Telecommun Syst (2016) 61:59–76
DOI 10.1007/s11235-014-9952-x

A distributed transcoding and content protection system

Enabling pay per quality using the cloud

Daniel Díaz-Sánchez · Rosa Sánchez-Guerrero · Patricia Arias · Florina Almenarez · Andrés Marín

Published online: 1 January 2015

Abstract Video coding is a process for adapting media content to the constraints of transmission networks delivery and terminal device visualization. Moreover, content protection is also necessary. Nowadays the heterogeneity of client devices is increasing leading to different resolutions, qualities and form factors. Due to this, transcoding and protection are essential processes to be conducted in modern video distribution networks to adapt video to devices and network constraints and to enable pay per quality schemas enforcing content licenses. Unfortunately, transcoding and protection can be no longer considered linear since every single content should be transcoded in several formats and sometimes protected, so it would require a long time to finish. Modern scalable coding techniques, as H264 SVC, can help to save processing power and bandwidth providing in a single stream several video versions. However, if the enhancements of a SVC encoded content are protected separately, it would possible to enable pay-per-quality providing an additional degree of freedom to content delivery industry. Unfortunately, transcoding and protection entail huge doses of processing power at provider side and should be distributed. Moreover, processing key streams to decrypt enhancements that were encrypted separately can increase the complexity at receiver side. Cloud computing emerges as a potential solution for coping with large population of users with heterogeneous visualization devices. The elastic nature of cloud computing can be an advantage given the difficulty to predict the computing resources video content would require to be distributed during the entire content life. This article describes a system that distributes and parallelizes the video transcoding process as well as the content encryption, following the SaaS approach in cloud computing. Moreover, the article describes an experimental approach for generating and processing a flexible key stream that would help to simplify key management at receiver side and would allow legacy receivers to consume SVC content with separate enhancement protection.

Keywords Transcoding · Content protection keyword · Cloud computing

D. Díaz-Sánchez (✉) · P. Arias · F. Almenarez · A. Marín
Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Spain
e-mail: dds@it.uc3m.es

P. Arias
e-mail: ariasp@it.uc3m.es

F. Almenarez
e-mail: florina@it.uc3m.es

A. Marín
e-mail: amarin@it.uc3m.es

R. Sánchez-Guerrero
Telefónica de España, Calle Gran Vía 28, Madrid, Spain
e-mail: Rosamaria.sanchezguerrero@telefonica.com

1 Introduction

Video coding is a process for efficient adaptation of media content for matching properties and constraints of transmission networks and terminal devices minimizing the quality loss due to the adaptation [22]. Traditionally, the uniformity of client devices, typically TV sets with similar aspect ratio and resolution, made the process of video coding pretty straightforward in comparison to today's situation. However nowadays landscape of client devices is increasing in variety leading to an scenario in which the plethora of media displays (target devices) is among the major challenges [3] in efficient video distribution over fixed [5] and mobile networks [13]. Different resolutions, qualities and form factors, require generating versions of the same content for fitting different devices. Moreover, multimedia transmission in mobile networks, that imposes variable bandwidth conditions [17], may impose additional requirements as time scalability. This process of versioning, called transcoding, entails huge doses of media processing that is computationally expensive. Fortunately, H264 scalable video coding (SVC) [15] has been developed by ITU-T and ISO/IEC to enable scalability of the video bitstream in the spatial, fidelity or temporal domain. SVC allows to filter the video bitstream reducing the complexity and saving bandwidth.

Traditionally, a media provider delivered a content over a single or several broadcast distribution networks using MPEG2TS, the content was then consumed by the client and the process finished. The advent of broadband networks and the availability of a return channel enables new services requiring a media provider not only to provide a single video version but many to fit several devices that would be eventually distributed through different service flavours as live streaming, video on demand [9], near video on demand and catch-up TV (that can be even web-based). Thus, the traditional content workflow is dramatically altered requiring more processing at the head-end.

Other aspect of video distribution systems is content protection. Commercial content must be protected by using conditional access (CAS) techniques during acquisition process. However, in a scenario where the content is provided in several qualities, content protection requires also high doses of processing power. Moreover, it is reasonable to charge a customer more for watching a high definition football match than for watching the same program with less quality. Protecting different video versions as well as SVC enhancements separately with a different key stream is a desirable feature for content distribution so they can offer the content in different qualities and prices. However it can add more complexity to the receiver as well as it would require more processing at provider head-end.

Due to this, transcoding and protection can be no longer considered linear since every single content should be transcoded in several formats and sometimes protected, so it would require a long time to finish. This article describes a system that distributes and parallelizes the video transcoding process as well as the content protection. The system analyzes the video input and splits the input in several pieces that can be transcoded individually, sorts the output of the transcoding process and merges the output considering scenarios in where the results will be immediately streamed over the network and others in which the results will be merged and stored for later usage (web-based catch-up).

This article tackles also SVC parallel encoding and separate enhancement protection. Finally, to facilitate key management at receiver side when using a different key stream per component in SVC, we will sketch out how the key streams can be provided as a graph in which every node represents a video quality and depends on the previous lower qualities using a Markov's state chain. In such a way, a receiver does not need to decode the entire graph, by inferring the probability of selecting one path or other, speeding up the decryption process.

The article is organized as follows. Section 2 introduces the related work, cloud computing concept and its relation with multimedia management, the the MapReduce model that our system uses as well as some background on content protection. Section 3 describes the objectives of our distributed transcoding system and its architecture. Section 4 sketches out the structure of the flexible key management system and the operation of the receiver. A prototype implementation and some experimental covering scalable and non scalable video results are presented in Sect. 5. Finally, the work is summarized in Sect. 6.

2 Background and related work

The transparency of the Internet design has allowed to join new networks to the Internet's network-of-networks model permitting also the deployment of new services and applications leading to the well known hourglass model around the IP protocol. Internet has developed as a critical infrastructure for our society and economy and plays an active role in the daily life of millions of people. The content types accessible trough Internet have diversified from Hypertext to many others. Hypertext is still among the most popular content types but multimedia content is increasing its presence very fast requiring more processing power [12] and new strategies to minimize unnecessary expenditures in bandwidth. For instance, video traffic over the Internet is growing by the 60 % every year [8]. In regard to the role, the Internet has evolved from an limited academic scope to a mass phenomenon taking more and more relevance in business and e-commerce since all processes have been significantly automated and improved.

Concerning the user experience, Internet has influenced the evolution of computing paradigms from the mainframe to grid computing being the popularization of personal computers (PCs) an important milestone. The PC represented a commodity hardware that enabled the execution of several general purpose applications locally at reasonable cost, so it started to be present at average home environments which prepared an adequate breeding ground for the Internet. Moreover, in the recent years we faced the popularization of service-oriented paradigms that have finally lead to the *cloud computing* approach that is envisaged to satisfy the constant demand of computing resource, data storage or software functionality. Cloud computing brings to the Internet-of-services a high degree of flexibility and scalability that the plethora of new client devices as smartphones, tablets, phablets and any other upcoming device would need.

2.1 Cloud computing

Cloud computing is considered the present to near future key-computing-paradigm by making software more attractive a service as well as offering scalability, reliability and availability [25].

Despite cloud computing is nowadays a reality, there are many voices pointing to the lack of an accepted definition for this computing paradigm [10]. It is based on the old concept of computing as a utility [20] but it has now emerged as a commercial reality, because of important drops in the hardware costs needed to build and operate large scale data centers [23] and the introduction of flexible business models that allow companies to start small and increase hardware resources only when there is an increase in their needs, permitting them to pay for use of computing resources when needed and release them when they are no longer useful [2] (scale-up and down). This factors made possible the emergence of new applications and services that would not have been so compelling in the absence of cloud computing.

The core concept behind cloud computing is Software as a Service (SaaS). Cloud computing, and its complexity, born from squeezing or generalizing the SaaS concept to exhaustion. According to this, if in SaaS an application can be a service, also does the environment over which the application is executed, and even the hardware that executes the entire software. Following this reasoning, cloud computing is a resource aggregation of applications, components and frameworks that can be configured for serving several purposes.

This generalization has ended up in three different service models [24] known as infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) being IaaS the most basic. PaaS is built on top of IaaS and SaaS over the latter. IaaS, the most basic model, offers basically processing power and storage typically in form of virtual machines requiring users to deploy complete operating system images.[1] PaaS provides a more elaborated playground consisting on a programming language and an execution platform.[2] Finally, SaaS provides functional applications to cloud users requiring no cloud management from client side.

2.2 Cloud computing in multimedia

Cloud computing has been identified as a paradigm that allows accessing and manipulating resources that are located in a different place to the client computer [21]. In general, it is necessary to analyze the cost/benefit ratio of moving large data sets to the cloud against the benefit of speeding up data processing since the costs of WAN have fallen more slowly than hardware and others [14]. Thus, applications in which the result of processing large data sets is smaller than the input data, require the data to be placed near the processing power (large data centers where the cloud infrastructure is located). On the contrary, applications in which the outcome is as large as the input should be analyzed in detail.

Media management usually involves operations over large data sets and the outcome is usually as large as the input data. For instance, popular public clouds provide media sharing services that require users' data (media) to be put in the cloud over wide area networks to be eventually downloaded from other clients also over the WAN, so the benefit of having data in the cloud in this case does not depend on the WAN cost or the processing power. It is just a matter of availability. Several commercial media distribution as Netflix or Ultraviolet allow users to stream purchased movies to devices through the cloud, regardless of where the content was purchased improving the user experience in content delivery [1]. However, they do not take into account other services available in the Internet in order to improve user experience. Other popular services are those managing user generated contents as pictures or videos. To cope with this, several cloud platforms have increased their presence providing a general purpose cloud. Some of them are web-based file hosting services offering storage and management of files across Internet through synchronization mechanisms as iCloud. So, they offer a mixture of SaaS and PaaS. More sophisticated approaches provide a personal cloud server which includes an UPnP-AV media server to stream media files on the local network, a web-based file browser and media player to manage and playback media files, as well as a web portal for remote access at home (inXtron). Moreover, within this approach users can create their own personal cloud and sharing files with their friends and family. However, the majority of these solutions do not provide to developers, the ability to create their own applications and exploit cloud computing functionalities. Similarly, as in the case of the presented solutions for content media distribution, the above personal cloud proposals lack a unified standards-based framework to be interoperable (data lock-in).

Other solutions that do not require availability as a key feature, explore to make the user equipment part of the cloud instead of hiring cloud services from third parties [6,7]. In such cases, the processing power is placed where the data is. This empowers a new cloud concept: private multitenant distributed cloud.

The definition of new versatile multimedia codecs, including scalable ones, as SVC, and enforced a new business model around transcoding. [3] sketches out "Split and

[1] Amazon EC2.

[2] Google App Engine, Microsoft Azure.

Merge", a cloud based platform that distributes the video encoding process in order to reduce video-encoding times. Moreover, several companies have flourished following the same business model: reducing the video encoding times providing subscriptions or pay-as-you-go pricing models. Some representative examples are fixcloud.com, encoding.com, Encodeo, zencoder.com, all of them offering cloud-based encoding services to third parties. Some works, as [16], concentrates on providing metrics around the quality of the encoding outcome. Nevertheless, the aforementioned examples of cloud based transcoding initiatives provide batch video processing and they do not deal with live streams transcoding or content protection. Parallel transcoding of video streams, on demand transcoding, triggered for instance by network congestion or content protection are tasks that should be conducted at provider head-end in a timely fashion. Moreover, our system deals with content protection.

2.3 Map reduce

Our system uses Hadoop,[3] an open source project from the Apache software foundation, to build a distributed transcoding system. Hadoop allows distributed processing of large data sets across clusters of commodity computers using a functional programming model known as MapReduce. Hadoop file system, known as Hadoop distributed file system (HDFS), manages the data knowing at any time the rack where a worker node is and replicates the data among several nodes. Thus, applications can use this information to run a process in the nodes that are persisting the data. In this way, the backbone traffic is reduced and also the impact of hardware errors.

A Hadoop cluster includes a master and multiple worker or slaves nodes. A simplified diagram of a Hadoop cluster is presented in Fig. 1. The master executes a JobTracker to which client applications submit MapReduce jobs and that pushes the work to available TaskTrackers pursuing to keep the work as close to the data as possible. TaskTrackers are executed by the master node and by worker nodes. A TaskTracker in a given node has several slots available so the JobTracker delivers work to TaskTracker nearest to the data that has an available slot.

Every node on Hadoop executes a DataNode that serves blocks of data over TCP/IP. The NameNode at the master persists metadata that will be used to locate data blocks across the DataNodes.

The programming model Hadoop employs is MapReduce [4]. In MapReduce the Map step basically takes the input and splits it in smaller tasks assigning a key and a value to every subtask and distributing them to worker nodes that will execute the map function over a portion of the data. In the reduce step, the framework sorts the output of the map step according to the key-value pairs and fed it to worker nodes that execute the reduce function to combine or merge the data. Figure 2 sketches out how Hadoop executes MapReduce jobs. The process is composed of several steps:

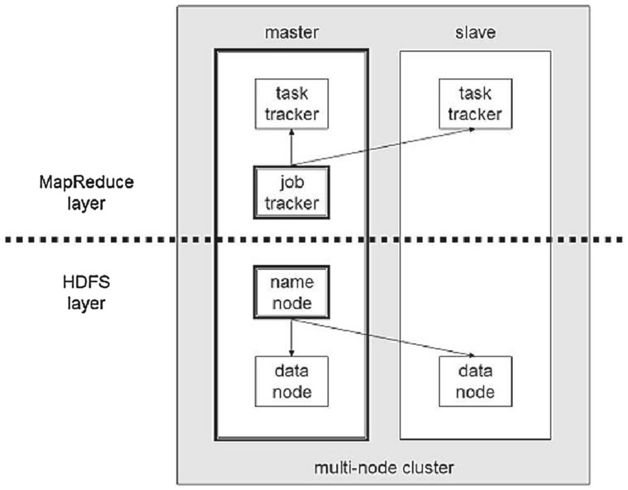


Fig. 1 Hadoop MapReduce layer and HDFS relation between master and slave nodes

- Configuring the job, that essentially requires to specify which parts of the program are in charge of splitting the input data into "InputSplits" or "Slices", that are small pieces of data over which a map function is executed,
- Assigning key-value pairs to the "InputSplits" to convert the byte-oriented view of the problem into a record-oriented view,
- Performing the mapping that executes an operation over an "InputSplit",
- Optionally sorting of map output
- Performing the reducing, that merges map output into other data sets
- Writing to the file system of the result.

Once the job is configured, and it has assigned a JobID from Hadoop, the framework copies the program binaries (Java Archives—JAR) to the distributed file system.

Then, the job is submitted to the JobTracker that initializes the job. The input data is split. Splits are persisted to the distributed file system and are assigned a key-value pair. Then the JobTracker assigns a map task per split to the worker node that has the split persisted in its DataNode and has an available slot. The worker nodes execute the map task. The master collects the results of the map execution and sorts them out as they are available. Once the master has enough data, it delivers the result to other workers to perform the reduce (it does not require all the map tasks to be finished) and the final writing to the file system.

[3] Apache Hadoop http://hadoop.apache.org.

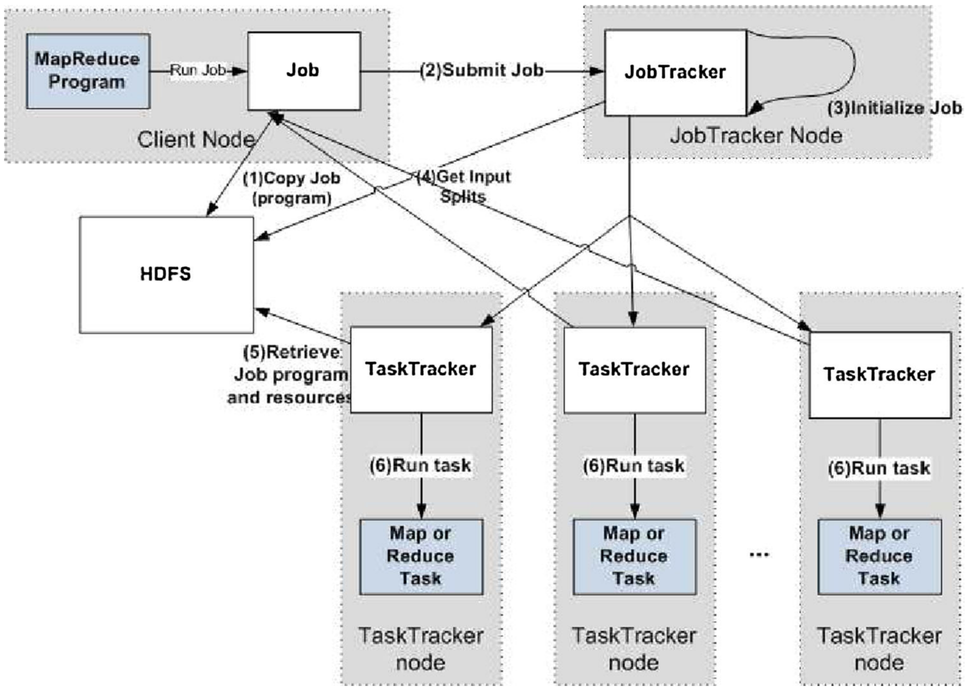


Fig. 2 Hadoop MapReduce job execution

2.4 Content protection

The acquisition process in PayTV is usually governed by conditional access systems (CAS). A very good example is DVB conditional access systems, that are nowadays widely deployed arround the world. DVB CAS are defined in three specifications: DVB CA (conditional access)[11], DVB CSA (common scrambling algorithm) and DVB CI (common interface). Conditional access relies on identification, authentication, authorization, key distribution, content encryption and rights expressions to overcome piracy. Identification, authentication and authorization are handled primarily by user equipment in cooperation with conditional access system hardware except for IPTV, that is a special case since there is a persistent two-way communication between the user equipment and the content originator and may use other means for performing such operations.

CASs can express complex rights using proprietary means, but those rights can only govern the decision of acquiring content or not. It is desirable to handle the acquisition process using some dedicated tamper proof hardware in the receiver. Nevertheless, some solutions implemented on software or rely on Trusted Platform Modules work appropriately in some contexts. The cornerstone of a PayTV system is the descrambler (decryption system) that might be integrated in the visualization device. Furthermore, key material to decrypt contents should be provided by a Conditional Access Module (CAM) also known as subscriber module. A CAM implements a proprietary key distribution protocol. It can be either a pluggable module (PCMCIA), a built-in module or a software implementation. Finally, a smart card or any other device is in charge of processing key streams inside EMM/ECM messages and fed the result to the descrambler.

Conditional access system key distribution involves many participants. From the source of the content to the user equipment every entity involved should be coordinated. Besides, rights expressions for content life-cycle management are responsibility of the content provider and should be always under the bound of the content.

2.4.1 Overview

Content providers protect audio and video using a combination of scrambling and encryption. Audio and video is scrambled with an unpredictable session key called control word (CW) that changes every crypto period (that use to be a short period of time). Beyond this basic encryption, there are several key distribution protocols with different levels and indirections that helps to distribute securelly the CWs to a big population of subscribers. Despite the operation of these key distribution mechanism varies among conditional access providers, it is typical to encrypt CWs with a Service key (SK), and distribute them using the key management system inside entitlement control messages (ECMs). ECMS are broadcasted together with the content. The descrambler needs the CW to decrypt the content. To prevent service interruptions, user equipment needs to be notified of ongoing changes in both CWs and SK.

SKs might change every few hours or minutes. Thus, providers use a type of message called entitlement manage-

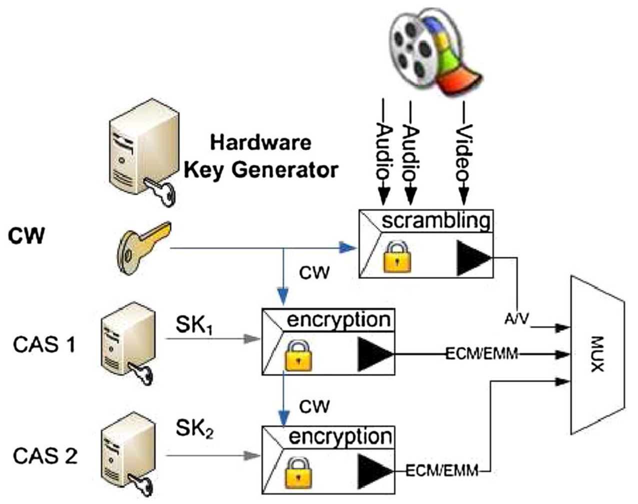


Fig. 3 Video and audio encryption. Audio and video are encrypted with a CW. The CW is distributed by different conditional access systems

ment message (EMM) to update SK. EMMs contain the SK and DRM information encrypted with the card key CK (a shared secret known only by the provider and the tamper-proof hardware); thus, EMMs can only be decrypted by a given smart card (though some CASs target a group of cards with a single EMM using a group key). Both ECMs and EMM are consumed by the CAM module and their format is private (defined by the CAS provider). ECMs are common to all subscribers but EMMs are specific to a subscriber. CA information is broadcasted using different streams (one per CAS) for a given content. Providers, using different CASs can use the same broadcast infrastructure (DVB SimulCrypt) saving bandwidth (see Fig. 3).

To convey the CW to a trusted descrambler, the service provider distributes a tamper-proof device, that decrypts EMMs and ECMs; but prior to send the CW to an unknown descrambler, it is identified by the smart card by sending the descrambler's serial number to the provider. The provider then checks the descrambler tamper proof hardware validity using a known manufacturer lists and sends the public key of the descrambler to the card. Next, the card generates a session key, and sends it to the descrambler encrypted with the descrambler's public key (SaC channel). The key is derived by using ElGamal authenticated key agreement (half-certified Diffie Hellman). The identity of the CAM is not checked since the misuse of a descrambler is not a security risk.

ECMs are received by the demodulator and sent by the CAM to the card for decryption. The card decrypts the ECM using the SK, and sends the CW to the descrambler through the SaC. To protect descrambled contents from being accessed once acquired, the decryption hardware, if not integrated in the visualization device, should export contents through a high-bandwidth digital content protection (HDCP, HDMI, GVIF) or a similar secure interface.

2.4.2 Header-content providers and CAS providers

The primary task of content provider is encryption or scrambling. common scrambling algorithm (CSA) is generally used by DVB and IPTV, other use digital encryption standard. IPTV reuses CSA since DVB IPTV just encapsulates MPEG-2 Transport Streams in IP, reusing thus, the traditional conditional access infrastructure and hardware. However, new scrambling algorithms, based in Advanced Encryption Standard (AES), are under development, as ATIS CSA, DVB-CSAv3 or DVB-CPCM Local Scrambler Algorithm. To ensure unpredictability, content providers use a 8 byte CW for scrambling (often hardware generated).

The content provider, as the first element in the chain, manages not only the scrambling but also the DVB content plane. Content providers must cooperate with different conditional access providers since data flow must incorporate conditional access information. Besides ECM and EMM, data flow must incorporate some tables with information such as where to find a specific ECM/EMM channel for a given CAS. Digital video broadcasting (DVB) specifications rely on MPEG-2 transport stream for video and audio coding and also for conveying important information, as key material, to the end users. MPEG-2 systems allow combining data streams with audio and video, being MPEG-2 transport streams (TS) the preferred multiplex stream for broadcasting over networks where errors occur. MPEG-2 defines a program as a set of packetized elementary streams (PES) containing audio, video and clock reference information. Those PES packets are fragmented and sent over broadcast networks as TS payloads with a fixed length of 188 bytes. All TS corresponding to a fragment of a given PES includes the same packet identifier (PID). Besides, transport stream includes also data tables describing the relationships between the set of PIDs and the programs. Those tables are called program specific information (PSI). There are three PSI tables related with CAS: the program association table (PAT), the program map table (PMT), and the conditional access table (CAT). The PAT is always broadcasted using PID 0, and contains one PID per program indicating where PMT can be found. The PMT contains all the PIDs related with a program (remember a PID identify PES with video, audio, clock and ECM of the program). Finally, the CAT points to the PIDs containing EMM PES for the different CAS (see Fig. 4). Encryption can occur at TS packet or PES level and only in one of those levels. The encryption is signalled by the scrambling control field, a header contained in TS packet or PES headers. There are two possible keys at each moment: even or odd.

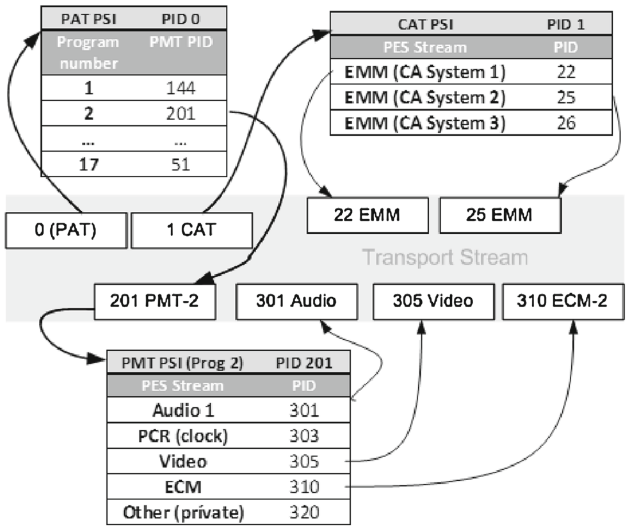


Fig. 4 Transport stream PSI tables and their relation with CAS

3 Cloud based distributed transcoding system

This section describes the internals of our transcoding system that distributes the transcoding process among several nodes in a data center providing several versions of a given content. Transcoding saves bandwidth but requires high processing power that's why the cloud and its flexibility are interesting for the scenario illustrated in Fig. 5.

The system also distributes video protection to minimize the impact of encrypting several video versions and distributing the key stream. The goals of our system are the following:

The first is to serve as a transcoding solution for both static inputs (batch processing of stored media) and live input. The batch processing of media files is pretty much simpler that live stream processing. The latter requires to split the input data as it arrives to the system requiring an adequate buffering.

The second goal is to allow the system to persist the result of the distributed transcoding to the file system, to stream it through the network or both. If the result is to be streamed, the distributed operation needs to be orchestrated: the result of processing input splits, adjacent to those which their transcoding result are ready to be streamed, arrive to the output module before the outgoing buffer under runs.

The third goal is the ability to change the transcoding format of part of the content or to provide additional formats during a transcoding operation. In this way, if the network conditions of a critical mass of client devices suddenly changes, the system could deliver a more appropriate format without interrupting the service and congesting the network. In that regard, the latest version of our software stack is able to produce scalable video using SVC if needed.

The fourth goal is to perform video protection in a distributed fashion being able to protect versions with different key streams enabling pay per quality systems.

Finally, the fifth goal is to allow the system to scale up and down automatically. If the system is required to deliver more formats during a given period of time, more processing nodes should be added to the system. On the contrary, if the demand decreases, idle nodes can be disconnected to save power.

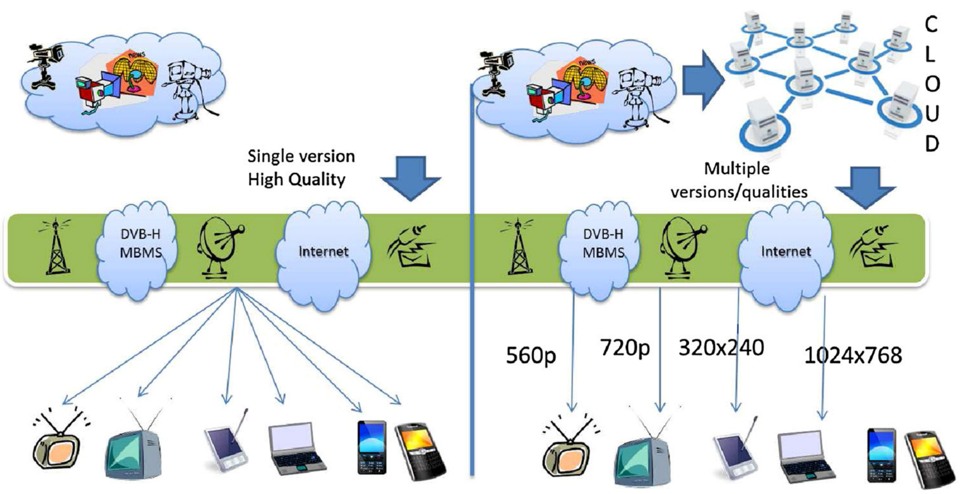
3.1 Architecture

The architecture of the system has several well differentiated modules that are the Input Splitter, the encoder, the Merger, the Wrapper, and the output module (see Figs. 6, 7).

3.1.1 Input splitter

The input module is in charge of receiving, buffering and persisting fragments of live streams, persisting input files for batch processing to the distributed file system and splitting

Fig. 5 Cloud based distributed transcoding system concept



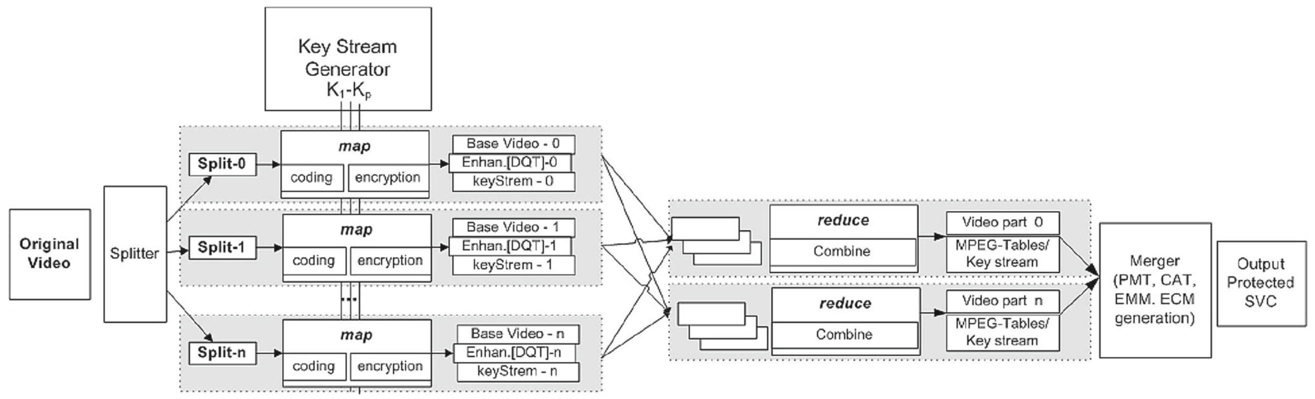


Fig. 6 Map reduce for distributed video coding and protection

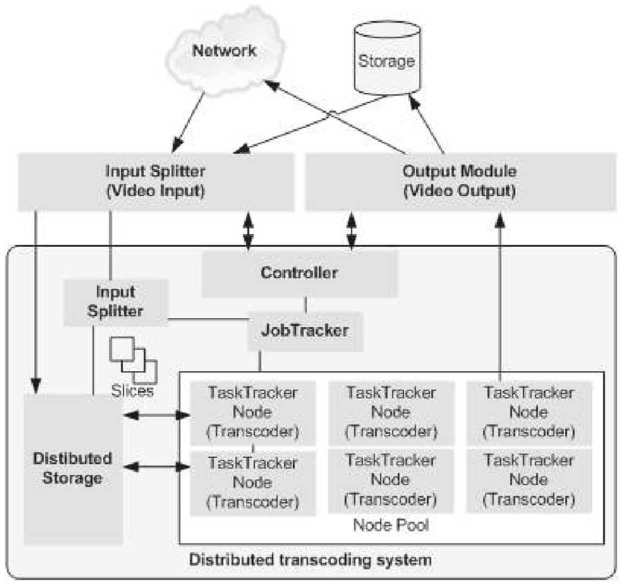

Network
Storage
Input Splitter (Video Input)
Output Module (Video Output)
Controller
Input Splitter
JobTracker
Slices
Distibuted Storage
TaskTracker Node (Transcoder)
Node Pool
Distributed transcoding system

Fig. 7 Architecture of the distributed transcoding and protection system

the input data into small independent pieces (called slices) to be processed in worker nodes.

This module performs one of the most critical parts of the system since the slice generation and distribution will influence the entire process. Despite Map Reduce architectures are adequate for processing large data sets, the nature of the information to be processed in this case (video) adds extra complexity to the system. Unlike other distributed operations in which the input data can be easily divided into split of the same size (i.e. lines of a text), multimedia content cannot be divided just splitting the content in its raw format, thus the overall encoding time will depend on the size of the splits. In other application domains, for instance processing a text, the splitting process is pretty straightforward and would lead to slices of a similar or equal size. Unfortunately, splitting encoded video is not that simple unless is fully decoded to YUV format and then split. Our system allows to split already encoded videos as well as YUV formatted ones. In the first case, the input module inspects the video streams, analyzes the group of pictures (GOP) and breaks the stream into slices from key frame to key frame to keep inter frame dependencies. Slicing is done so that it ensures the piece of video information contained in the slice can be processed separated from the rest of the video. In such a way, the slices can be considered independent and can be processed separately in different worker nodes. However the size of the slices depends on the video characteristics and would lead to different processing times in worker nodes making difficult to foresee the overall processing time in advance.

In the second case, the input data can be split in slices of similar size. Despite there may be some deviations in the slice processing time at worker nodes due to network congestion or work balancing, it would not necessarily be as uneven as in the first case.

Moreover, if the input data are obtained from a live stream and the keyframes are quite separated, it may be possible to spend significant time waiting for the next keystream until transcoding starts. Nevertheless, this approach provides the best transcoding results.

Once the slices are generated, they are persisted to the Hadoop distributed file system. After that, a record reader is invoked in order to assign a key-value pair (K_n, V_n) to each slice so the entire transcoding operation can be distributed among nodes in a very simple way, dealing just with key-value pairs. Once each InputSplit is correctly identified, the JobTracker invokes map functions in TaskTracker (worker) nodes. The map functions implements the encoder.

3.1.2 Encoder

The encoder is the core of the system. It implements the encoding routines that will take a slice, encode it according to preferences, encrypt it and encapsulate it into MPEG-TS. It is composed core is composed by the Hadoop JobTracker and TaskTracker nodes, the distributed file system and the Controller.

The controller is responsible of accepting new transcoding operations from the provider, creating new jobs and submitting them to the JobTracker. It also interacts with the Input Splitter and the Merger in order to receive input for the job and to deliver the result to the appropriate place.

The TaskTrackers are in charge of running map operations that involves the transcoding of slices upon JobTracker request. A map operation can involves the transcoding of an slice into one or several formats, the protection of the result using common scrambling algorithm when required and the encapsulation into MPEG-TS. Once each slice is correctly identified, the Hadoop JobTracker invokes map functions in TaskTracker nodes. The map function configure the video transcoder according to the requested output formats and encrypts them. The map receives several key streams from the key stream generator (a traversal module). Each key stream is a succession of 8 bytes unpredictable keys called control words (CW). The idea is to changes CWs frequently so breaking a EMM/ECM message to get a CW is worth nothing.

The JobTracker controls the Job execution and provides statistics to the Controller.

3.1.3 Merger

TaskTrackers also run reduce operations merging the results of the map operation into a single result. The reduce functions collect the results of the map functions and invoke the Merger that joins different MPEG-TS pieces into a single stream. This operation can be handled by one or several worker nodes and will depend also on the destination of the video (if persisted to storage results can be merged as they are available otherwise, if streamed, the process of merging results should be orchestrated by the job tracker to prevent buffer underrun problems)

3.1.4 Wrapper

This module is co-located with the Merger. It is in charge of generating and multiplexing different PSI tables as program clock references (PCR), program map tables (PMT) and program association tables (PAT) at fixed intervals according to the configuration. It also generates EMM and ECM messages according to the key distribution protocol that is described in Sect. 4. Finally, the Merger creates and adds the conditional access table (CAT) pointing to the EMM tables.

3.1.5 Output module

The output module takes the result of the transcoding operation and stores it in the file system or takes the results as they are available and streams them through the network. In order to fulfill the aforementioned goals the input and output module should be able to interact with file systems for batch transcoding operations as well as with the network to accept live streams as input and to stream the results of transcoding operations when necessary.

4 Scalable key management

In this section we present a flexible key management scheme with two purposes. We consider a content transmission scenario in with SVC provides a base layer and several enhancements in the spatial, temporal and fidelity range. The headend encrypts the base video and the enhancements using the same key stream but uses a different key stream per component (we call it enhanced key stream set) to perform post-scrambling obfuscation, so decrypting the high definition video would require having access to all the key material. Since the process of acquiring the content would be as complex as the complexity of acquiring and processing the multiple key steams used to encrypt the base video and the enhancements we can perform fine grained access control to different qualities.

Moreover, we purse to keep the system backwards compatible so the base video should be available for receivers not able to process the enhanced key stream set. Descramblers may be implemented in hardware (and there is only one in a system) and changing the control word may be expensive in time. If the descrambler is implemented in software, instantiating several descramblers may be also very expensive. For that reason, in some scenarios, would be necessary to use the same control word, and the same descrambler, for the base video and enhancements and resort to post-scrambling obfuscation to govern the access to enhancements. Finally, for compatibility, the system encapsulates the resulting video stream in a MPEG-2 TS and generates the appropriate conditional access tables and messages (EMM/ECMs).

4.1 Proposed key distribution schema

4.1.1 Video protection and key transmission

To illustrate the process, consider a video stream with a base video Q_b and three enhancements or combinations: Q_2 is a spatial enhancement over Q_b, Q_3 is a temporal enhancement over Q_b and Q_4 is the combination of Q_2 and Q_3 over Q_b. The base video Q_b is protected with a combination of scrambling and encryption using a single key called control word CW (CAS) that change every crypto period. To facilitate the enhancement decryption and to simplify the receiver,

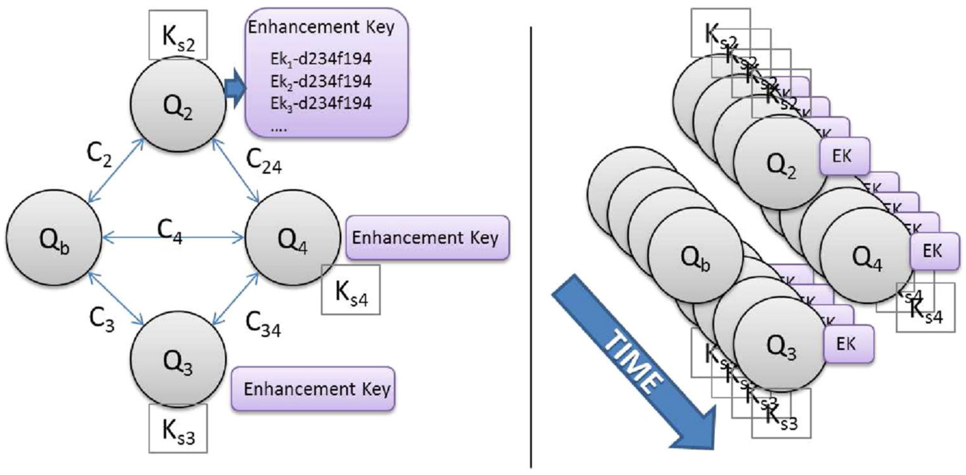


Fig. 8 Every quality enhancement has associated a key table with n randomly generated keys called enhancement keys (EKs). These tables change with the time

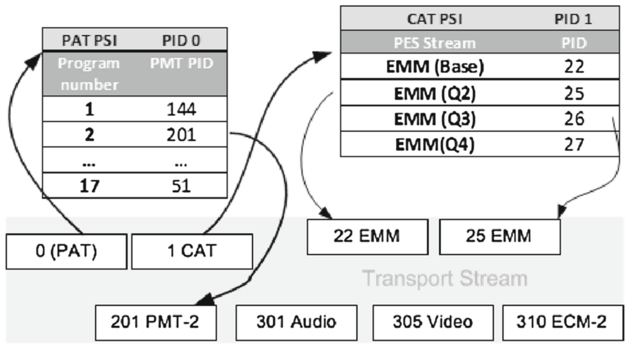


Fig. 9 Enhancement separate key stream transmission in MPEG-2 transport stream

our system can be used in devices with a single descrambler so enhancements, in that case, are protected with the same key as the base video.

To enable pay-per-quality and thus to control who access to a given quality in a per user basis, we rely in an indirection. Every quality enhancement i has associated a key table with n randomly generated keys $KQ_{i[1-n]}$ called enhancement keys (EK_s). Those tables are encrypted with a session key Ks_i that changes every session period and distributed to the receivers along with the video stream or by any other means. Figure 8 illustrates the key schema.

To protect enhancements from unauthorized access whereas coping with legacy descramblres and muxers and limited devices, they are scrambled using the same key (CW) as the base video and then video packets are transformed or obfuscated using logical operations with (EK_s). The operations can be XOR, AND, OR or combinations of many operations and keys from the E_K tables. The way to get the video packets of the enhancements from the obfuscated descrambled version is transmitted along with the video inside EMM/ECM packets (see Fig. 9). Table 1 shows an example of enhancement recovery.

Table 1 Example of enhancement obfuscation for Q2

Time	E_K items	Calc.
0	$E_{K(t=0)}[12]$	DOE $\oplus$ $E_K[12]$
1	$E_{K(t=1)}[2]$, $E_{K(t=1)}[11]$	(DOE $E_K[2]$) $\oplus$ $E_K[11]$
2	$E_{K(t=2)}[0]$, $E_{K(t=2)}[9]$	DOE $\oplus$ $E_K[0]$ $\oplus$ $E_K[9]$

DOE descrambled obfuscated enhancement

4.1.2 Video acquisition at receiver

The E_K graph allows quality transitions under some given conditions (shown in Fig. 8 as C_i). For instance, if the data link changes and data rates increase, C_2 and C_3 may be met. Consider the receiver's policy encourages an improvement in the spatial quality upon an improvement in the data rate. The receiver may request Q_2 to the mux (and eventually to the network if the content is streamed) and start descrambling the appropriate E_K tables if the data rate starts to increase preparing the player to improve the quality smoothly. Figure 10 illustrates the process.

Conditions (C_i) inform the receiver about the processing power, bandwidth and other constrains necessary to access a given quality. Alternatively, conditions may express information about the resulting quality leaving the responsibility of calculating the requirements to the receiver. We are currently working on a language to express that kind of constraints for multiple device architectures. In this way, the receiver can perform calculations and select the most appropriate quality in a given time leading to smooth video transitions that improve the quality of experience.

4.1.3 Video state and transition guessing

In this section we present a flexible way to extract the key stream based on Merkle hash trees generalized by Markov's state chains. Hence, our goal is to exploit the benefits offered

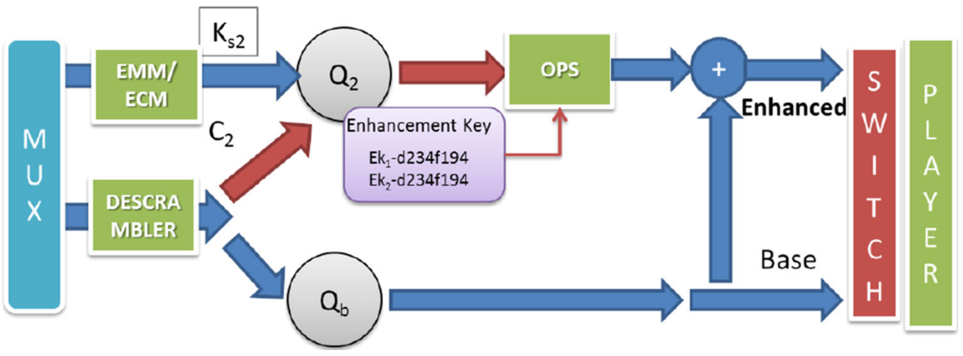


Fig. 10 The receiver decrypts the base and the enchancement and process the enhancement according to the obfuscation operations to get the enhanced layer. The switch will change the video stream when conditions are met

by Merkle hash trees [18] to distribute the media content in a secure, efficient and flexible manner.

A m-ary Merkle hash tree is a m-ary tree where each internal node holds the hash of the concatenated values of its at most m children nodes and the leaf nodes hold data, such as encoding keys for initial spatial and temporal resolutions (e.g K_{s1}, K_{t1}). The key schema presented before was illustrated by an example in which several video transitions and quality enhancements defined a set of states. Our system uses a m-ary hash tree generalized by a structure we will refer to a "hash DAG", based on a directed acyclic graph [19], which allows to store and distribute the different encoding keys depending on the quality of spatial and temporal resolution required by the user. Thus, the m-ary Merkle hash tree is mapped to a hash DAG.

To maintain consistency with m-ary Merkle hash tree, we define a vertex v to be a child of a vertex w if there is a directed edge from v to w, and w to be a parent of v if v is a child of w. Furthermore, we define a set of codification states, $CS = \{cs_{s1}, cs_{t1}, cs_{s1,s2} \dots\}$ as a hash DAG on an ordered vertex set V, where each vertex $v \in V$ represents a different encoding state and has one or two associated values xv (eg. x_1, x_2, $x_3 \dots$) and public parameters pv (eg. p_1, p_2, $p_3 \dots$) (see Fig. 11).

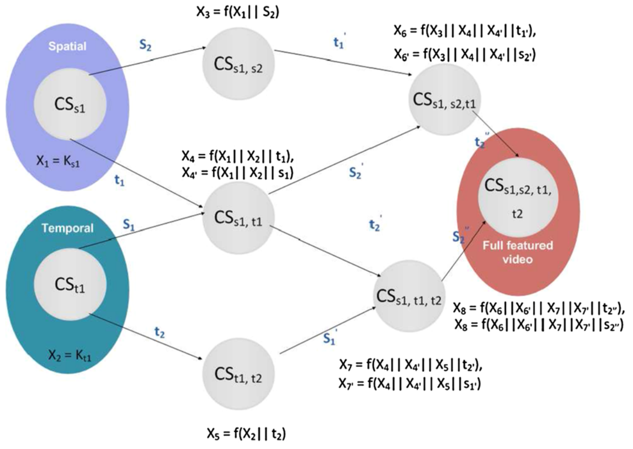

$X_3 = f(X_1 || S_2)$
Spatial
S_2
$CS_{s1,s2}$
t_1'
$X_6 = f(X_3 || X_4 || X_{4'} || t_{1'})$,
$X_{6'} = f(X_3 || X_4 || X_{4'} || s_{2'})$
CS_{s1}
$CS_{s1,s2,t1}$
$X_1 = K_{s1}$
t_1
$X_4 = f(X_1 || X_2 || t_1)$,
$X_{4'} = f(X_1 || X_2 || s_1)$
S_2'
t_2''
$CS_{s1,s2,t1,t2}$
Temporal
S_1
$CS_{s1,t1}$
t_2'
S_2''
Full featured video
CS_{t1}
$CS_{s1,t1,t2}$
$X_8 = f(X_6 || X_{6'} || X_7 || X_{7'} || t_{2''})$,
$X_8 = f(X_6 || X_{6'} || X_7 || X_{7'} || s_{2''})$
t_2
S_1'
$X_2 = K_{t1}$
$CS_{t1,t2}$
$X_7 = f(X_4 || X_{4'} || X_5 || t_{2'})$,
$X_{7'} = f(X_4 || X_{4'} || X_5 || s_{1'})$
$X_5 = f(X_2 || t_2)$

Fig. 11 Hash DAG for secure distribution of media cloud content with different spatial and temporal resolutions

Note that, the value(s) associated with each vertex is/are calculated using a hash function f and depends on its position in the graph. Moreover, each edge has a public parameter with is used to associate a codification state cs_i with cs_j. Finally, x_v and p_v satisfy $x_v \bigoplus p_v = x_v$. So, the content provider sent the key for decoding k_v, only if x_v can be computed.

Figure 11 illustrates a hash DAG with some possible states and associated hash values to obtain encoding keys for two different spatial and temporal resolutions (s_1, s_2 and t_1, t_2, respectively). The codification states (temporal, spatial or any combination) denoted as CS_{s1} and CS_{t1} hold the keys to obtain a spatial resolution s_1 and a temporal resolution t_1, respectively, that allows Alice to watch a video on her smart phone while going by train from work to home. Then, when Alice arrives at home, she desires to acquire a higher spatial resolution and the codification state which holds the hash value to obtain the corresponding enconding key is represented by $CS_{s1,s2}$. Thus, when she wants to transfer and reproduce the video from her smart phone to her TV, increasing the spatial resolution from s_1 to s_2, the receiver should be able to reach the state labeled as $CS_{s1,s2}$.

Therefore, to decode the video, the receiver has to able to calculate the hash values that compose the path from CS_{s1} to $CS_{s1,s2}$ by using the public parameters s_2 and p_3, in order to obtain the hash value x_3 and the corresponding key, k_3. Furthermore, it must noted that, representation of the set of encoding states through Markov's chains offers the advantage of the receiver to be able to estimate the transition probabilities between the different states. In this way, the proposed key scheme does not require the receiver to build the complete hash DAG to perform the content decryption reducing so the complexity of the receiver.

4.2 Operation

This section sketches out how a video is encoded and protected in a distributed fashion. The entire process is depicted in Fig. 6. The original video passes through the Input Splitter that searches for keyframes and divides the input in small pieces called InputSplits (from keyframe n to keyframe

$n+1$). The size of a split depends on the distribution of key frames in the original video and the underlying distributed file system but the minimum InputSplit size can be set to a very small value, for instance, the size can be set to 184 bytes when dealing with MPEG-2 packetized elementary streams (PES).

The splits are then persisted to Hadoop distributed file system and a record reader is invoked in order to assign a key-value pair (K_n, V_n) to each InputSplit. Once each InputSplit is correctly identified, the JobTracker invokes map functions in TaskTracker nodes.

The map function performs two internal tasks over the split according to the configuration. It first configure the video transcoder pipelines according to the requested output formats. Once the different output formats are calculated, the map function encrypts them. The map receives several key streams. Each key stream is a succession of 8 bytes unpredictable keys called control words (CW) that changes every crypto period.

The encryption algorithm used by default in our system is a combination of scrambling and encryption known as common scrambling algorithm (CSA). The output of the map functions is then fed to the reduce functions (that may execute in other nodes).

The reduce function joins the pieces returned by map functions together in a single video per output format. If an output format is requested to be encapsulated into MPEG-2 transport stream, the reduce function and the Merger generates MPEG-2 tables as program clock references (PCR), program map tables (PMT) and program association tables (PAT) at fixed intervals according to the configuration. It also generates EMM and ECM messages according to the key distribution protocol. Finally, the Merger creates and adds the conditional access table (CAT) pointing to the EMM tables. The content protection process for other video encapsulations is handled out-of-band.

5 Prototype implementation

The system has been implemented using Apache Haddop as a framework for distributing and balancing the load. The distributed application (see Fig. 12) that runs in Hadoop, in charge of splitting the data, performing the transcoding operations and merging the output are entirely written in Java.

The video encoding routines (encoder) lean on GStreamer for transcoding video except for AVC and SVC formats. GStreamer is a pipeline-based multimedia framework written in C that allows a programmer to create a variety of media-handling components, including simple audio playback, audio and video playback, recording, streaming and editing. GStreamer uses a plug-in architecture which makes the most of GStreamer's functionality implemented as shared libraries. Plug-in libraries get dynamically loaded to support a wide spectrum of codecs, container formats, input/output drivers and effects. The distributed application uses GStreamer through a Java Wrapper called gstreamer-java.[4]

The input and output modules also uses GStreamer through gstreamer-java to support live streams. The current prototype supports live streams in UDP, RTSP and RTP. Consumes MPEG2 transport streams and produces AVI (several codecs), OGV and MP4 outputs.

The encoder relies on Joint Scalable Video Model (JSVM) for AVC and SVC. The prototype performs content encryption using a modified version of the open source application VideoLan. VideoLan contains a software implementation of CSA scrambling algorithm. We added to VideoLan support for injecting ECM/EMM PES packets. The process of encapsulating the outcome into a MPEG-TS transport stream is handled with the aid of OpenCaster. Every external component to Hadoop is accessed using Java through Java Native Interface (JNI).

5.1 Non-scalable video results

This sections shows some experimental results obtained during the testing phase of the prototype. To demonstrate the flexibility of the system we performed linear (single node) and distributed (4 nodes) transcoding operations using the same node type (a single 2.40 GHz i7-core with 1 Gb RAM) on two different MPEG2TS videos.

The first video (VIDEO1—4s) is a 720 p (1,280 × 720) high definition MPEG1/2 video (mpgv) with 120 fps (frames per second) and 344 Kbps AC3 audio. The second video (VIDEO2—2 min 40 s) is a medium definition video (544 × 480) MPEG1/2 video (mpgv) with 60 fps and 160 Kbps MP3 audio.

The processing time for linear and distributed operation are shown in Table 2 (videos are transcoded to OGV and the results of the distributed transcoding operation are not merged into a single file).

The table shows the time spent by the linear transcoder (row 1), the time the Input Splitter employs in splitting the data (row 2), the sum of the map IO and transcoding operations times (as if the inputs were sequentially transcoded—row 3) and finally, the time spent by the slowest node to have the outcome available in a distributed operation (row 4).

As it can be seen, the distributed transcoding system employs less time to transcode several small pieces of content than linear transcoding. Moreover, since the process runs in parallel in several nodes (pipelining), the total transcoding

[4] https://code.google.com/p/gstreamer-java.

Fig. 12 Class diagram of the distributed system

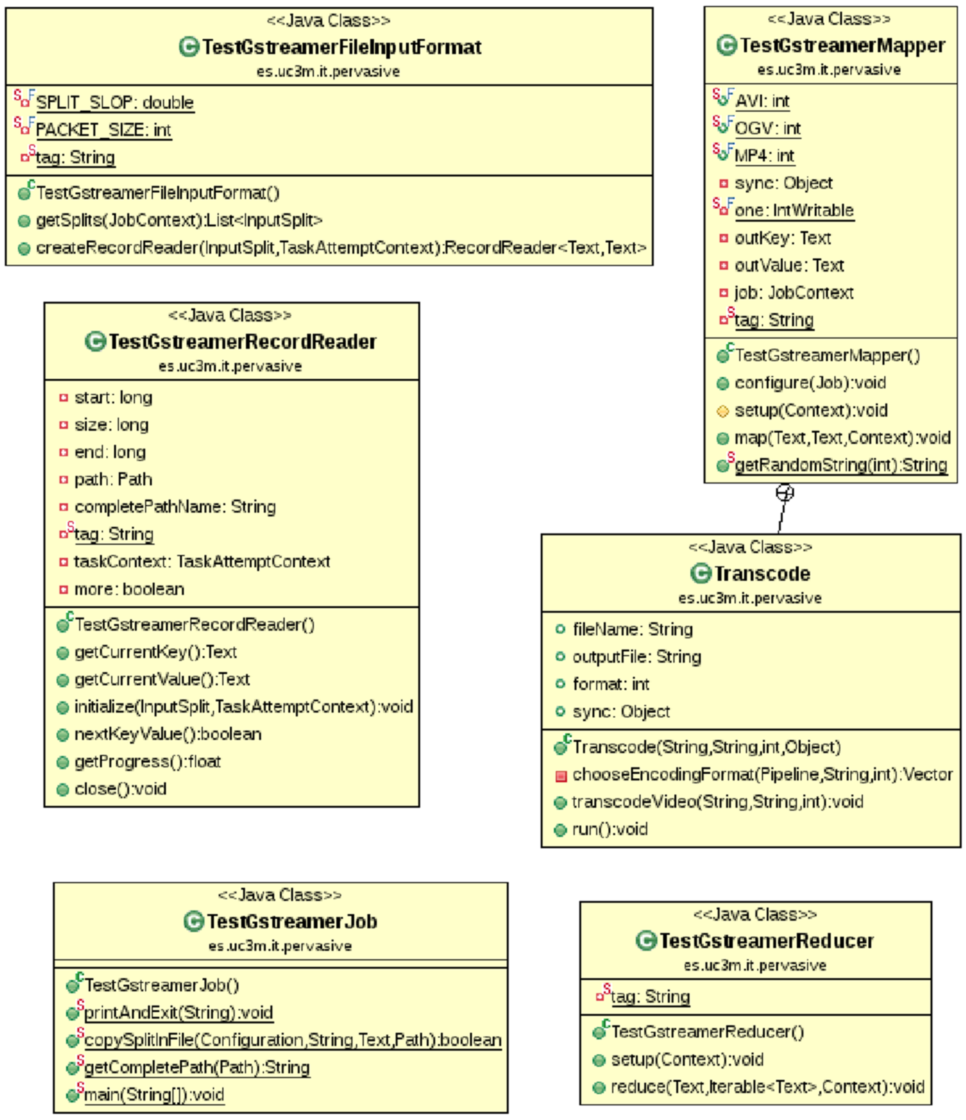


Table 2 Transcoding times

		VIDEO1 (ms)	VIDEO2 (ms)
1	Linear	108,872	547,627
2	Input formatting	251	329
3	IO and transcoding	52,722	417,764
4	Maximum time to finish	25,165	158,918
5	% Distributed versus linear	23.11 %	29.01 %

time is dramatically reduced (70 % less in the worst case—row 5 in Table 2).

Figure 13a, b and c show the resource consumption of the distributed transcoding operation on VIDEO1. In this case, the input is divided in three InputSplits so only three of the four available nodes are used.

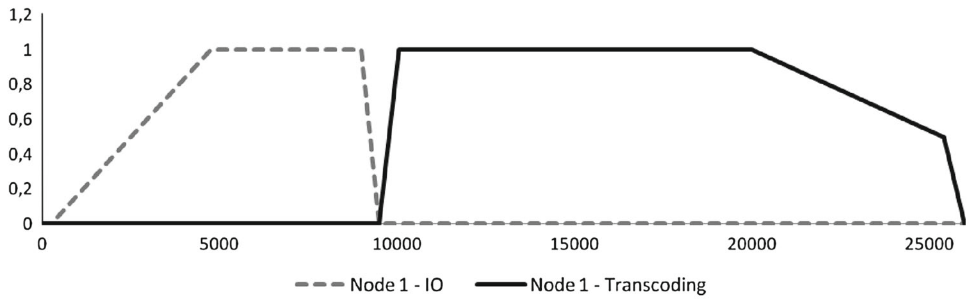
Figure 14a, b, c and d show the resource consumption of the distributed transcoding operation on VIDEO2 that has been divided in 14 InputSplits. Since each node has 4 execution slots, every node should deal with 3/4 InputSplits. If the number of nodes is increased, the overall transcoding time can be significantly reduced due to the pipelining.

5.2 Scalable video results

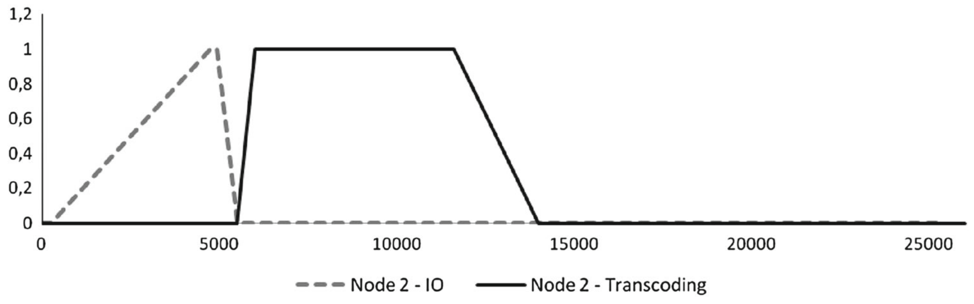
The system provides excellent results also for SVC despite, as discussed in Sect. 3.1.1, the performance of a given output format of depends also on how the input is split into slices (that also depends on the format of the input data). We have conducted some test whose results are shown in Tables 3 and 4.

In this case, the playground consisted on a Hadoop cluster with three nodes: one master/worker and two worker nodes

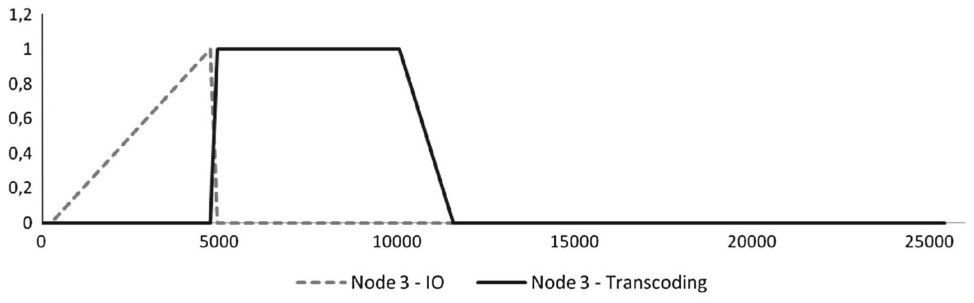
Fig. 13 Node resources usage during transcoding time (ms) for VIDEO1



(a) Resource consumption transcoding an InputSlice 49% of total input



(b) Resource consumption transcoding an InputSlice 26% of total input



(c) Resource consumption transcoding an InputSlice 25% of total input

(Core i5 2.66 GH, 4 cores per node) . Each node can accommodate up to four operations, so every node would deal with four slices. Table 3 shows the time employed in coding using H264.AVC, encrypting and wrapping (MPEG-TS) the well known CITY (YUV) test video in 4CIF at 60 fps keeping the original resolution and frame rate.

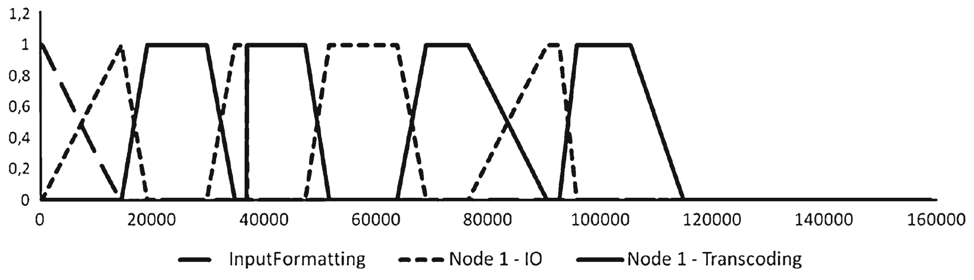
Table 4 shows the time consumed in coding using H264.SVC with SNR scalability. The resulting video contains an AVC base layer and two enhancements. The original video is the same used in the AVC tests. In both cases, the Input Splitter generated ten slices of 32 frames of YUV encoded video. The two worker nodes dealt with four slices each (from Slice 01 to 04 and 05 to 08) and the master node processed the remaining parts (09–10).

Despite SVC protection requires more processing time (each enhancement should be encrypted separately) in both cases there is a significant time reduction: in AVC the linear encoder takes 1,391,356 ms whereas the distributed encoder employs 955,909 ms (68 % of the linear encoder time); in SVC, the linear encoder takes 1,282,156 ms whereas the distributed encoder employs 843,307 ms (65 % of the linear encoder time).

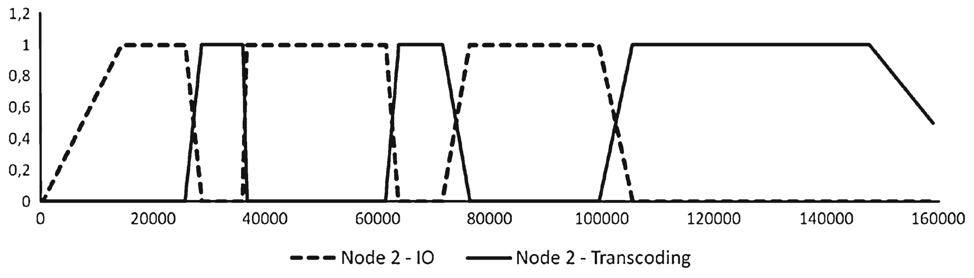
6 Conclusion

This article describes a complete video transcoding system that distributes the load among several nodes in a cluster.

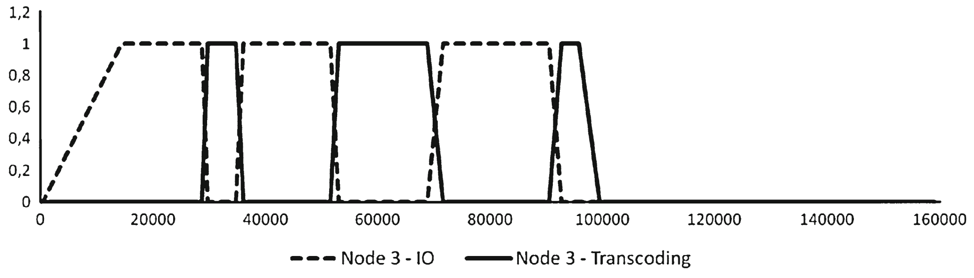
Fig. 14 Node resources usage during transcoding time (ms) for VIDEO2



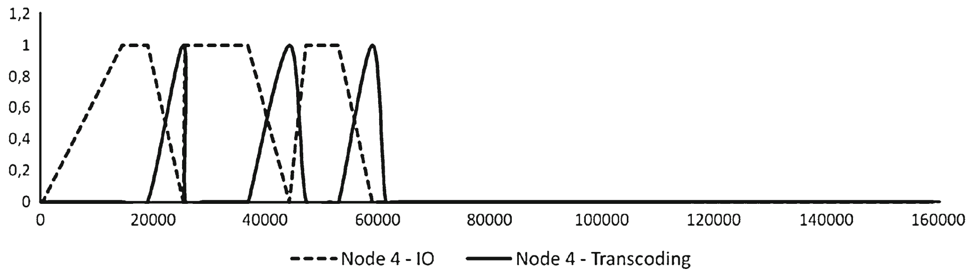
(a) Resource consumption transcoding 4 InputSlices



(b) Resource consumption transcoding 4 InputSlices



(c) Resource consumption transcoding 3 InputSlices



(d) Resource consumption transcoding 3 InputSlices

The distributed content encoding and protection at provider's head-end employs a distributed architecture that scales better than linear encoders and content protection systems, it enables pay-per-quality systems in a flexible way, as well as it provides multiple output formats in a cost effective manner.

The cost effectiveness is achieved by sharing resources that could be underused in other cases. In few words, our system, serves as a transcoding solution for both static input (batch processing of stored media) and live input, persists the result of the distributed transcoding to file systems, or streams the through the network, is able to change the transcoding format of part of the content or to provide additional formats during a transcoding operation and allows to scale up and down dynamically depending on the needs. The entire system has been implemented and tested and some experimental results are provided demonstrating the benefits of distributed transcoding systems over linear transcoding systems.

Moreover, it proposes a distributed encryption and a flexible key management that facilitates content filtering, key extraction and content decryption at receiver side. The proposed a protection system for SVC that protects enhancements separately using conditional access system

Table 3 AVC distributed encoding and protection (ms)

AVC	Coding	Encrypt	Wrap	Total	Real time
Splitting				513,316	513,316
Split 01	128,410	2,311	1,156	131,877	
Split 02	139,736	2,236	1,118	143,090	
Split 03	128,336	2,567	1,283	132,186	
Split 04	131,598	1,579	790	133,967	143,090
Split 05	133,871	1,606	803	136,281	
Split 06	126,737	2,535	1,267	130,539	
Split 07	135,904	1,631	815	138,350	
Split 08	144,295	2,020	1,010	147,325	147,325
Split 09	147,746	2,955	1,477	152,178	
Split 10	132,858	2,657	1,329	136,844	152,178
Total time	1,349,491	22,097	11,049	1,895,953	955,909
Full video	1,350,831	24,315	12,157	1,387,303	1,391,356

Table 4 SVC (SNR 2 layers) distributed encoding and protection

SVC	Coding	Encrypt	Wrap	Total	Real time
Splitting				479,311	479,311
Split 01	116,987	4,913	1,228	123,129	
Split 02	114,130	4,793	1,198	120,122	
Split 03	113,624	6,817	1,704	122,146	
Split 04	112,925	5,420	1,355	119,701	123,129
Split 05	113,551	4,088	1,022	118,661	
Split 06	113,287	6,117	1,529	120,934	
Split 07	111,669	4,690	1,173	117,532	
Split 08	112,369	4,719	1,180	118,268	120,934
Split 09	113,951	4,786	1,196	119,933	
Split 10	112,154	4,038	1,009	117,201	119,933
Total time	1,134,647	50,383	12,596	1,676,937	843,307
Full Video	1,198,277	71,897	1,198,277	1,282,156	1,282,156

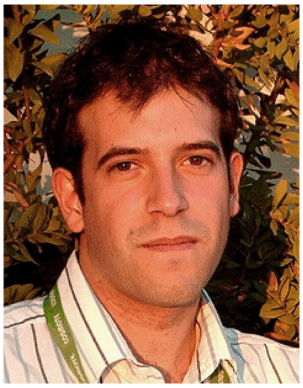
that is backwards compatible with existing systems and that requires a single physical descrambler or a single software instance of a descrambler to access the protected enhancements.

Acknowledgments The authors would like to thank Raul Varela Izquierdo for his contribution to this work. This work was partially founded by the Spanish Ministry of Science and Innovation under the project TEC2010-20572-C02-01 (CONSEQUENCE) and by the State of Madrid (Spain) under the Contract Number S2009/TIC-1650 (e-Madrid)

References

1. Agboma, F., & Liotta, A. (2012). Quality of experience management in mobile content delivery systems. *Telecommunication Systems*, *49*(1), 85–98. doi:10.1007/s11235-010-9355-6.
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., Zaharia, M. (2009). *Above the clouds: A berkeley view of cloud computing*. Technical Report. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009). URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
3. Breitman, K., Endler, M., Pereira, R., & Azambuja, M. (2010). When tv dies, will it go to the cloud? *Computer*, *43*(4), 81–83. doi:10.1109/MC.2010.118.
4. Dean, J., & Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113. doi:10.1145/1327452.1327492.
5. Develder, C., Lambert, P., Lancker, W., Moens, S., Walle, R., Nelis, J., et al. (2012). Delivering scalable video with qos to the home. *Telecommunication Systems*, *49*, 129–148. doi:10.1007/s11235-010-9358-3.
6. Diaz-Sanchez, D., Almenarez, F., Marin, A., Arias, P., Sanchez-Guerrero, R., Sanvido, F. (2011). A privacy aware media gateway for connecting private multimedia clouds to limited devices. In *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP*, (pp. 1–8). doi:10.1109/WMNC.2011.6097259.
7. Diaz-Sanchez, D., Almenarez, F., Marin, A., Proserpio, D., & Arias, P. (2011). Media cloud: An open cloud computing middleware for content management. *IEEE Transactions on Consumer Electronics*, *57*(2), 970–978. doi:10.1109/TCE.2011.5955247.
8. D.I.T.F. on the Future Internet Content (2009). Draft report of the task force on interdisciplinary research activities applicable to the future internet (2009). URL http://www.future-internet.eu. External Technical Experts: G. Camarillo, S. Dustdar, J. Magen, S. Paulus
9. Du, D., Liu, J., Hsieh, J., & Vetter, R. (1998). Building video-on-demand servers. *Telecommunication Systems*, *9*, 255–286. doi:10.1023/A:1019152024565.
10. Erdogmus, H. (2009). Cloud computing: Does nirvana hide behind the nebula? *IEEE Software*, *26*(2), 4–6. doi:10.1109/MS.2009.31.
11. ETSI (1996). *Digital video broadcasting (dvb); support for use of scrambling and conditional access (ca) within digital broadcasting systems*. Technical Report. ETR 289, ETSI.
12. Garcia, A., Kalva, H. (2011). Cloud transcoding for mobile video content delivery. In *IEEE International Conference on Consumer Electronics (ICCE), 2011* (pp. 379–380). doi:10.1109/ICCE.2011.5722637.
13. Garrido-Cantos, R., Cock, J., Martínez, J., Leuven, S., Garrido, A. (2011). Video transcoding for mobile digital television. *Telecommunication Systems* (pp. 1–12). doi:10.1007/s11235-011-9594-1.
14. Gray, J. (2008). Distributed computing economics. *Queue*, *6*(3), 63–68. doi:10.1145/1394127.1394131.
15. H.264 : Advanced video coding for generic audiovisual services. Technical Report H.264, ITU-T (2007).
16. Huang, Z., Mei, C., Li, L., Woo, T. (2011). Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy. In *Proceedings of IEEE INFOCOM, 2011* (pp. 201–205). doi:10.1109/INFCOM.2011.5935009.
17. Mardanian Dehkordi, A., Tabataba Vakili, V. (2011). An improved equation based rate adaptation scheme for video streaming over umts. *Telecommunication Systems* (pp. 1–13). doi:10.1007/s11235-011-9668-0.
18. Menezes, A., Vanstone, S. A. (eds.) (1991). Advances in Cryptology - CRYPTO '90, 10 th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11–15, 1990, *Proceedings, Lecture Notes in Computer Science*, Vol. 537. Springer.
19. Page, T. (2009). *The application of hash chains and hash structures to cryptography*. Technical Report: Royal Holloway, University of London.

20. Parkhill, D. F. (1966). *The challenge of the computer utility [by] D. F. Parkhill*. Boston: Addison-Wesley Pub. Co., Reading, Mass.
21. Velte, T., Velte, A., & Elsenpeter, R. (2010). *Cloud computing, a practical approach* (1st ed.). New York, NY: McGraw-Hill Inc.
22. Vetro, A., Christopoulos, C., & Sun, H. (2003). Video transcoding architectures and techniques: An overview. *IEEE Signal Processing Magazine*, *20*(2), 18–29. doi:10.1109/MSP.2003.1184336.
23. Voas, J., & Zhang, J. (2009). Cloud computing: New wine or just a new bottle? *IT Professional*, *11*(2), 15–17. doi:10.1109/MITP.2009.23.
24. Voorsluys, W., Broberg, J., & Buyya, R. (2011). *Introduction to cloud computing*. Hoboken, New Jersey: John Wiley & Sons Inc.
25. Weiss, A. (2007). Computing in the clouds. *netWorker*, *11*, 16–25.



Daniel Díaz-Sánchez obtained his Telecommunication Engineering Degree from Carlos III University of Madrid in 2003. He joined the Telematic Engineering Department in 2004 as a researcher cooperating with Pervasive Computing Laboratory team in some European Projects as Ubisec and Trust-es. He continued with his research activities while he prepared his MSc and Ph.D degree. He obtained the MSc degree in Telematics in 2006 and his Ph.D in 2008. Now he is an associate professor of the Telematic Engineering Department. In 2009 he was given an especial Ph.D award from Universidad Carlos III and the best Ph.D thesis award on electronic commerce from La Caixa as part of the awards promoted by the Official Telecommunication Engineering Association. Daniel is member of IEEE and co-author of approximately 50 international publications. Among those publications there are contributions to Computer Networks, Telecommunication System Journal and Transactions on Consumer Electronics. Besides, he contributed also to several conferences organized by IEEE, ACM and IFIP.

Rosa Sánchez-Guerrero received a Telecommunication Engineering degree from University Carlos III de Madrid in 2009 and she obtained the MSc degree in Telematics in 2011. In 2011, she was given the best bachelor thesis award on technologies for Banking and Finance from Sabadell Bank as part of the awards promoted by the Official Telecommunication Engineering Association. Currently, she works as researcher at the Department of Telematics Eng. in the Univ. Carlos III of Madrid, working within the Pervasive Computing research group. Her research topics include the problem of identity management, security and privacy in healthcare.

Patricia Arias Cabarcos receiwved her Telecommunication Engineering degree from University Carlos III of Madrid in 2008. She obtained the MSc degree in Telematics in 2009 from University Carlos III of Madrid and Politechnic University of Catalonia. Her bachelor thesis received the National award for best project on technologies for Banking and Finance, granted by the Professional Association of Telecommunication Engineers and Banesto. Currently, she is pursuing a Ph.D at the Department of Telematics Engineering in the Univ. Carlos III of Madrid, working within the Pervasive Computing research group. Her research focuses on the problem of identity management in open and dynamic environments, with special attention to risk analysis and the underlying trust models.

Florina Almenarez Mendoza received her Ph.D degree from the University Carlos III of Madrid (Spain) in 2006 and is currently an associate professor at UC3M. She received an award-winning as Magna Cum-Laude in her Computer Engineering degree. Her research interests include trust and risk management, identity federation, security and privacy in ubiquitous computing, and SIM-based applications. She leads the research activities of the PerLab group in advanced trust models for open and dynamic environments.

Andrés Marín Lópezis an assistant professor in the Telematics Department at the Carlos III University of Madrid. In 1997, he received a Ph.D Telematics Engineering from the Polythecnical University of Madrid, Spain. His research interests include pervasive computing, trust and security, mobile devices, and multimedia in future networks. He is author of more than 150 peer-reviewed papers for journals, conferences and workshops, and has participated in different international and national competitive financed projects. He is a member of the Pervasive Computing Laboratory (www.it.uc3m.es/pervasive) at the GAST group (www.gast.it.uc3m.es). Contact him at amarin@it.uc3m.es. Daniel Díaz Sánchez obtained his Telecommunication Engineering Degree from Carlos III University of Madrid in 2003. He joined the Telematic Engineering Department in 2004 as a researcher cooperating with Pervasive Computing Laboratory team in some European Projects as Ubisec and Trust-es. He continued with his research activities while he prepared his MSc and Ph.D degree. He obtained the MSc degree in Telematics in 2006 and his Ph.D in 2008. Now he is an associate professor of the Telematic Engineering Department. In 2009 he was given an especial Ph.D award from Universidad Carlos III and the best Ph.D thesis award on electronic commerce from La Caixa as part of the awards promoted by the Official Telecommunication Engineering Association. Daniel is member of IEEE and co-author of approximately 50 international publications. Among those publications there are contributions to Computer Networks, Telecommunication System Journal and Transactions on Consumer Electronics. Besides, he contributed also to several conferences organized by IEEE, ACM and IFIP.