. The most popular approaches along these lines are the scale-invariant feature transform (SIFT, [26]) and its descendants. Key points are extracted as local maxima in a multi-scale representation of the image giving both location and scale information. These are enhanced by features vectors giving local directivity information which is obtained from histograms of local directivity vectors in a neighbourhood of each key point. Rotation invariance is obtained by normalising these histograms with respect to the local directivity at the key point. The key points can either be indexed by themselves or in the form of so-called visual words. The latter are defined as histograms of features computed by counting how often features belonging to a fixed set of clusters obtained by clustering a large set of images occur.

Fast fingerprint matching is facilitated by spatial indexing techniques such as kd-trees and locality sensitive hashing.

A recent trend in signal processing is sparse coding. This can also be applied to image fingerprinting [28]. In this context, image features are represented in terms of an over-complete dictionary D of features which is just a finite set of feature vectors. A feature vector x is then represented by solving the minimisation

$$\min_y \|x - Dy\|_2$$

subject to the condition that as many entries as possible of the representation y are zero, i.e. $\|y\|_0 \leq T$ where T is a threshold on the number of nonzero entries. Fingerprints can then be defined as histograms of representation coefficients (entries of vectors y as given above) for a selected subset of the most representative elements of the dictionary D.

22.3.3 Video Fingerprinting

The techniques discussed in the previous sections can be applied directly also to videos. Audio fingerprinting can be applied to the audio tracks of a video and image fingerprinting can be applied to individual frames. The latter may address either all frames or only keyframes.

In this section, we focus on techniques which are on the one hand suited for synchronisation purposes. This rules out any techniques based on statistics collected over the whole of a video and not allowing to identify the temporal location of a frame or an excerpt within a video. On the other hand, we focus on such approaches taking into account the temporal dimension of videos and thus going beyond frame-wise application of image-based techniques.

The most basic temporal structure of a video is given by its sequence of shots. By shot detection, a video can be segmented into a sequence of relatively homogeneous parts. The lengths of these parts can then be used as a comparatively rough fingerprint [36]. A more content-based approach is to incorporate difference information from consecutive frames [29, 31], block motion vectors such as used in video compression [18], or point of interest tracking [24].

As seen in the previous sections, transform domain methods are particularly popular because of their invariance properties with respect to such effects as translation, rotation, and scaling. For this reason, they are also found in video fingerprinting approaches. In particular, three-dimensional transforms extend over both the spatial and the temporal domain of a video and can therefore cover both aspects in an integrated way. Examples are given by the three-dimensional discrete cosine transform [11] and the three-dimensional random bases transform [10]. Another approach is based on moment invariants extracted from circular regions directly incorporating motion information into the fingerprint [32].

Due to their data size, videos are in most cases stored and transmitted in compressed form. For two reasons, it is very desirable to extract fingerprints directly from the compressed data stream [37]. First, decompression is computationally expensive and hence slows down both indexing and retrieval. Second, a lot of video analysis such as motion estimation is performed in the compression process and can be repurposed for fingerprinting.

22.3.4 Music Fingerprinting

This section shows how fingerprinting is used in a beautiful example of the synthesis of many music information retrieval techniques. Music can be encountered in very different forms: as a live performance, as an audio recording, as sheet music, as written lyrics, and many others. Synchronising such different renderings for the same piece of music can lead to a greatly enhanced experience. Two examples of systems using different synchronisation techniques to achieve this are the SyncPlayer

framework [22] and the Complete Classical Music Companion [2]. Where the former makes use of audio fingerprinting directly to identify a music recording and then present other information which has been synchronised to the score, the latter makes use of an interesting adaptation of the Shazam audio fingerprinting approach to synchronise live piano performances to sheet music.

The general set-up of this latter system is as follows. A live performance of piano music is recorded by a microphone, and the resulting signal is fed into a computer. The signal is first translated into notes using a recurrent neural network [6]. Then, the note sequence is matched to a database of scores of known piano pieces. If the piece was identified successfully, the performance is then tracked through the score and the current playing position can be highlighted in the score. More sophisticated applications (an automatic page turner, for example) may be based on this information.

We will concentrate on the adaptation of the Shazam fingerprinting approach to this scenario. Different than in the scenario discussed in Sect. 22.3.1, fingerprinting is applied to a symbolic representation of music. The goal of this adaptation is to make fingerprinting very robust to tempo variations and other modes of musical expression.

Recall that for audio fingerprinting, a fingerprint can be defined based on two points of interest (t_1, f_1) and (t_2, f_2) resulting in the triple $(f_1, f_2, t_2 - t_1)$ formed from two frequency values and a time difference. This allows invariance with respect to time translation. However, tempo variations lead to significantly different fingerprints. The general idea for music fingerprinting is to replace frequency in the raw signal by note pitches and to combine the time information from three notes into a ratio of time differences. More concretely, if three notes (p_1, t_1), (p_2, t_2), and (p_3, t_3) are given by their pitches p_i and their onset times t_i, a fingerprint is defined by the quadruple $(p_1, p_2, p_3, \frac{t_3 - t_2}{t_2 - t_1})$. An attention scheme similar to the target zone concept is used to select note triples from the score or the note transcript and thus arrive at a set of fingerprints. During the matching process, diagonals of different slopes may occur depending on the actual tempo of the interpretation. This can be handled by storing $t_2 - t_1$ in the side information and trying to estimate the tempo difference between score and interpretation.

22.3.5 Efficiency

In fingerprinting, three aspects of efficiency need to be addressed:

1. Fast matching,
2. Short response times with respect to content size, and
3. Short response times with respect to the number of simultaneous users.

Most fingerprinting algorithms are designed to address the first two aspects. Usually, some variant of a hashing scheme such as the one described in Sect. 22.3.1 is employed to make the matching of query fingerprints to the fingerprint database fast.

In particular, the complexity should increase only logarithmically with the size of the database.

The third aspect, handling many parallel users, is usually addressed by both having short response times on single queries and scaling the solution over many computing nodes. This is very well-suited for parallelisation because in principle, one core can be used per query if the database is properly replicated.

The case of media synchronisation for second screen applications is somewhat special because many very similar parallel queries have to be handled, when millions of viewers want to synchronise to the same programme at its start time.

22.4 Error Correcting Codes

Ever since Shannon's [35] and Hamming's [17] seminal papers, error correcting codes have been a staple of securing communication over channels likely to corrupt a signal. The effects on audio, image, and video signals by noise, lossy encoding, etc., can be understood and handled in this way.

An error correcting code over an alphabet Σ is simply a subset $\mathscr{C}$ of Σ^n. The elements of $\mathscr{C}$ are called code words and are all of the same length n. If $\mathscr{C}$ has 2^m elements, it is straightforward to assign a different code word to any m-bit string. The basic idea behind error correcting codes is that they introduce redundancy into a message making it possible to detect and correct a certain number of errors. The simplest example for this idea is given by repetition codes which simply repeat each bit in a message a fixed number of times. For example, if each bit is repeated three times, up to two bit errors can be detected and a single error can be corrected by decoding each triple of bits using a majority vote. Good codes achieve both a high minimal distance between pairs of codewords and a high rate. The former determines the number of errors that can be detected and corrected. The latter is a measure for how much information the code can transmit per bit. It is given as the ratio $\frac{\log_2 |\mathscr{C}|}{\log_2 |\Sigma^m|}$ of the logarithm of the number of codewords and the logarithm of the number of words to be encoded.

In watermarking, the application of error correcting codes is straightforward. Rather than embedding an actual message m, it is first encoded by an error correcting code $\mathscr{C}$ and the encoded message $E_{\mathscr{C}}(m)$ is then embedded in the signal. This leads to a reduction in the amount of data that can be embedded depending on the rate of the code. However, it leads to higher robustness of the watermark because a number of errors depending on the design of the code can be corrected. Hence, if the watermark has only a small number of bit errors, it can still be recovered. An example of Reed-Solomon codes used in image watermarking can be found in [33].

In fingerprinting, this process is often reversed. We assume that a fingerprint extracted from a signal is encoded as a bit string. This can then be understood as a code word from an error correcting code that might have been distorted during transmission. Only certain codewords can occur and again if a small number of bits

of a fingerprint is corrupted due to alterations of the underlying signal, these errors can be detected and corrected leading to a correct identification of the fingerprint. Examples of this approach can be seen in fingerprinting schemes designed to be applicable to highly distorted material [5, 23] or to timescaled material [4].

22.5 Conclusions

Watermarking and fingerprinting are proven and seasoned enablers of media synchronisation. Both techniques allow identifying a media stream and its current playback position for a wide range of media types including audio, image, and video. By providing this information, they make it possible to present secondary content precisely synchronised to the primary stream.

Since robustness issues and efficiency issues for individual queries can be considered solved, two main questions should guide the decision which approach is to be used in a synchronisation application:

- **Can the media stream be modified?** It is not always the case, in particular with broadcasters, that the media stream can be modified. In such a case, watermarking is not applicable.
- **What is necessary to scale to a large user base?** A main difference between the two techniques is the way in which they scale. Watermarking naturally scales with the number of users because the extraction of information from a watermark is a purely local process. In contrast, fingerprinting is often realised via a remote infrastructure responsible to solve the matching task. This infrastructure needs to be scaled according to the number of users, in particular, the number of simultaneous users.

This chapter is focussing mainly on technical aspects of enabling media synchronisation applications. Beyond this, whether or not such an application will become a success depends heavily on non-technical aspects such as user acceptance of the underlying technology, of the application itself, and its contents and style. This is discussed in the other chapters of this book.

References

1. Allamanche, E., Herre, J., Hellmuth, O., Fröba, B., Cremer, M.: Audioid: towards content-based identification of audio material. In: 110th AES Convention
2. Arzt, A., Widmer, G., Böck, S., Sonnleitner, R., Frostel, H.: Towards a complete classical music companion. In: Raedt, L., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.), ECAI, Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 67–72. IOS Press (2012)
3. Baras, C., Moreau, N., Dymarski, P.: Controlling the inaudibility and maximizing the robustness in an audio annotation watermarking system. IEEE Trans. Audio, Speech, Lang. Process. **14**(5), 1772–1782 (2006)

4. Bardeli, R., Kurth, F.: Robust identification of time-scaled audio. In: Audio Engineering Society Conference: 25th International Conference: Metadata for Audio, Jun 2004
5. Bardeli, R., Schwenninger, J., Stein, D.: Audio fingerprinting for media synchronisation and duplicate detection. In: Media Synchronisation Workshop (2012)
6. Böck, S., Schedl, M.: Polyphonic piano note transcription with recurrent neural networks. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 121–124, March 2012
7. Cano, P., Batlle, E., Mayer, H., Neuschmied, H.: Robust sound modeling for song detection in broadcast audio. In: Audio Engineering Society Convention, vol. 112, Apr 2002
8. Cano, P., Batlle, E., Kalker, T., Haitsma, J.: A review of audio fingerprinting. VLSI Signal Process. **41**(3), 271–284 (2005)
9. Clausen, M., Kurth, F.: A unified approach to content-based and fault-tolerant music recognition. IEEE Trans. Multimedia **6**(5), 717–731 (2004)
10. Coskun, B., Sankur, B., Memon, N.: Spatio-temporal transform based video hashing. IEEE Trans. Multimedia **8**(6), 1190–1208 (2006)
11. Coskun, B., Sankur, B.: Robust video hash extraction. In: 2004 12th European Signal Processing Conference, September 6-10, 2004, pp. 2295–2298. Vienna, Austria (2004)
12. Deguillaume, F., Csurka, G., O'Ruanaidh, J.J., Pun, T.: Robust 3d DFT video watermarking. In: Electronic Imaging'99, pp. 113–124. International Society for Optics and Photonics (1999)
13. Fastl, H., Zwicker, E.: Psychoacoustics: facts and models. In: Springer Series in Information Sciences. Springer, New York (1998)
14. Galar, M., Derrac, J., Peralta, D., Triguero, I., Paternain, D., Lopez-Molina, C., Garca, S., Bentez, J.M., Pagola, M., Barrenechea, E., Bustince, H., Herrera, F.: A survey of fingerprint classification part i: taxonomies on feature extraction methods and learning models. Knowl. Based Syst. **81**, 76–97 (2015)
15. Galar, M., Derrac, J., Peralta, D., Triguero, I., Paternain, D., Lopez-Molina, C., Garca, S., Bentez, J.M., Pagola, M., Barrenechea, E., Bustince, H., Herrera, F.: Experimental analysis and ensemble proposal. A survey of fingerprint classification part ii. Knowl. Based Syst. **81**, 98–116 (2015)
16. Haitsma, J., Kalker, T.: A highly robust audio fingerprinting system. ISMIR **107–115** (2002) (2002)
17. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **26**(2), 147–160 (1950)
18. Hampapur, A., Hyun, K., Bolle, R.M.: Comparison of sequence matching techniques for video copy detection. In: Storage and Retrieval for Media Databases 2002, January 23, 2002, pp. 194–201. San Jose, CA, USA (2002)
19. Hua, G., Huang, J., Shi, Y.Q., Goh, J., Vrizlynn, L.L.: Thing. Twenty years of digital audio watermarking—a comprehensive review. Signal Process. **128**, 222–242 (2016)
20. Kim, H.-G., Cho, H.-S., Kim, J.Y.: Robust audio fingerprinting using peak-pair-based hash of non-repeating foreground audio in a real environment. Clust. Comput. **19**(1), 315–323 (2016)
21. Kirovski, D., Malvar, H.S.: Spread-spectrum watermarking of audio signals. IEEE Trans. Signal Process. **51**(4), 1020–1033 (2003)
22. Kurth, F., Müller, M., Damm, D., Fremerey, C., Ribbrock, A., Clausen, M.: Syncplayer-an advanced system for multimodal music access. ISMIR **5**, 381–388 (2005)
23. Kurth, F., Ribbrock, A., Clausen, M.: Identification of highly distorted audio material for querying large scale data bases. In: In Proceedings 112th AES Convention, p. 9 (2002)
24. Law-To, J., Buisson, O., Gouet-Brunet, V., Boujemaa, N.: Robust voting algorithm based on labels of behavior for video copy detection. In: Proceedings of the 14th ACM International Conference on Multimedia, October 23–27, 2006, pp. 835–844. Santa Barbara, CA, USA (2006)
25. Lin, S.D., Shie, S.-C., Guo, J.Y.: Improving the robustness of DCT-based image watermarking against JPEG compression. Comput. Stand. Interfaces **32**(12), 54–60 (2010)
26. Lowe, D.G.: Object recognition from local scale-invariant features. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 2, pp. 1150–1157 (1999)

27. Meerwald, P., Uhl, A.: A survey of wavelet-domain watermarking algorithms. In: Proceedings of SPIE, Electronic Imaging, Security and Watermarking of Multimedia Contents III, pp. 505–516. SPIE (2001)
28. nan Li, Y.: Robust content fingerprinting algorithm based on sparse coding. IEEE Signal Process. Lett. **22**(9), 1254–1258 (2015)
29. Oostveen, J., Kalker, T., Haitsma, J.: Feature extraction and a database strategy for video fingerprinting. In: Recent Advances in Visual Information Systems, 5th International Conference, VISUAL 2002, March 11–13, 2002, Proceedings, pp. 117–128. Hsin Chu, Taiwan (2002)
30. Ouali, C., Dumouchel, P., Gupta, V.: Fast audio fingerprinting system using GPU and a clustering-based technique. IEEE/ACM Trans. Audio, Speech, Lang. Process. **24**(6), 1106–1118 (2016)
31. Radhakrishnan, R., Bauer, C.: Content-based video signatures based on projections of difference images. In: IEEE 9th Workshop on, Multimedia Signal Processing, 2007. MMSP 2007, pp. 341–344, Oct 2007
32. Radhakrishnan, R., Bauer, C.: Video fingerprinting based on moment invariants capturing appearance and motion. In: Proceedings of the 2009 IEEE International Conference on Multimedia and Expo, ICME '09, pp. 1532–1535. IEEE Press, Piscataway, NJ, USA (2009)
33. Ruanaidh, J.K.O., Pun, T.: Rotation, scale and translation invariant spread spectrum digital image watermarking. Signal Process. **66**(3), 303–317 (1998)
34. Seo, J.S., Haitsma, J., Kalker, T., Yoo, C.D.: A robust image fingerprinting system using the Radon transform. Signal Process. Image Commun. **19**(4), 325–339 (2004)
35. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J., **27**, 379–423, 623–656, July, Oct 1948
36. Shivakumar, N.: Detecting digital copyright violations on the internet. PhD Thesis. Stanford University (1999)
37. Tahboub, K., Neeraj, J.G., Mary, L.C., Edward, J.D.: An HEVC compressed domain content-based video signature for copy detection and video retrieval (2014)
38. Wang, A.: An industrial strength audio search algorithm. In: International Conference on Music Information Retrieval (ISMIR) (2003)

Chapter 23
Network Delay and Bandwidth Estimation for Cross-Device Synchronized Media

Mu Mu, Hans Stokking and Frank den Hartog

Abstract Driven by the growth in mobile phone and tablet ownership, recent years have witnessed an increasing trend towards coordinated media experiences across multiple devices. The quality of experience (QoE) over such new generation applications is dictated by the quality of service (QoS) of underlying networks. Inevitable network delay and bandwidth fluctuations affect the communications and media synchronization between connected devices. Therefore, network measurement is becoming the key to providing essential information for the QoE assurance of cross-device synchronized media. Amongst many network measurement techniques, packet probing is considered as the most effective for end-to-end evaluations. Packet probing may seem straightforward, but it requires a good understanding of the methodologies and how the results should be interpreted. This chapter provides a guide and some best practices in packet probing, accompanied with a use case where delay measurement enhances cross-device media synchronization and the QoE of an immersive media application.

Keywords Packet probing · Network monitoring · Inter-destination media synchronization · Immersive media · Quality of experience

M. Mu (✉)
The University of Northampton, Northampton, United Kingdom
e-mail: mumu@ieee.org

H. Stokking
TNO, The Hague, The Netherlands
e-mail: hans.stokking@tno.nl

F. den Hartog
DoVes Research, Canberra, Australia
e-mail: frank.den.hartog@dovesresearch.net

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_23

23.1 Introduction

There is an increasing trend towards coordinated media experiences across multiple devices, driven by the growth in mobile phone and tablet ownership [1–3]. People are spending far more time watching video content on their mobile devices. A recent study conducted by the Streaming Video Alliance shows that more than 40% of consumers watch more than an hour and over 25% are watching two or more hours of video per week on their smartphone [4]. Some projects are also looking to widen the experience beyond that of a single television image. Microsoft's IllumiRoom project augments the area around a television using projection, with visualizations designed to enhance gaming experiences [5], and a similar concept has been demonstrated by the BBC who produced a short film to demonstrate the potential of their Surround Video technology within a domestic, living-room environment [6].

One of the key challenges in delivering cross-device immersive experiences is media synchronization, which is heavily influenced by the delay and available bandwidth of the network. Research on the topic of media synchronization is conventionally categorized into *intra-stream synchronization*, *inter-stream synchronization* and *inter-destination synchronization* (IDMS). Intra-stream synchronization addresses the fidelity of media playback with respect to temporal relationships between adjacent media units within the same stream. Inter-stream synchronization refers to the preservation of temporal dependencies between the playout processes of correlated media streams [7, 8]. With the increasing demand for the accurate orchestration of media playback across associated end systems, IDMS has become a deterministic factor in assuring the quality of experience (QoE) over new generation media applications. Montagud et al. extensively reviewed 19 emerging media applications that require IDMS from the level of "very high" (10 μs–10 ms) to "low" (500–2000 ms) [7]. In a recent study on perceived synchronization of multi-sensory media, Yuan et al. concluded that users may tolerate haptic and air-flow media being one to three seconds behind corresponding video content [9]. On the topic of clock synchronization, Jin et al. introduced a synchronization approach for low-power wireless networks under a dynamic working condition, using timestamp exchanges and local measurement of temperature and voltage [10]. In the W3C Webtiming group, a Timing Object design capitalizes on a monotonic behaviour and applies skew changes gradually [11]. Readers can refer to Chap. 19 for more details. Gotoh et al. used a small packet train (a sequence of packets) to estimate forward and backward network delay separately. They combine this with NTP clock synchronization to compensate for the difference in forward and backward delays and thus improve timing synchronization across asymmetric networks [12]. Another application of bandwidth estimation in the media delivery context is in MPEG DASH (Dynamic Adaptive Streaming over HTTP). In DASH, normally clients measure available network bandwidth on the basis of the delivery of previously delivered media segments.

This basic method lacks sophistication, e.g. it does not take Transport Control Protocol (TCP) slow start mechanisms into account. Work is in progress to improve this, such as [13], which uses bandwidth estimation techniques such as the ones described in this chapter. Also, MPEG has worked on a new standard called Server and Network Assisted DASH (SAND), as explained in [14]. SAND allows the network (DANEs, DASH Aware Network Elements) to assist the DASH clients, e.g. by supplying them with network QoS parameters on available bandwidth and network latencies. Normally, a DASH client will start a media session with the lowest quality segments available and increase quality when bandwidth is found to be available. Having the network informing the client beforehand on available bandwidth allows the client to converge to the optimum much faster. For DANEs to be able to provide this information, they need to determine bandwidth and delays, e.g. using probing methods such as the ones described in this chapter.

Although the aforementioned research provides valuable insights into the user perception of media synchronization, it is often essential to accurately measure delay and available bandwidth in practice and use the results to dynamically improve the synchronicity of media. Packet probing is a collection of techniques that deal with measuring delays, available bandwidth and capacity on networks. Packet probing measurements are based on sending packets across the network in specific combinations (e.g. in short bursts or back-to-back packet pairs) and monitoring transmission and receipt times of such packets. By performing such measurements for various packets sizes or for various packets in a packet train, network delays, available bandwidth and (bottleneck) capacity estimations can be achieved. Packet probing can be done either active, i.e. actively sending out packets specifically for probing, or passive, i.e. by monitoring existing network traffic. Packets sent over the Internet encounter various types of delays. For probing methods to lead to accurate measurement results, the methods need to take the different types of delays into account. Because measurements are normally performed on end devices sending and receiving the probe packets, the actual measurements contain the sum of all delays introduced on the end-to-end network path. By using specific configurations of probe packets and certain algorithms (using transmission and receipt times of the probe packet), it is possible to derive specific measurements about the network path. Typical measurements results are one-way delay (as opposed to round-trip delay), path capacity and (bottleneck link) available bandwidth. This chapter provides detailed analysis and measuring methods of four types of delays, including *serialization delays*, *propagation delays*, *queuing delays* and *processing delays*. This is followed by a tutorial of two packet probing techniques for bandwidth estimation: variable packet size probing and probe gap method probing. The chapter also gives a use case study to demonstrate the incorporation of active delay measurement in a media application to ensure the QoE of synchronized media playout across user devices.

23.2 Overview of Network Delays and Delay Estimation Techniques

Network probing techniques use special packet configurations and measure transmission and receipt times of packets. Special algorithms are then used to derive information about the network, on delays, capacity and available bandwidth. For a good understanding of probing techniques, a thorough understanding of the various delays on end-to-end network paths is indispensable. This section explains the various delays in detail and the next section explains the main probing techniques. Together, this provides a valuable insight into the topic of network probing, which is applied in the remainder of this chapter.

23.2.1 Serialization Delay

Serialization delay of a packet of data is the delay caused by the bandwidth of the physical network connection (wired or wireless) on which packets are sent. It reflects the time that is needed to actually transmit the packet. For a packet of size L (bits) at a link of transmission rate C (bits/s), the serialization delay is equal to L/C. For example, to send a packet of 10,000 bits on a link with a transmission rate of 10,000,000 bits/s will take 0.001 s or 1 ms. This means that the serialization delay is dependent on the packet size and the link transmission rate. Serialization delay is depicted schematically in Fig. 23.1. Note that serialization delay of a packet is the time it takes to transmit that packet. If a significant amount of data is sent on the network, this data is sent using multiple packets. Packets are transmitted one by one; whilst one packet is actually being transmitted, subsequent packets are buffered (i.e. queued) whilst waiting their turn. In such a case, all packets of such data beyond the first suffer from both queuing delays (see next section) and serialization delays.

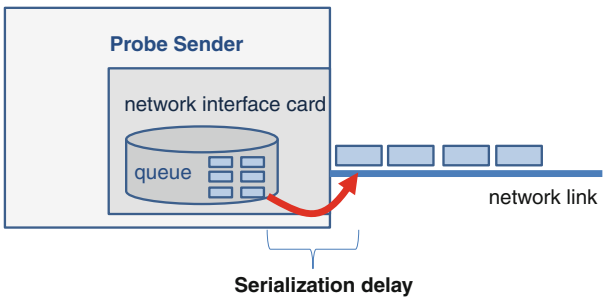


Fig. 23.1 Serialization delay

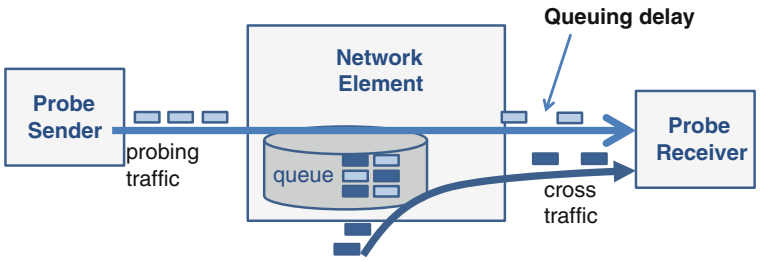


Fig. 23.2 Queuing delay

23.2.2 *Queuing Delay*

Queuing delay is the delay applied to a probe packet due to cross-traffic, as illustrated in Fig. 23.2. When multiple data streams are sent across the same network link, they are normally queued/buffered, allowing only one packet to be transmitted at a time on the network link. This can mean that a probe packet is buffered for a certain amount of time, waiting for its turn to be transmitted. This is known as queuing delay. Note that such queuing delay can occur in any device on the path between a sender and a receiver, including the sender itself. Since this delay is influenced by other data packets that traverse the network and the queue, it is independent of the probe packet size.

23.2.3 *Propagation Delay*

Propagation delay is the time it takes for a packet to physically traverse a network link. This time is dependent on the medium used for transmission and is independent of the packet size. For example, on a 50 m long Ethernet cable (CAT cable), the propagation delay is 50 m/177,000,000 m/s = 0.28 µs. 177,000,000 m/s is the speed at which the electrical signal travels across CAT cables; see also Table 23.1. Putting a 1500 byte packet (maximum normal Ethernet packet size) on a 1 Gbit/s network link causes a serialization delay of 1500 × 8 bits/1,000,000,000 bits/s = 12 µs. The propagation delay in this example is only 2.3% of the serialization delay. Thus, unless packets travel for very long distances or on very high network speeds, the propagation delay is negligible in most cases when performing network measurements. This is schematically depicted in Fig. 23.3. Table 23.1 shows the propagation speed on several different mediums.

Table 23.1 An overview of the propagation speed of various network mediums

Medium	Percentage of the speed of light in a vacuum (299,792,458 m/s) (%)	Propagation speed (km/s)
Thick coax	77	231,000
Thin coax	65	195,000
Twisted pair (e.g. CAT cable)	59	177,000
Fibre	66	198,000
AUI cable	65	195,000

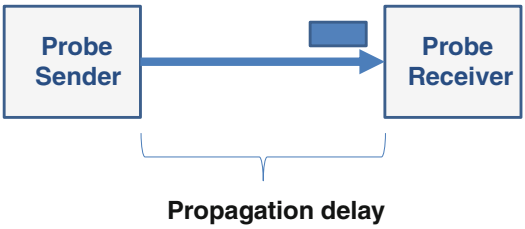


Fig. 23.3 Propagation delay

23.2.4 Processing Delay

To enable measurements, the probe sender and the probe receiver must put a timestamp on a probe packet for "packet sent" and for "packet received". The actions of timestamping the packets are usually conducted by a piece of software on a device; it is not integrated into the network hardware itself. Thus, when sending a packet, there is a small amount of time between the time that a timestamp is attached to the packet and the actual start of the packet transmission on the network. Similarly, at the receiver side, there is a small amount of time between the reception of a packet and the time that a corresponding timestamp is processed (Fig. 23.4). This small difference is called *processing delay*. For different devices, the timestamping action may occur in different parts of the device. Normally, a hardware network interface card can be used by the probing system via a device driver through a network stack (e.g. a TCP/IP stack). A popular tool for timestamping is *libpcap* [15]. For most Operating Systems (OSes), the packet is timestamped as part of the process of the network interface's device driver, or the networking stack, handling it. Depending on how the OS receives the packets from the network interface, the processing delays may vary. Also, in some cases, the processing delay may be even larger if the timestamping is done in software that is installed on top of the OS. This may be the case when access to the kernel is not available and timestamping is done in the application, e.g. in Web applications. The processing delay may also be dependent on the packet size, as the packets will have to be forwarded by the networked card across a hardware bus on the mainboard to a

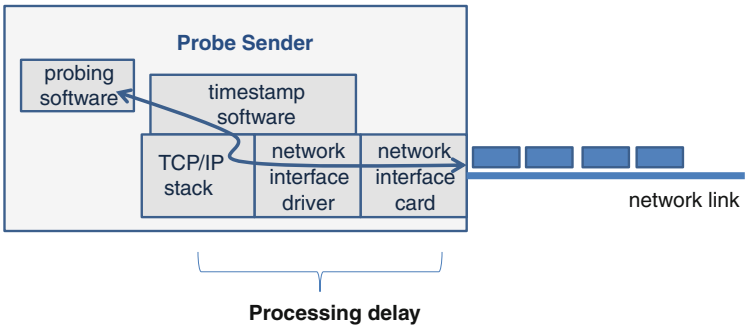
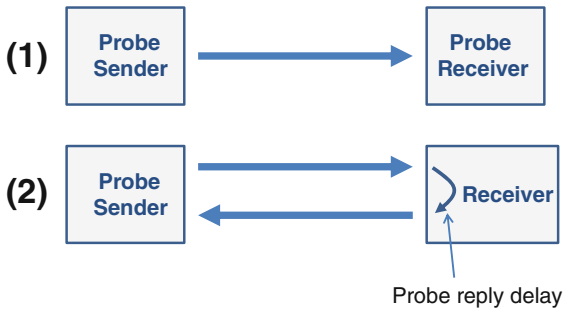


Fig. 23.4 Processing delay

Fig. 23.5 Probe reply delay



memory location elsewhere. Thus, the bus speed incurs a further serialization delay, and hence this will be dependent on the packet size.

23.2.5 Probe Reply Delay

In general, packet probing can be done in two different ways, as depicted in Fig. 23.5:

1. From a sender to a receiver (one way). The sender records outgoing timestamps whilst the receiver records incoming timestamps. In such a probing set-up, sender and receiver's clocks must be accurately synchronized.
2. From a sender to a receiver and back to the sender (round trip). In this case, the receiver sends a probe reply to the probe sender, after a probe request is received. The receiver does not need to conduct timestamping on the probe reply. The sender timestamps both the probe request sent and the probe reply received.

In the second situation, which is called a round-trip probe, the receiver will need some amount of time to process the request and formulate the reply. This will cause

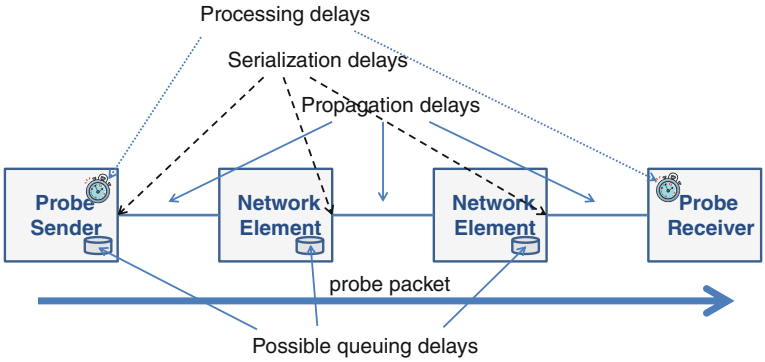


Fig. 23.6 Overview of the sources of delays in a network path from a probe sender to a probe receiver

Table 23.2 Overview of some characteristics that can be measured using network probing

	Capacity	Available bandwidth
Network link	Capacity on a link	Available bandwidth on a link
Network path	Capacity on a path, i.e. the bottleneck capacity on that path	Available bandwidth on a path, i.e. the bottleneck available bandwidth on a path

processing delay in addition to the serialization delay and propagation delay of this reply packet back to the probe sender.

Normally, probe packets go through several network elements, as shown in Fig. 23.6. In such a case, a probe packet will encounter numerous delays, including processing delays on the sender and receiver of the probe packet, serialization delays for each network link, propagation delays on each network link and possible queuing delays on each transmitting device.

When probing a network path, there are various characteristics to be measured. This is shown in Table 23.2. On the one hand, either capacity or available bandwidth can be measured. When measuring capacity, the delay of interest is the serialization delay. When measuring available bandwidth, which is the capacity not currently used for other traffic, the delay of interest is the queuing delay. As Fig. 23.6 shows, on a network path, there are often multiple points that introduce serialization delays and possibly multiple sources of queuing delays. When a network link is measured, the delays for that link need to be measured. When a network path is measured, it is the bottleneck link that dictates the measurement results. This bottleneck link is the link with the least throughput, i.e. the link with the highest serialization delay will determine the throughput of the entire path. Thus, when measuring the capacity of a network path, the measurement reflects the capacity of the bottleneck link. The same principle applies to available bandwidth

measurements, except that the determining delays are both queuing and serialization delays.

When measuring capacity or available bandwidth, it is essential to understand the characteristics of the network under measurement. Most network links nowadays are full duplex, meaning that they have capacity in the forward and the reverse direction at the same time. A symmetrical full-duplex link will have the same capacity in both forward and reverse direction. For instance, a 100 Mbit/s symmetrical full-duplex Ethernet link would have 100 Mbit/s capacity in both directions simultaneously. It can thus be said that the total capacity of this link is 200 Mbit/s.

If the goal of probing is to determine the capacity or available bandwidth, the delay used to measure this is thus the serialization delay and/or the queuing delay, as these are the delays that correspond to capacity and available bandwidth. These must thus be singled out. This chapter places emphasis on measuring capacity and thus focuses on measuring serialization delays.

The general method to single out serialization delay, i.e. to remove queuing delays from the results, is to send out a probe several times and then use the minimum value of all the measured delays. This is shown in the formula below.

$$delay\_min = \min_{[i=1\ldots n]} delay(i) \tag{23.1}$$

The logic behind this configuration is that cross-traffic is assumed to be stochastic. When enough probe packets were sent, one of these packets is assumed to be free of cross-traffic when going from the sender to the receiver. This particular packet will take the least amount of time to arrive, thus have the minimum value in measured delay of all the measured packets. In the further descriptions of the probe methods, this queuing delay is no longer discussed, and we assume that it has been dealt with by taking the minimum out of several measurements.

23.3 Bandwidth Estimation Techniques

Key to using probing techniques to measure bandwidth and available bandwidth is the delays, as explained in the previous section. Many different probing techniques exist, and for a recent overview of these techniques and of the various comparisons between them, we refer to [16]. The goal of this section is to explain how to use certain specific packet combinations to single out the various types of delays as described in the previous section, based on two well-known and illustrative example algorithms: Variable Packet Size (VPS) probing and Probe Gap Method (PGM) probing. Both methods have the advantage of being able to be used in round trip, thus only requiring implementation on the probe sender.

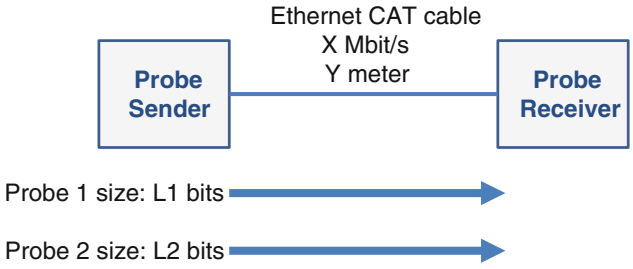


Fig. 23.7 VPS probing works by sending multiple packets with different sizes

23.3.1 Variable Packet Size Probing

The main idea of Variable Packet Size probing is to send probes of different sizes and to estimate the network capacity based on the differences in measured delays between the probes. Figure 23.7 shows a simple VPS method sending two probes of different sizes, probe 1 with size L1 (bits) and probe 2 with size L2 (bits). In this example, L2 is greater than L1. The letter "L" stands for "Length" of the packet.

The delays encountered by probe packet 1 are:

- Processing delay: Z1 s
- Serialization delay: S1 = L1/X s
- Propagation delay: Y/177,000,000 s

The delays encountered by probe packet 2 are:

- Processing delay: Z2 s
- Serialization delay: S2 = L2/X s
- Propagation delay: Y/177,000,000 s

Now suppose that the total delay measured for probe 1 is T1 and for probe 2 is T2. To calculate the capacity of the link, the following formula can be used (as explained further below):

$$C = \frac{L2 - L1}{T2 - T1} \tag{23.2}$$

In other words, the capacity is determined by dividing the size difference between the two probes by the difference in measured delay. The sizes are known, and the measured delays can be split into their various parts. We now get:

$$C = \frac{L2 - L1}{\left(Z2 + S2 + \frac{Y}{177,000,000}\right) - \left(Z1 + S1 + \frac{Y}{177,000,000}\right)} = \frac{L2 - L1}{Z2 + S2 - Z1 - S1} \tag{23.3}$$

One observation is that the propagation delay is cancelled out of the equation because the delay difference between multiple probe packets is used in the formula. Since the propagation delay is the same for the various probe packets, they cancel each other out in the equation. Next, the processing delays are assumed to be the same for both packet sizes. Or better said, the difference in processing delays for various-sized packets is near-zero. In practice, it turns out that the processing delays are negligible for all practical purposes, although it is unclear if this assumption holds for very high bandwidth links. But given this assumption, the formula for capacity becomes:

$$C = \frac{L2 - L1}{S2 - S1} \tag{23.4}$$

Our analysis suggests that the difference in measured total delays is equal to the difference in serialization delays. So, the measurement of the total delays can be used to determine capacity. Note that for actual measurements, normally more than two packet sizes will be used, and a linear fit will be used to determine capacity instead of a (straightforward) formula. This is shown in Fig. 23.8, a figure taken from a related work on network probing [17].

If probing is performed with a round-trip probe, probe reply delays are also assumed to be equal for different sized probes. This is done by making the probe reply packets all of similar size, regardless of the size of the probe in the forward

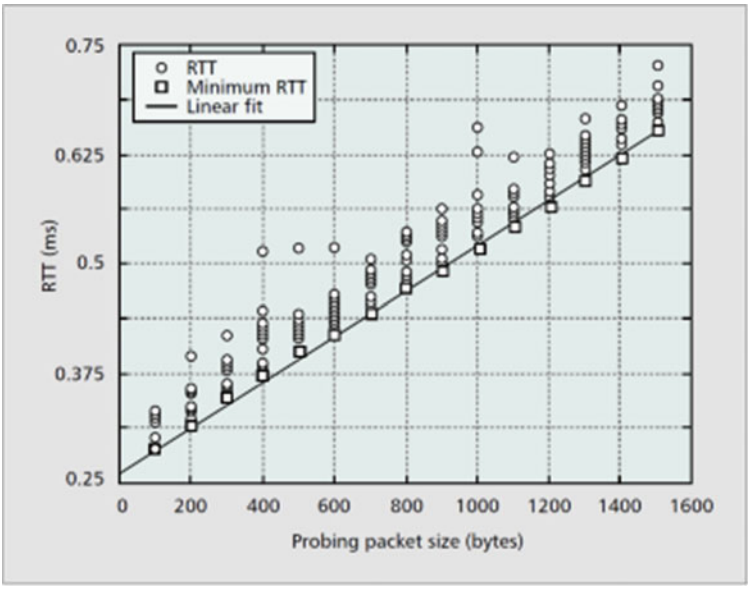


Fig. 23.8 Round-trip time (RTT) measurements, minimum RTTs and the least squares linear fit of the minimum RTTs for the first hop of a path [17]

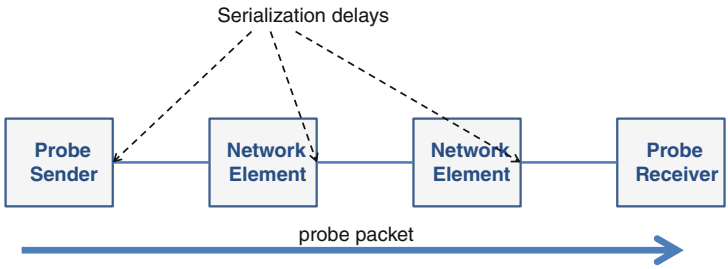


Fig. 23.9 A single probe packet on a network path will often suffer multiple serialization delays

direction. Under this assumption, the probe reply delays will also be cancelled out in the equation. For very high network speeds, this assumption will still need to be further investigated.

One of the main disadvantages of VPS is that it will not work correctly when the path of the probe consists of multiple network links, i.e. when it is performed in an end-to-end fashion, as shown in Fig. 23.9 and further explained below. VPS assumes that packets of different sizes have a different serialization delay. The assumption holds true except when probe packets travel through multiple links on a network path, where probe packets encounter multiple and possibly different serialization delays. The measured delay for the entire network path then contains multiple serialization delays instead of just one. Both S1 and S2 will contain these multiple serialization delays, and the difference between S1 and S2 will as well. As follows from Eq. (23.4), this will lead to a large underestimation of the capacity. Although for practical purposes the propagation delays and processing delays will be dealt with in the same way as for a single link, there will still be multiple serialization delays to consider.

To address this issue, VPS will probe each network element separately in a hop-by-hop fashion until the probe receiver is reached, as shown in Fig. 23.10. In this way, the measured delay of the first probe can be subtracted from the measured

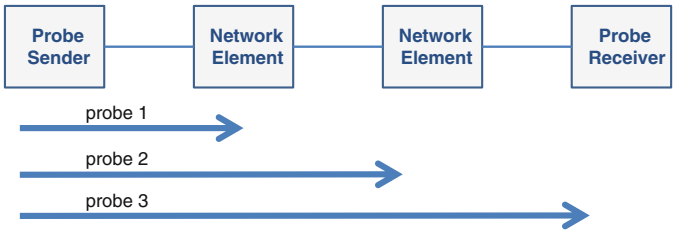


Fig. 23.10 For VPS, each network link on a network path should be probed separately

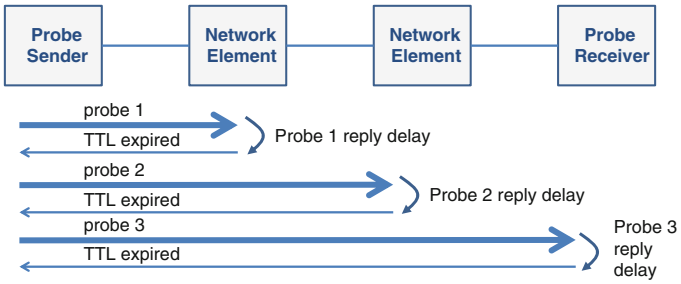


Fig. 23.11 VPS probing often makes use of the Time To Live (TTL) parameter in IP packets

delay of the second probe and so on. This will result in the measured delay per link, and the Eq. (23.4) above can be applied.

To conduct such progressive measurement in practice, VPS relies on the IP parameter "Time To Live" (TTL). This is a parameter that indicates the number of hops an IP packet may travel. Every router in the network path of the packet must subtract 1 from the TTL value, before forwarding it to the next router. If the TTL value reaches 0 after subtracting 1, a router must not forward the packet, but must instead send a single packet Internet Control Message Protocol (ICMP) "TTL expired" reply to the packet sender. VPS uses this parameter by sending out probes with increasing TTL values, thereby probing each network element in turn until the final link is probed, as shown in Fig. 23.11. This assumes that the reply delays for the different packet sizes are equal. Although the probes are sent with various probe sizes, the reply to the probe is always of the same size. This is because the replay is an ICMP TTL expired message, which is always of a certain size.

One thing to consider when applying VPS is the existence of switches in the network path, as is the situation in most home networks. A network switch is a layer-2 network element, i.e. working at the link layer, and does not work at the IP layer (layer-3). This means that a switch will neither inspect and adjust the IP TTL parameter nor will it sent out ICMP TTL expired packets. VPS probing will thus probe the link between two IP capable devices (i.e. routers), including any layer-2 elements (i.e. network switches) on that path. However, network switches can incur serialization delays. Any VPS capacity probing for a link between two routers that contains one or more switches can result in an underestimation of the capacity on that link, hence affecting the accuracy of the measurement.

Most VPS-based tools have been developed more than ten years ago (e.g. pchar [18]), and we are not aware of commercial implementations or heavy use by network managers today. However, VPS has recently been found particularly useful for estimating the link capacity for the sake of Software-Defined-Network-based (SDN-based) end host traffic control [19].

23.3.2 Probe Gap Method Probing

The second probing method is Probe Gap Method probing. The main idea of PGM probing is to send two probe packets of the same size back-to-back. This means that there is no delay in between sending the two packets, i.e. the first bit of the second packet is sent immediately following the last bit of the first packet. As these packets travel across various network links to their destination, they are influenced by various types of delays. Queuing delay is dealt with by taking the minimum from several measurements, and propagation delays will be equal if both probe packets travel the same links. Furthermore, since the packets are of equal size, processing delays, probe reply delays and serialization delays will also be the same for both packets.

What can cause dispersion between the two probe packets, though, is the serialization delay for each network link. Dispersion is defined here as the difference in arrival time of the first packet and the arrival time of the second packet. Network links with higher speeds have smaller serialization delays; network links with lower speeds have higher serialization delays. As both probe packets travel across different links, they are dispersed based on the serialization delays. The initial dispersion is caused by the first link, as the probe sender transmits the packets on the network. If the probe packets pass through a network link that is faster than at least one previously travelled slower link, the dispersion remains the same. However, if the probe packets reach a network link that is slower than any previously travelled link, the dispersion will increase due to the longer serialization delay.

As shown in Fig. 23.12, the middle network link is the slowest link. As the packets are sent on the network by the source on the first network link (on the left), they are dispersed. As the second link (the middle one) is slower, the serialization delay of the probe packets is higher than on the first network link. This causes the dispersion to increase. As the packets continue their route onto the third network link (the right one), the dispersion remains the same. Because both packets are of the same size, their serialization delay is equal, and the dispersion remains the same.

Since the size of the dispersion between the two probe packets at the receiver is determined by the slowest network links, this means that PGM can be used to measure the bottleneck link on the probe path. And, this means that PGM can be

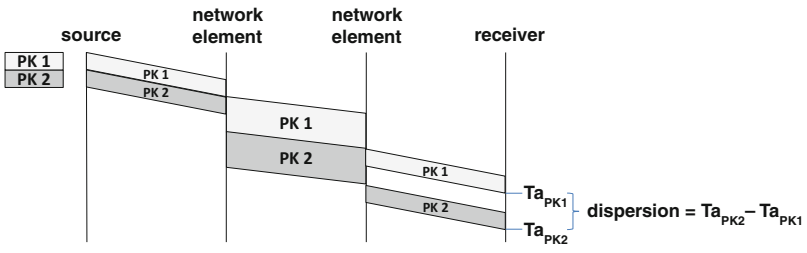


Fig. 23.12 Probe Gap Method measurement

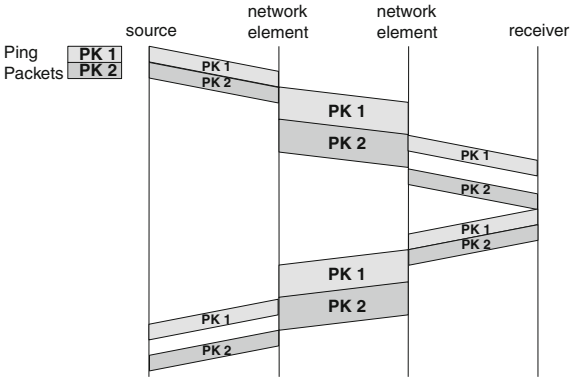


Fig. 23.13 PGM probing on a round-trip path using ICMP requests (Pings)

used *only* to measure the bottleneck link, as it is solely this link that determines the final dispersion when packets arrive at the receiver. Capacity on this bottleneck link can be determined by using the following formula:

$$C = \frac{L}{\mathrm{Ta}_{\mathrm{PK2}} - \mathrm{Ta}_{\mathrm{PK1}}} = \frac{L}{D} \tag{23.5}$$

The dispersion (D) is determined by subtracting the arrival time of the first packet ($\mathrm{Ta}_{\mathrm{PK1}}$) from the arrival time of the second packet ($\mathrm{Ta}_{\mathrm{PK2}}$). Capacity is then determined by dividing the size of the probe packet (L) by the dispersion.

PGM can be performed with a separate source and receiver, but can also be performed using round-trip probes, as shown in Fig. 23.13. In this case, the probe is a Ping packet (i.e. ICMP request and reply). Ping is a computer network utility that sends Echo Request packets from a source to a destination and measures the round-trip time for the Echo Reply packets from the destination to return to the source. Using Ping packets also means the packets are of the same size in the forward and reverse direction.

Using PGM in a round-trip fashion does have several implications. Often, for round-trip measurements, the popular Ping is used. An ICMP request and its reply are of similar size. When the network is symmetrical, i.e. the bandwidths on the links on the forward path are equal to the bandwidths on the links on the reverse path, using ICMP requests has no influence on the outcome. But, in the case of asymmetric links, i.e. forward and reverse bandwidth being different, this does have an influence. In the case of asymmetric links, the bottleneck link is in either one or the other direction. Using ICMP requests in this way does not show if the network is symmetric or not, it only gives the capacity of the bottleneck link.

The other choice of probe packet is an User Datagram Protocol (UDP) packet to a so-called unused port, i.e. a port that is normally not used for other purposes. Such an UDP packet will cause an ICMP "destination port unreachable" reply. Normally, the size of the UDP packet is chosen as large as possible, as this maximizes the length of the serialization delay, which will lead to maximum achievable accuracy.

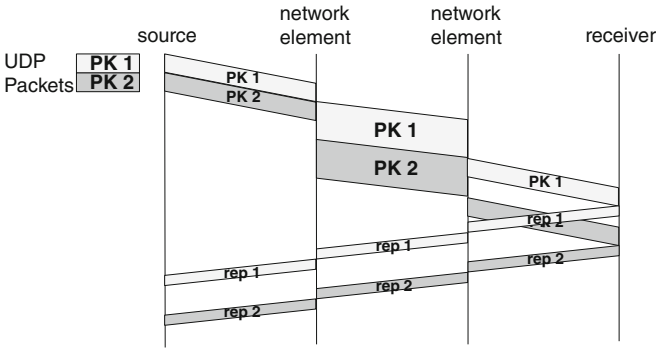


Fig. 23.14 PGM probing using UDP packets to an unused port causes small-sized reply packets from a receiver

The ICMP reply is a small-sized packet, as shown in Fig. 23.14. Any serialization delay for the ICMP reply will be much smaller than the serialization delay for the probe packet on the bottleneck in the forward direction. On extremely asymmetric networks, this may become an issue, as the serialization delay of the ICMP replies may be larger than the serialization delay of the probe packets on the bottleneck link. As a result, probing with UDP packets generally yields the bottleneck link capacity in the forward direction only.

For the use of PGM, two main limitations remain. The first one is that only the bottleneck link is measured. For many practical purposes, this is sufficient, e.g. to determine the speed from a specific client to a specific server. This speed will be limited by the bottleneck, so it is not a problem that PGM is limited to just that. But, for network monitoring purposes in general, this is too limited. PGM will not help in determining the capacity on a variety of links, whereas VPS is capable of just that. On the other hand, PGM can deal with switches and routers. Switches also incur serialization delay, so if the bottleneck link is between two switches or between a switch and a router, PGM will measure that correctly, as opposed to VPS. This assumes that the network elements use a store-and-forward mechanism, i.e. that packets are completely received by a network element before they are forwarded. This is the case for most network elements used throughout the Internet. The second limitation with the PGM probing method is that it does not discover the location of the bottleneck link, which can be any link in the probed path. When necessary, multiple probing points can be installed in a path to further investigate the root cause of the bottleneck.

A more recent development of PGM-based probing tool is Allbest [20], which proves the suitability of PGM for probing heterogeneous home networks (including Wi-Fi and Power Line Communications) for multimedia purposes.

Table 23.3 summarizes the strengths and weaknesses of VPS versus PGM. One of the strengths of PGM is that only some tens of probes are needed to obtain a fairly reliable estimate of the bottleneck link capacity. VPS (see Fig. 23.8) needs an

Table 23.3 A comparison of VPS and PGM

	Strengths	Weaknesses
VPS	• Straightforward and accurate calculation of link capacity • Can be used in round trip • Suitable for SDN-based end host traffic control	• Only works if the path does not contain layer-2 devices • If the path consists of multiple layer-3 devices, VPS needs to probe the path link after link, which is a tedious and error-prone procedure
PGM	• Limited amount of probe packets needed to obtain reliable result • Can be used in round trip • Is aware of layer-2 devices in the network • Suitable for heterogeneous networks	• Only measures bottleneck link capacity in a path • Does not provide information about the location of the bottleneck link in the path

order of magnitude more. However, this is still very little compared to the amount of traffic that popular techniques in the Internet use, such as Speedtest[1] and iPerf.[2] These techniques are typically based on Probe Rate Model probing, i.e. flooding the network [16].

23.4 Use Case Study

This section introduces a use case where media synchronization is the key to ensure a high degree of user experience whilst the synchronization hinges on timing measurement. The use case scenario considers a shared environment, such as a cafe, with a group of individuals that are drawn to an online video story that one of them recorded whilst attending an international music festival. After browsing the video library on a tablet device, one member of the group initiates playback of the recorded content to showcase the festival. The experience, however, is underwhelming and limited by the capabilities of the individual device—notably, the sound emanating from the tablet seems flat. To improve the experience, three of the group use their own smartphones and join the media playback. These additional devices contribute with a background soundtrack, along with ambient light and vibrations, which help to recreate the immersive experience of the musical event. This scenario serves to highlight the objective of this work: creating a vibrant and immersive orchestrated media experience across a series of heterogeneous user devices, connected via a range of networks (such as Wi-Fi and 4G—the fourth generation of mobile telecommunications technology), through the formation of a "device cloud" (Fig. 23.15).

[1] https://www.beta.speedtest.net/.

[2] https://www.iperf.fr/.

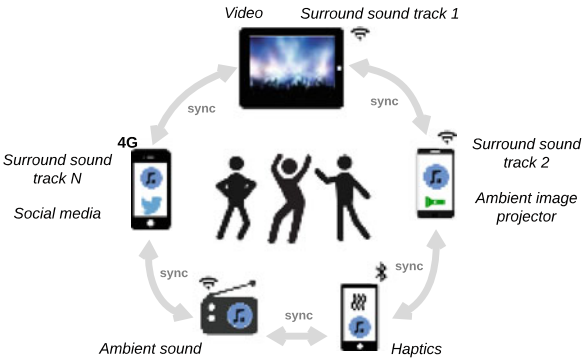
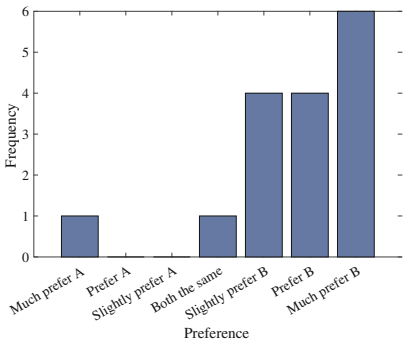
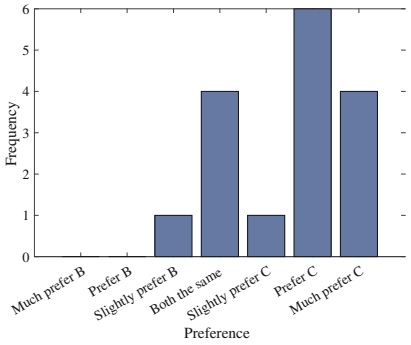


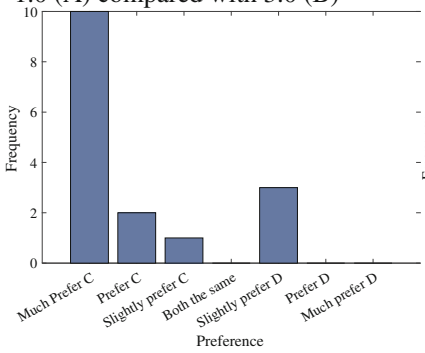
Fig. 23.15 Enabling immersive media experiences across multiple devices



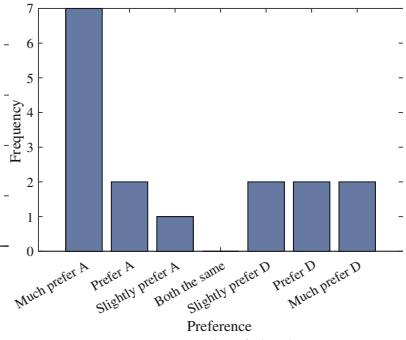
1.0 (A) compared with 3.0 (B)



3.0 (B) compared with 5.1 (C)



5.1 (C) compared with 5.1x (D)



1.0 (A) compared with 5.1x (D)

Fig. 23.16 Subjective paired comparison results

23.4.1 Reference Devices

To realize this use case, three reference devices are needed: *master device*, *auxiliary device* and *sync server* with each representing a specific role within a multi-device environment. The media playback session on the master device is the temporal reference point of all auxiliary devices. Auxiliary devices may join at any point to enhance the media experience whilst maintaining their synchronicity to the master. The sync server is a central point where measurements related to playhead position and player statistics are gathered and dispatched. The playback statistics of all devices are monitored and logged by the sync server using sync signalling (detailed in the next section). When the sync server detects a noticeable gap (i.e. asynchrony) in playhead position between any auxiliary device and the master device, a sync message will be sent to the corresponding auxiliary device with additional timing information such as the master device's playhead position and the round-trip network delay of both master device and the auxiliary device. Using such information, the auxiliary device then studies the degree of asynchrony and determines the optimal solution for re-synchronization based on impact to user experience and the capacity of relevant devices. In a recent work [8], we investigated the joint perceptual impact of asynchrony and media playback rate change and developed a model to eliminate cross-device media asynchrony with the least perceivable accumulative impact.

23.4.2 Sync Signalling and Delay Measurement

To accurately orchestrate the media playback on connected devices, it is essential to measure the discrepancy between the playhead positions of media streams. It is a common practice to employ a sync server which operates as a relay node to help exchanging information (sync messages) such as live playback statistics amongst connected devices. One of the key challenges in using sync messages is the time referencing when constructing and interpreting sync messages.

The most straightforward means of referencing time is to exploit the absolute time provided by the internal clocks on user devices and use it to timestamp each event (e.g., "Device 1 is playing frame number 326 at (local) time 15:56:12.240"). To ensure the synchronicity of the clocks, Network Time Protocol (NTP) [21] is commonly used to adjust a device's clock using a time broadcast by an NTP server, with transmission time compensated for by a one-way delay (OWD) metric. Clocks may also drift after NTP synchronization; hence, some time-critical applications require the clock synchronization process to take place periodically. The NTP-based clock synchronization requires additional ports and connections at user devices. It should also be recognized that clock synchronization requires, by its nature, long periods to maintain accurate timekeeping. Sufficient measurements must be taken, which involve periodic exchange of NTP messages, to allow

statistically stable measurement. An alternative design, which departs from the conventional designs with dependencies on NTP, employs Web technologies such as WebSockets to enable efficient full-duplex communication channels for the exchange of timing information directly and synchronously. Using this mechanism, playback control messages such as "skip to playhead position 1" are executed immediately as synchronous messages when they are received. Time offsets are appended to the control messages to compensate for different types of delays in the network and system (detailed in the following paragraphs). Therefore, the solution does not require the clocks of corresponding devices to be synchronized with each other. All devices must establish a socket connection with the sync server, which uses a "heartbeat" mechanism to send out periodic (e.g. every second) "keep-alive" messages to the clients to see if they are still online. The clients subsequently respond with an *ACK* acknowledgement message with standard player data (such as what the player is playing, playback speed, buffer level, playhead position) attached.

Using the statistics gathered through ACK messages, the sync server maintains the playhead position information on all connected devices. It also has specially designed mechanisms to compensate for any signalling or playback delay (detailed in the following paragraphs). In a typical digital TV scenario, media servers process all media content and interleave time bases as part of the media transport streams in order to measure and control media synchronicity at the client side [22, 23]. Such an approach is not feasible for distributed user-generated content. The use case capitalizes on the timing measurements discussed in the last section to actively measure network delays.

In practice, sync messages carrying the current playhead position of the master device p_{Master} may take the time of Δ_{t_0} to arrive at an auxiliary device, by which time the master has already a new playhead position of $p_{Master} + \Delta_{t\_0}$. When a media player is instructed to adjust its playhead position (e.g. by seeking forward to a specific point in the media stream), the instruction will be delayed by Δ_{t_0} in transmission due to network latency. Moreover, when the media player re-synchronizes by adjusting its playhead position, the media player must request a new data range from the content server and wait for the player buffer to be filled to a certain level before the playback can resume. This process often leads to an additional processing delay of Δ_{t_1} determined by the available bandwidth, the buffer size and/or buffering strategy at the end device. Without additional delay measurement and compensation, the streams at auxiliary devices may lag the master for $\Delta_{t\_0} + \Delta_{t\_1}$, which could be on the scale of hundreds of milliseconds to tens of seconds. To mitigate the impact of such a delay, the sync signalling module monitors the round-trip time of the sync messages exchanged between user devices and the sync server and estimates the network delay Δ_{t_0}. This is similar to the design principle behind NTP, but executed and maintained natively. Because the sync messages, functioning as probe packets, conducts delay measurement directly, the mechanism is more efficient for interactive media applications, having very little overhead. Furthermore, since the probing process is managed by the same application that conducts sync messaging, it measures not only propagation, serialization

and queuing delay, but also processing and probe reply delay for a media application. For the additional media processing delay Δ_{t_l} of a media player requesting and presenting media content retrieved from a media server, the framework logs the time that an instruction is sent from the playback management to the media player and the time that the media player completes the execution. The delay between the two is monitored to form a statistical measurement of the media processing delay Δ_{t_l}. Such a measurement process requires interactions with the playback and buffer management module on the media player.

Using this method of measuring delays has many advantages compared to probing with, for instance, Real-time Transport Control Protocol (RTCP) over Real-time Transport Protocol (RTP) [24]. When using RTCP,

- The RTCP probing message must be timestamped using NTP or any other common, system-specific clock such as "system uptime", introducing the various complications as detailed above,
- Since RFC3611 [25] has defined an extensible structure for reporting using an RTCP Extended Report (XR), it has become difficult to determine the serialization delay accurately, as the packet size of the RTCP messages is not a given constant anymore.

23.4.3 Implementations

The design has been implemented using open Web technologies, such as JavaScript. Any user device that supports JavaScript can participate in the delivery of immersive media without additional plug-ins. A customized Node.js server operates as the sync server handling device discovery and sync signalling. Node.js (nodejs.org) is an open-source, cross-platform JavaScript runtime environment for executing JavaScript code server-side.

After a client receives heartbeat messages from the sync server, it immediately acknowledges the server with a response message. The message is used by the server to check the status of the connected clients. Information about each client's player is piggybacked onto the heartbeat acknowledgement back to the sync server, shown in JSON Listing 23.1. This is mainly data referring to the media currently being played, directly from the HTML5 player (e.g. currentTime, ended, readyState and networkState). Once received at the server, it is timestamped and stored.

The heartbeat messages are also used to calculate the current network delay between all the clients and the server, using a probing mechanism where timestamping is made on heartbeat messages on sending and receiving. This round-trip time data is stored alongside the player data for each client. The player data, state and network measurements data then serve as a central reference to be used when creating or maintaining synchronicity.

As an up-to-date record of the player data is maintained, when the sync server identifies a detrimental degree of asynchrony to the master on a user client, it will

```
"frame":630,
"buffered":{}.,
"currentTime":25.230594,
"networkState":1,
"paused":false,
"playbackRate":1,
"played":{},
"readyState":4,
"seekable":{},
"duration":102.656
```

Listing 23.1 Heartbeat sent from clients to the sync server

```
target:iydGdPh-uNlDKcpRLbkU,
action:sync
```

Listing 23.2 Sync log

internally register such a sync request and log its socket connection ID shown in JSON Listing 23.2. The sync server can then collate all of the required data (i.e. master player data, delay measurements and auxiliary device network measurements) necessary for the auxiliary device to make the precise decisions for re-synchronization, as shown in JSON Listing 23.3. The messages also include last_hb and server_time in epoch time (in milliseconds). last_hb is the time when the latest heartbeat was received from a client, and server_time is the sync server time when the message is sent. These two pieces of timing information are used to verify the age of heartbeat messages in relevance to the server time. If the age is very large, information carried by the corresponding messages might be discarded.

Once the client receives the sync message with an action such as skip, an arrival timestamp is added and an immediate decision is made based on the difference between the master and auxiliary players' current playhead positions. The true current playback time of the master p_{Master} can be calculated by accounting for the time taken for the master's playhead position to traverse the network (half the two RTTs, assuming that the network is symmetrical) and for its time spent at the sync server. Details of the system set-up and sync configuration can be found in [8, 26].

```
"action":"sync",
"master_pd":{
        "frame":1230,
        "currentTime":49.217563,
        "networkState":1,
        "paused":false,
        "playbackRate":1,
        "played":{},
        "readyState":4,
        "duration":102.656,
},
"master_rtt":{
        "last_hb":1473024748184,
        "rtt":112,
        "rtt_sum":6853,
        "rtt_count":42
},
"server_time":1473024748723,
"aux_rtt":{
        "last_hb":1473024748183,
        "rtt":112,
        "rtt_sum":1405,
        "rtt_count":8
}
```


Listing 23.3 Sync message sent from sync server to auxiliary clients

23.4.4 Evaluation

We conducted an objective evaluation of the system performance in orchestrating synchronized cross-device media playback assisted by network probing mechanisms. Details of the evaluation can be found in [8]. The QoE of synchronized media is ultimately determined by the subjective human perception. The remainder of this chapter introduces our recent subjective evaluation to test the effectiveness and perceivable impact of delay measurement in synchronized media applications. We set up a realistic environment using a range of typical end-user devices. Specifically, the test investigates: (i) Would the coordinated delivery of associated media across multiple user devices greatly enhance the user experience? (ii) Is the user experience in any way correlated to the number of participating devices? (iii) Does media synchronization (enabled by delay measurement and compensation mechanism) improve the overall user experience in a multi-device configuration?

Table 23.4 Surround sound device configuration

Position	Device	Bandwidth
Video	MacBook Pro	20 Mbit/s
Centre speaker	Android phone A	2 Mbit/s
Left speaker	Android phone B	Not limited
Right speaker	Android tablet	3 Mbit/s
Rear left speaker	Raspberry Pi A	4 Mbit/s
Rear right speaker	Raspberry Pi B	3 Mbit/s
Low-frequency speaker	Raspberry Pi C	6 Mbit/s

The evaluation was established in a specifically configured office and comprised of various devices, including Android phones, an Android tablet, a MacBook Pro and Raspberry Pis with speakers attached, are connected using wired or wireless networks. To individually modify the networking characteristics of each connected device, we used the *netem* emulator [27]. With *netem* operating between the devices and media source, we applied bandwidth restrictions as shown in Table 23.4.

Our evaluation entailed showing participants a 2-min long *Star Wars Rogue One* trailer with 5.1 surround sound. The audio channels were encoded into separate Advanced Audio Coding (AAC) files plus a single video file from the original lossless Free Lossless Audio Codec (FLAC) audio source. User devices could request individual elements of the trailer (e.g. the audio for the front right speaker). As shown in Table 23.5, a total of four test conditions were generated (identified as A-D). For all the test conditions, the system is configured to conduct Adaptive Media Playout (AMP), which dynamically adjusts the playback rate of the media player (on auxiliary devices) to reduce its perceivable asynchrony to the master device. The first three test conditions exploit delay measurement (i.e. probing mechanism) to compensate any delays. We study the performance of the system in measuring the delay over an increasing number of clients, from 1.0 mono audio to 3.0 and 5.1 surround sound. The fourth test condition uses 5.1 surround sound but does not incorporate any delay measurements nor corresponding media sync adjustments. Details of the test configurations are shown in Table 23.5.

Table 23.5 Subjective evaluation test configurations

Test	Abbr	Description
A	1.0	Two devices: video and centre audio
B	3.0	Four devices: all devices from A plus two devices for front left and front right audio
C	5.1	Seven devices: all devices from B plus two devices for rear left and right audio and one additional device for low-frequency effects (bass)
D	5.1x	Seven devices: all devices from C but without active delay measurement and media synchronization mechanisms

A paired comparison was conducted between each subsequent test case by each participant using a comparison scale such as *much prefer A, prefer A*, *slightly prefer A*, *Both the same*, *slightly prefer B*, *prefer B*, *much prefer B*. The combinations of paired comparison include "1.0 (A) compared with 3.0 (B)", "3.0 (B) compared with 5.1 (C)", "5.1 (C) compared with 5.1x (D)" and "1.0 (A) compared with 5.1x (D)". After this, each participant underwent a small interview and was asked to describe their experience of each test case. In the interview, we asked each participant three questions (1) "Do you have any comments about what made any particular test case annoying to watch?" (2) "Given the scenario of watching a video on somebody's device; would you be willing to contribute your own device (e.g. mobile, tablet, wearable) to enhance the overall user experience?". A total of 16 participants completed this study: 11 males, 4 females and one preferred not to say, with 7 aged between 18–30, 2 between 31–40 and 4 older than 40. Although the number of participants is not sufficient to support statistical modelling, the evaluation with 16 participants gives a strong indication of how synchronized media delivery (with the help of network measurement) ensures user experience in a cross-device immersive application.

In the first test, when comparing 1.0 with 3.0, the vast majority (87.5%) said they preferred 3.0 over 1.0, with this scenario being our most obvious preference; see Fig. 23.16. This clear preference can be attributed to the limited quality and volume of the mobile device speaker which in test case B was boosted through additional devices. This is supported by our participant responses to question 1, which included "[Disliked 1.0] only one sound source" and "Video A [(1.0)] did not have great sound quality". These negative aspects towards test case A could be addressed by using a device with a higher quality speaker, but this comparison does confirm that the user experience is enhanced using multiple coordinated devices over a single mobile device.

Comparing 3.0 and 5.1, 68.75% of participants said they preferred 5.1; see Fig. 23.16. The experience is once again enhanced with the addition of three more devices, creating a positive correlation of improvement. The preference is less significant than the 1.0–3.0 comparison, but the additional devices, in this case, were providing a low-frequency speaker and two speakers for the rears, which might be considered to be less significant than the front speakers in a surround sound configuration. One participant was not in favour of this configuration, stating that "Video C [(5.1)] had too much going on" suggesting that a minority of people may be uncomfortable with this set-up due to sensory overload.

When comparing the 5.1 test cases where the delay-compensated media synchronization is enabled and disabled, 81.25% of the participants said that they preferred the 5.1 configuration with media synchronization enabled; see Fig. 23.16. This would indicate that delay measurement capabilities do have a positive impact on the overall user experience and are required for a successful synchronization. One participant affirms this view during the interview, stating "(Cross-device) lip sync with video D was particularly problematic—and thus particularly annoying" highlighting that without synchronization the levels of non-synchronicity are clearly perceivable, particularly during periods of dialogue within the trailer.

A final comparison was made between 1.0 and 5.1x (without synchronization) to query whether users would rather have a single device experience over a multi-device experience without complete synchronicity; see Fig. 23.16. Although 62% of participants preferred a single device, there remains a significant minority that would favour multiple devices, despite them being slightly out of sync (i.e. not synchronized to each other). We consider this comparison therefore somewhat less conclusive and will ultimately depend on an individual's personal preference of sacrificing quality over synchronicity.

When participants were asked during the interviews whether they would be willing to contribute their own devices to receive an enhanced experience, over half of the participants responded positively. However, there were concerns raised about the security associated with connecting to other people's devices and the potential implications for battery consumption. One participant said "*Yes, probably. Depends on the access method; would trust APIs (Application Programming Interface) in iOS more than a third-party magic app!*" whilst another suggested "*Yes, if I trust this person that I wouldn't get a virus from his device. Not if there is secret data on my device (like company data)*". Finally, the interviews gave an opportunity for participants to suggest additional devices that could contribute towards a more immersive media experience. A multi-view scenario and the use of Internet of Things (IoT) devices, such as digital wallpaper, were amongst some of the suggestions, "*Imagine if, watching Star Wars in the same room, one viewer could see things from the Imperial perspective and the other viewer could see the Rebel perspective…*", and having additional devices "*…integrated into a chair or something…*" to create a multi-sensory experience.

23.5 Conclusions

Recent years have seen the increasing attention and development in cross-device immersive media. Media synchronization is the key to ensure the user experience of the new media. Maintaining the synchronicity between devices and applications hinges on accurate network delay and bandwidth measurement. This chapter has given an overview of the different types of delays that a cross-device media may encounter as well as details of the probing methods that can be used to actively estimate delay and available network capacity. We have also analysed the advantages and limitations of using network probing methods as lessons learned from our experience. In addition, incorporating network probing in media applications is challenging in practice. We introduced a recently developed cross-device media application as a use case to demonstrate how probing mechanisms can be embedded in the application to improve the QoE using Web technologies. A subjective user experiment of a surround sound test further proved that the synchronicity of media is a deterministic factor of the user experience in immersive media.

Definitions

Network probing Network probing techniques use special packet configurations, and measure transmission and receipt times of packets to derive information such as delays, capacity and available bandwidth of a network path.

Cross-Device Synchronized Media Associated media content delivered synchronously across multiple (types of) devices to elicit immersive user experience.

References

1. Geerts, D., Leenheer, R., De Grooff, D.: In front of and behind the second screen: viewer and producer perspectives on a companion app. In: Proceedings of the ACM International Conference on Interactive Experience of Television and Online Video (TVX 2014)
2. Cesar, P., Bulterman, D.C.A., Jansen, A.J.: Usages of the secondary screen in an interactive television environment: control, enrich, share, and transfer television content. In: Changing Television Environments, pp. 168–177. Springer (2008)
3. Total Audience Report Q3 2016. The Nielson Total Audience Series (2017)
4. Mobile Video: Exposed, in Streaming Video Alliance (2016)
5. Jones, B.R., et al.: IllumiRoom: peripheral projected illusions for interactive experiences. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems
6. Thomas, G., et al.: Surround Video. White Paper WHP 208. https://www.downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP208.pdf
7. Montagud, M., et al.: Inter-destination multimedia synchronization: schemes, use cases and standardization. Multimed. Syst. (2012)
8. Mu, M., et al.: Closing the gap: human factors in cross-device media synchronization. IEEE J. Sel. Top. Sign. Process. (2016)
9. Yuan, Z., et al.: Perceived synchronization of mulsemedia services. IEEE Trans. Multimed. **17**(7), 957–966 (2015)
10. Jin, M., et al.: DualSync: taming clock skew variation for synchronization in low-power wireless networks. In: The 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016. IEEE (2016)
11. W3C, Multi-device timing architecture. https://www.w3.org/community/webtiming/architecture/ (2016)
12. Gotoh, T., Imamura, K., Kaneko, A.: Improvement of NTP time offset under the asymmetric network with double packets method. In: Conference Digest 2002 Conference on Precision Electromagnetic Measurements, 2002. IEEE (2002)
13. Jeong, J.-M., Kim, J.-D.: Effective bandwidth measurement for Dynamic Adaptive Streaming over HTTP. In: 2015 International Conference on Information Networking (ICOIN). IEEE (2015)
14. Thomas, E., et al.: Enhancing MPEG DASH performance via server and network assistance. SMPTE Motion Imaging J. **126**(1), 22–27 (2017)
15. Jacobson, V., Leres, C., McCanne, S.: libpcap: a portable C/C++ library for network traffic capture. https://www.tcpdump.org/
16. Paul, A.K., Tachibana, A., Hasegawa, T.: An enhanced available bandwidth estimation technique for an end-to-end network path. IEEE Trans. Netw. Serv. Manage. **13**(4), 768–781 (2016)

17. Prasad, R., et al.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Netw. **17**(6), 27–35 (2003)
18. Mah, B.: Estimating bandwidth and other network properties. In: Internet Statistics and Metrics Analysis Workshop on Routing and Topology Data Sets: Correlation and Visualization (2000)
19. Al-Najjar, A., et al.: Link capacity estimation in SDN-based end-hosts. In: 2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS). IEEE (2016)
20. Delphinanto, A., Koonen, T., Den Hartog, F.: Real-time probing of available bandwidth in home networks. IEEE Commun. Mag. **49**(6) (2011)
21. Mills, D., et al.: Network time protocol version 4: protocol and algorithms specification (2010)
22. Ferreira Moreno, M., de Resende Costa, R.M., Gomes Soares, L.F.: Interleaved time bases in hypermedia synchronization. IEEE MultiMed. **22**(4), 68–78 (2015)
23. Yuste, L.B., et al.: Understanding timelines within MPEG standards. IEEE Commun. Surv. Tutor. **18**(1), 368–400 (2016)
24. Schulzrinne, H.: A transport protocol for real time applications. RFC 3550 (2003)
25. Friedman, T., Caceres, R., Clark, A.: RFC 3611. RTP Control Protocol Extended Reports (RTCP XR), pp. 1–49 (2003)
26. Mu, M., et al.: QoE-aware inter-stream synchronization in open N-Screens cloud. In: The 13th Annual IEEE Consumer Communications & Networking Conference
27. Netem. https://www.wiki.linuxfoundation.org/networking/netem

Afterword

It is a pleasure for me to contribute a personal afterword to this interesting compendium of chapters on media synchronization. The topic of content synchronization has been central to most of my research activities during the past 30 years. Reflecting on this past (and the work that others have done during this period) helps focus my own attention on the role that media synchronization can play in the future.

In many ways, media synchronization remains an under-appreciated problem. For most researchers, media synchronization has consisted of little more than a discussion of the ways that media jitter—the intra-stream packet arrival delay associated with a single media object—can be quantified and managed. The (often implied) temporal relationship among these packets made transferring continuous media content more complex than sending the discrete blocks of textual content for which common networking infrastructures were designed. When viewed solely as a technical, networking problem, media synchronization is reduced to a fairly straightforward problem of predicting the quality of a given network connection and making a go/no-go decision on accessing a media object. While this characterization seems fairly trivial, it is in fact the model for end-user adapting to synchronization problems: if the intra-content delay becomes unbearable, simply stop watching that content.

Of course, the simplifications of the intra-stream packet model are an illusion, although a rather persistent one. In the more general case, a *media object* (as experienced by the viewer) is much more dynamic: the various components of this (real or virtual) object may be stored in multiple independent content streams that get glued together at the 'experiencing' end; each of the streams may be decomposed into a number of alternative components that can be used when adapting for network delays or user interests; the entire presentation may be subject to a number of commercial, legal, or user-preference constraints; and the entire content collection being bundled into a single presentation may be created dynamically rather than consisting of pre-stored bits. There are multiple levels of media synchronization that take place across the infrastructure (thus: at servers; across the network; and at the client), all of which need to be coordinated for effective content delivery. All of this synchronization is, at its heart, time-based, but must be

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7

implemented across infrastructures when a common notion of time is probably the single most relevant part of the processing chain that *does not* exist. It is this collection of issues that makes media synchronization interesting!

For many years, my research group (along with many others) worked on developing models of time and temporal processing for media-based networked presentations. Some of this work was presented earlier in this book. The goal of much of this work was not to develop temporal specification formalisms for reasoning about a presentation, but to develop a complete specification of the temporal issues that an infrastructure needed to address when delivering a complex media presentation. For many of us, this remains a fundamental problem: how can you define a model of a presentation that includes explicit control of content alternatives (both syntactic and semantic), nonlinear navigation that allows users follow their own narrative rather than being restricted to what a content provider thought might be interesting, and a model that realizes that most aspects of content selection and navigation are determined at the moment of viewing rather than at the moment of authoring. These, too, are at the heart of supporting media synchronization. The unifying theme presented by these issues is: understanding how content and context of use can be explicitly modeled to support the natural changes that occur over time.

One of the interesting changes that have occurred over time is in better understanding the viewer's tolerance to poorly supported media synchronization. In the later 1980s and early 1990s (when many of the fundamental problems of computer-based synchronization were being studied), it was assumed that viewers had no tolerance (beyond a few milliseconds) for content such as video and audio that was out of sync. Currently, we realize that an ideal model of content delivery in which these synchronization issues are solved is not only technically challenging, but also unnecessary. At least for conventional media-based content, users have become conditioned to adapt to delays within and across streams in certain contexts of use. This has allowed infrastructure providers to treat content synchronization relationships on a best-effort rather than a hard-synchronization basis. This relaxed view remains context dependent: accepting the occasional 'spinning pizza' while viewing content during an evening commute is different from experiencing the same delay in a movie theater. Part of the user tolerance to rebuffering on personal devices possibly reflects an economic trade-off, and part of it is a fatalism that assumes that there will never be an infrastructure with enough available bandwidth to support all the world's appetite for simultaneous watching of pet-trick videos. Still, at the theater, at a concert, at the movie theater, our expectations are different. All of these 'at' situations provide hope for researchers that continue to push the 'true-experience' feeling more closely to the user.

Looking toward the future, I expect that a more generalized notion of *content synchronization* will take on an increasing important role in both computer networking/systems research and research related to understanding the meaning of multiple streams of incoming information at the 'experience' level. The wide deployment of largely autonomous sensors and Internet-of-things devices provides a compelling basis for control and synchronization study, one in which knowing what not to process may become more important than synchronizing the left-over content.

I also hope that new insights develop in processing highly personalized, nonlinear, semantically differentiated content. In the long term, helping to synchronize a common understanding of content meaning may be more important than 'simply' delivering that content in a tightly controlled manner.

I'd like to thank the editors and authors of this book for putting together an interesting collection of chapters that do justice to the broad complexity of media synchronization. While many of us can see our own 'past' in this collection, it is comforting that many more will be able to determine their own 'future' in further studying the challenges that are outlined in this book's chapters. I hope all who have read the content—linearly or nonlinearly—have enjoyed it as much as I have.

Dick C. A. Bulterman
Centrum voor Wiskunde en Informatica (CWI), Amsterdam, *and*
Vrije Universiteit Amsterdam