Chapter 12
Collaborative Media Streaming Services Based on CDNs

Giancarlo Fortino, Carlo Mastroianni, and Wilma Russo

12.1 Introduction

In recent years, Content Delivery Networks (CDNs) have been demonstrated to be a highly efficient solution to provide media streaming services over the Internet ranging from TV broadcasts to video on-demand [1]. However, modern multimedia applications do not just perform retrieval or access operation on content but also create content, modify and manage content, and actively place content at appropriate locations to provide new, added-value services [2].

CDNs can be effectively used to support collaborative media streaming services and in particular, the collaborative playback service [4] which allows an explicitly-formed group of clients to request, watch, and control a streamed multimedia session in a shared way.

This chapter introduces a CDN-based architecture supporting the collaborative playback service which provides significant performance improvements from the point of view of media streaming delivery and control, with respect to the available centralized architectures supporting the collaborative playback service [4]. In particular, this chapter presents the Hierarchical COoperative COntrol Protocol (HCOCOP) which enables the shared media streaming control in collaborative playback sessions supported by CDNs. HCOCOP is mapped on the hierarchical control structure which is formed and supported by the CDN-based architecture when a collaborative playback session is set up. This hierarchical control structure is composed of a coordination server at the root level, one or more control servers at the intermediate level, and clients at the leaf level.

HCOCOP is implemented and evaluated through discrete-event simulation and in particular, the performance evaluation phase, which has involved symmetric and

Giancarlo Fortino
DEIS – Università della Calabria, Rende (CS), Italy, e-mail: g.fortino@unical.it

Carlo Mastroianni
ICAR-CNR (Italian National Research Council), Rende (CS), Italy, e-mail: mastroianni@icar.cnr.it

Wilma Russo
DEIS – Università della Calabria, Rende (CS), Italy, e-mail: russow@si.deis.unical.it

asymmetric topologies of the control structure and three different versions of HCO-COP (NoCoop, LocalCoop, and GlobalCoop), allows to analyze two significant performance indices (blocking probability and denial probability) which characterize the protocol performance.

The remainder of this chapter is organized as follows. We start with describing the architectures for providing collaborative playback services. Then we present an overview of the academic streaming CDN called COMODIN. We describe HCO-COP in Sect. 12.4, which is followed by its performance evaluation in Sect. 12.5. In Sect. 12.6, we describe two application domains enabled by a CDN-based cooperative playback system. Section 12.7 delineates future research directions. Finally, we conclude the chapter summarizing the main contributions.

12.2 Background and Related Work

The collaborative playback service enables an explicitly-formed synchronous group of users to select, watch and cooperatively control a remote media playback. Sessions supported by this service are named collaborative playback sessions (CPSs) [4].

In particular, a CPS includes three tightly correlated sessions:

- *Multimedia session*: A media playback in the form of a recorded audio/video presentation (e.g. a seminar), a movie, or a more complex synthetic multimedia object is synchronously transmitted to all members of the group to allow each member to watch it.
- *Control session*: The media playback is controlled by typical commands of a VCR (e.g. play, pause, seek, etc) that any member of the group can issue to change the state of the multimedia session.
- *Interaction session*: Group members can exchange messages among them for constructing and sharing knowledge on the basis of the content of the media playback.

An architecture supporting the collaborative playback service requires the following core services for organizing and running CPSs:

- *Group formation and management (GFM)*: The GFM service supports the formation and the management of collaborative groups. In particular, a group is formed around a media object selection made by the CPS group organizer and the explicit subscription of invited users.
- *Media streaming (MS)*: The MS service supports the streaming-based delivery of a selected media object to all the group members.
- *Streaming control (SC)*: The SC service allows the group members to control the multimedia session supported by the MS service.
- *Group interaction (GI)*: The GI service supports knowledge exchange among the group members through text-based messaging.

In particular, the SC service is based on a playback control protocol which allows for sending VCR-like control commands and handling the change of the CPS state when a control command affects the CPS. To regulate the activity of the group members in sending control commands the streaming control protocol has to employ coordination mechanisms. These mechanisms allow deciding which member of the group has the rights to send a control command (in case of floor-based coordination [3]) or which transmitted control command is accepted (in case of random-based coordination mechanisms [3]). Floor-based coordination relies on the concept of floor (or token) which must be explicitly acquired by a group member to send a control command. Conversely, according to random-based coordination, each group member can send a control command without requesting the floor so that contentions among control commands, simultaneously transmitted by different group members, can arise and a decision on which command should be accepted can be taken.

Moreover, the handling of the CPS state change is a crucial operation as it involves modifying the status of the playback and consistently propagating this modification to all group members.

In order to describe CPS control and state change handling, the reference Star-based abstract architecture shown in Fig. 12.1 is used. The components are: (i) the media streaming and control server (MCS), which incorporates the MS and SC services; (ii) the multicast communication channel (MCC) which supports the transmission of media streams and control commands; (iii) the collaborative client (CC) which interfaces a group member. The playback status of the CPS is managed by the MCS and changes when a control command is received. The automaton of the playback status, shown in Fig. 12.2, consists of two states (Playing and Paused) and related transitions labeled by the control commands (Play, Seek, Pause, Stop). Each

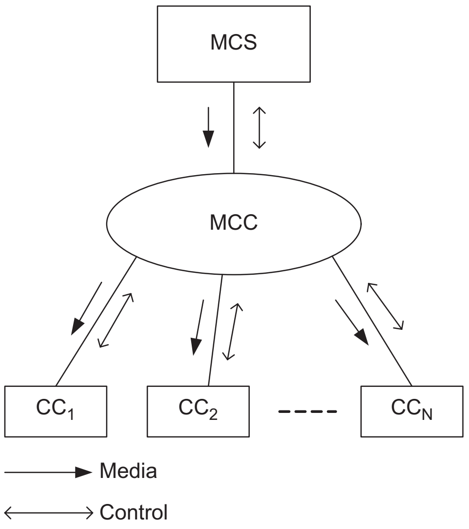


Fig. 12.1 Reference Star-based abstract architecture

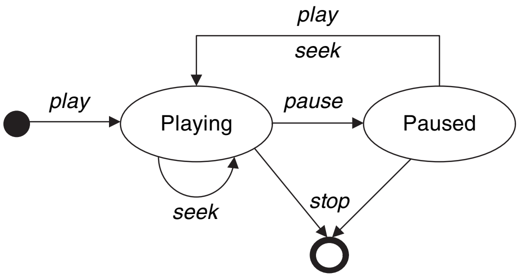


Fig. 12.2 Automaton of the CPS playback status

CC also contains an image of such automaton which must be kept updated consistently with the automaton of the MCS. Thus, when the MCS changes the playback status automaton, it sends an update message to all the clients so that they can change their automaton accordingly. During this update the MCS will not consider any other incoming control command.

To exemplify the interaction among MCS and CCs we use the time sequence diagrams shown in Fig. 12.3, in which we assume that the coordination mechanism is random-based. In particular, three scenarios are considered: (a) *without contention*, in which only CC_1 issues a control command; (b) *without contention and with command discard*, in which CC_1 issues *command*$_1$ and CC_2 issues *command*$_2$ which arrives after *command*$_1$; and (c) *with contention*, in which CC_1 and CC_2 issue a control command quasi-simultaneously. In case (a), CC_1 issues a control command and the MCS, after receiving the command, processes it, changes the playback status, and transmits the update message to all CCs. In case (b), the MCS during the control command processing and the CPS state update, rejects any other control command. In case (c), CC_1 and CC_2 issue two control commands which arrive at the MCS at the same time. The MCS must take a decision about which control command to accept so that it can discard the rest. Afterwards, the MCS behaves in the same manner as in case (a).

To date few systems have been designed and implemented to provide CPSs. Their architecture can be classified as Star-based architecture (see above) or CDN-based architecture.

The MBone VCR on Demand [9], the MASH Rover [13] and the ViCRO^C [4] systems rely on a Star-based architecture in which a single centralized server provides group organization, IP-multicast-based media streaming delivery, and streaming control. In particular, the media streaming delivery is based on IP-multicast for all systems. The media streaming control is based on IP-unicast for the MBone VCR on-Demand system, whereas IP-multicast is used for the other two systems. The streaming control protocols integrated in these systems use a random-based coordination mechanism, which resolves contentions by accepting the first incoming control command and discarding the others. The aforementioned systems experience two main issues: performance bottleneck represented by the centralized server and unfeasible deployment on the Internet due to the scarce availability of IP-multicast.

Furthermore a more recent streaming control protocol designed for a Star-based architecture is the COoperative COntrol Protocol (COCOP) [6]. COCOP relies on a

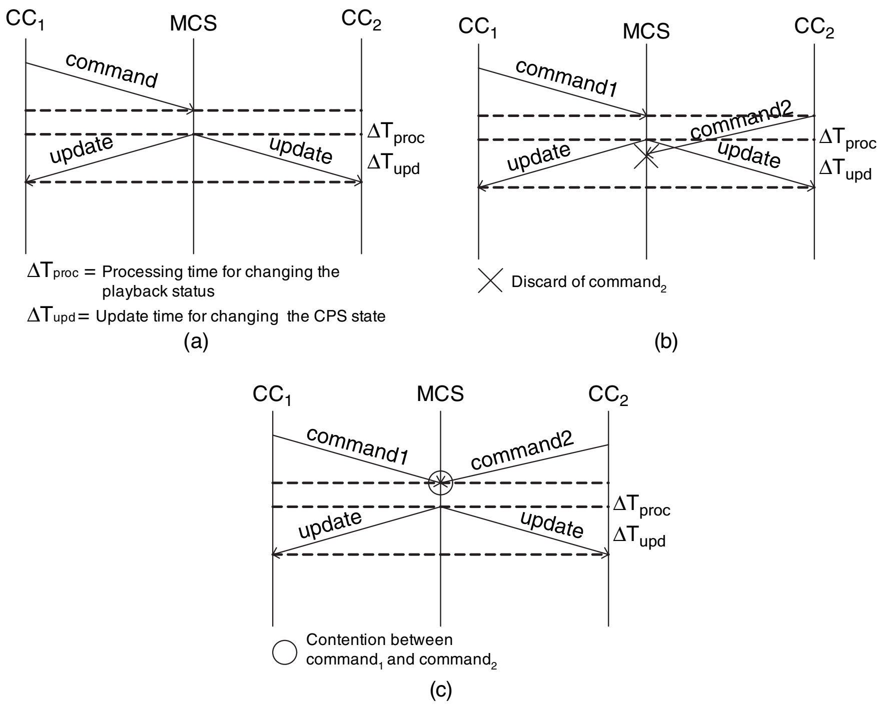

CC1
MCS
CC2
command
update
update
ΔTproc
ΔTupd
ΔTproc = Processing time for changing the playback status
ΔTupd = Update time for changing the CPS state
(a)
command1
command2
Discard of command2
(b)
Contention between command1 and command2
(c)

Fig. 12.3 Time sequence diagrams of the interactions between the MCS and two CCs: (**a**) no contention; (**b**) no contention and command discard; (**c**) contention

random-based mechanism similar to that of the aforementioned systems but it also introduces a cooperation mechanism according to which a group member avoids to send any control command when it senses that another group member has already issued a control command. It is demonstrated by Fortino et al. [6] that Cooperation greatly improves performance.

The COMODIN system [7] provides the same functionalities of the Star-based systems but relies on a CDN-based architecture. This approach not only allows overcoming the issues of the aforementioned systems but also increases efficiency of the media streaming control with respect to such systems.

12.3 An Overview of the COMODIN System

In this section we provide a brief overview of the COMODIN system. The architecture of the COMODIN system is organized into two planes (Fig. 12.4): the *Base* plane, which consists of a streaming CDN (SCDN) providing on-demand media streaming services, and the *Collaborative* plane, which provides the collaborative playback service.

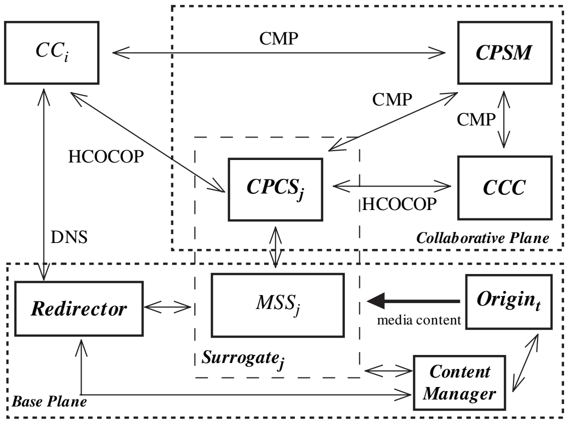


Fig. 12.4 The COMODIN architecture

The *Base* plane is composed of the following basic network components:

- The Origin, which archives the media objects to be distributed by the CDN.
- The Surrogate, which is a partial replica of the Origin with the additional ability to temporarily store content and deliver it to clients through the access network by using the Media Streaming Server (MSS) component.
- The Client, which is a multimedia application requesting specific media content made available through the CDN.
- The Redirector, which selects the most adequate Surrogate for each different client request on the basis of a redirection algorithm [10].
- The Content Manager, which coordinates the storage of media content between Surrogates and Origin servers.

The *Collaborative* plane consists of the following additional components to provide the collaborative playback service:

- The Collaborative Playback Session Manager (CPSM), which provides the group formation and management core service which is based on collaborative playback session management protocol (CMP). In particular, the CPSM allows for the formation, (un)subscription, initiation, joining/leaving, and termination of collaborative playback sessions (CPSs).
- The Collaborative Playback Control Server (CPCS), which is integrated with the MSS of the *Base* plane and supports the remote control of the media streaming shared among the members of a CPS.
- The CPCS Coordination Channel (CCC), which coordinates distributed CPCSs serving the same CPS through the coordination channel protocol (CCP).
- The Collaborative Client (CC), which is an enhancement of the Client component of the Base plane which interfaces the user with the collaborative playback service.

A CPS supported by the COMODIN architecture can be set up and run according to the following phases:

(1) *Organization*. An organizer CC connects to CPSM and requests the organization of a CPS.
(2) *Invitation*. The organizer CC invites other CCs to subscribe to the organized CPS by means of direct messaging.
(3) *Subscription*. Invited CCs connect to CPSM and subscribe to the CPS.
(4) *Initiation*. The organizer CC connects to CPSM, requests the initiation of the CPS, and the message is then redirected to a CPCS.
(5) *Join*. The CCs become CPS members through subscription and the message are then redirected to their respective CPCSs.
(6) *Execution*. The CPS is started by any member who issues the Play control request. A CPS's state changes by a sequence of successive control requests (Pause, Play, Seek). This phase, from the control point of view, is enabled by HCOCOP which is defined in the next section.
(7) *Termination*. The CPS can be terminated by its organizer CC by means of a voting mechanism.

An example CPS scenario featured by the COMODIN system and consisting of a group of four clients organized into two subgroups (A and B) of two clients attached to two different CPCSs (CPCS A and CPCS B), is shown in Fig. 12.5.

The numbers (1)–(7) identify the interaction scenarios (or message sequences exchanged between the active components) carried out in the aforementioned corresponding phases:

(1) The client belonging to the subgroup A ($CC_1{}^A$) organizes a CPS (hereafter called CPS_K).
(2) $CC_1{}^A$ invites three other clients ($CC_2{}^A$, $CC_1{}^B$, and $CC_2{}^B$) to subscribe to CPS_K.
(3) $CC_2{}^A$, $CC_1{}^B$, and $CC_2{}^B$ subscribe to CPS_K.
(4) $CC_1{}^A$ initiates CPS_K.

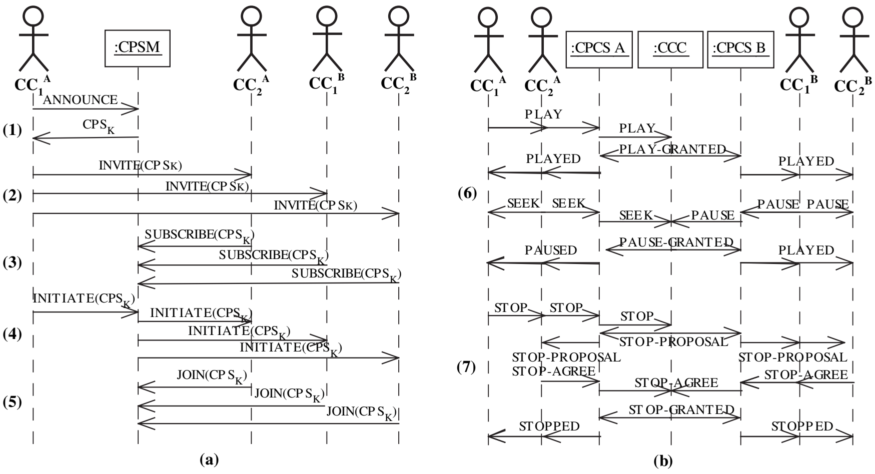


Fig. 12.5 A CPS scenario: (**a**) CPS set-up; (**b**) a running CPS

(5) CC_2^A, CC_1^B, and CC_2^B join CPS_K.
(6) CC_1^A starts the media playback. At a given time, CC_1^B requests a PAUSE and, quasi-simultaneously, CC_2^A requests a Seek; CC_1^B wins the competition because its command arrives before the other command.
(7) CC_1^A triggers a voting procedure to tear down CPS_K and CC_2^A, CC_1^B, and CC_2^B agree.

12.4 HCOCOP

The HCOCOP is an extension of the COCOP protocol [6] for CDN-based architectures. HCOCOP relies on the following characteristics:

- *Random-based mechanism for transmitting control commands*. A control command can be sent by any group member when he/she wishes to. The avoidance of explicit synchronization mechanisms (e.g. floor-based coordination) among group members increases interactivity even though contentions among issued control commands can arise.
- *FCFS policy for contention resolution*. If two or more control commands are quasi-simultaneously issued by different group members, the control command which will drive the CPS state change is chosen on the basis of an FCFS policy and the others are all discarded.
- *Cooperation-based mechanism to reduce the transmission rate of likely unsuccessful control commands*. A group member avoids to send a control command if it detects a control command issued by another group member. This mechanism lowers the number of contentions that can arise.
- *Soft state-based management of the CPS state*. Once a control command changes the playback status, the CPS state is updated by messages and timers without managing hard states.

HCOCOP is mapped onto the hierarchical control architecture of a CPS (hereafter called CPS_K) as shown in Fig. 12.6. It also shows where the automata (which define the protocol behavior) are located. The control structure components are derived from the architectural control components of the COMODIN collaborative plane when a CPS is executed: CCC_K, is the front-end of the CCC component for the CPS_K; $CPCS^K{}_i$ is the front-end of the i-th CPCS for the CPS_K; $C^{K,i}{}_x$ is the x-th collaborative client of the CPS_K served by the i-th CPCS front-end.

HCOCOP basically works as follows: if a client $C^{K,i}{}_x$ sends a control command (ClReq), its reference $CPCS^K{}_i$, before accepting it, forwards such ClReq to CCC_K to resolve possible conflicts which can be generated if clients attached to other $CPCS^K{}_w$ (with $w \neq i$) send a ClReq quasi-simultaneously. CCC_K accepts the first incoming ClReq, replies to all CPCSs, and discard other client requests for a given amount of time to regulate client interactivity and avoid session deadlocks. Possible conflicts generated by clients attached to the same $CPCS^K{}_i$ are instead resolved by $CPCS^K{}_i$ which adopts the same policy as the policy adopted by the CCC_K.

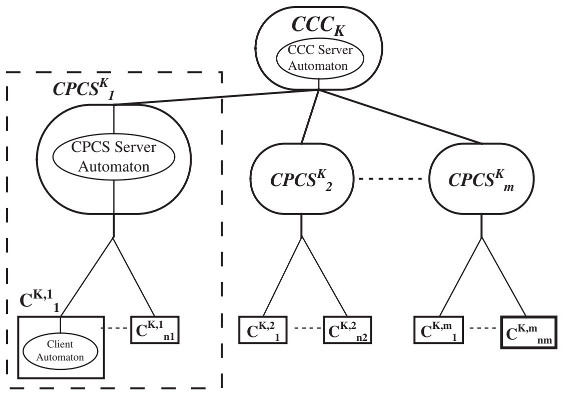


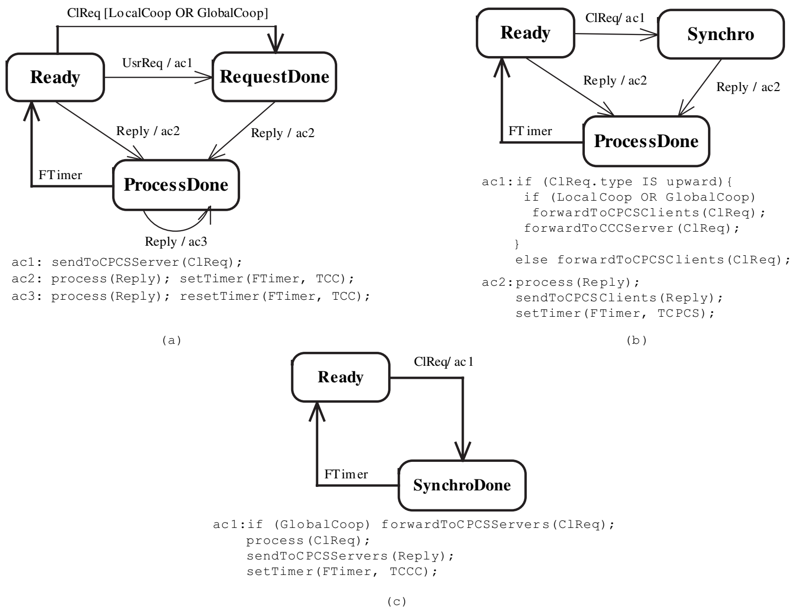
Fig. 12.6 The CDN-based control architecture of CPS_K

HCOCOP can operate under three cooperation modes:

- Global cooperation (*GlobalCoop*): the ClReq is forwarded downwards by the $CPCS^K{}_i$ to all its attached clients and by the CCC_K to all $CPCS^K{}_w$ ($w \neq i$) and then to all the attached clients. Such mechanism allows a client to detect a ClReq sent by other clients so as to refrain itself to send a ClReq which would be probably discarded.
- Local cooperation (*LocalCoop*): the ClReq is only forwarded downwards by the $CPCS^K{}_i$ to all its attached clients.
- No cooperation (*NoCoop*): the ClReq is not forwarded to any other client.

The automata defining the HCOCOP behavior are shown in Fig. 12.7. The *Client Automaton* (see Fig. 12.7a) of the client $C^{K,i}{}_x$ generates a client request (ClReq) when the user issues a control command (UsrReq) and enters into a Ready state. Then the request is sent to the $CPCS^K{}_i$, and it enters into the RequestDone state. This state is also entered when the client $C^{K,i}{}_x$ in the Ready state senses ClReqs sent by other clients attached to the same $CPCS^K{}_i$ (if *LocalCoop* is enabled) and also by other clients attached to $CPCS^K{}_w$ with $w \neq i$, if *GlobalCoop* is enabled. In the RequestDone state (in which the automaton remains until a Reply is received) additional ClReqs sent by other clients are ignored and the client $C^{K,i}{}_x$ is disabled from generating new control requests to limit the session load. It is processed after a Reply arrives. To control the interactivity degree of the session, new user control commands are blocked until a given time T_{CC} elapses.

The *CPCS Automaton* (see Fig. 12.7b) of the $CPCS^K{}_i$ can receive a ClReq while it is in the Ready state. Reception of a ClReq makes it enter into the Synchro state. If the ClReq comes from its attached clients, such ClReq (or *upward* ClReq) is forwarded to the CCC Server Automaton and, if local or global cooperation is enabled, it is also forwarded to its other attached clients. If the ClReq comes from the CCC (i.e. a ClReq originated by clients attached to other CPCS servers), such ClReq (or *downward* ClReq) is forwarded to all the attached clients. In Synchro or Ready states, upon receiving a Reply from the CCC Server Automaton, the CPCS Automaton processes the Reply and forwards it to all its attached clients. Afterwards



```
ac1: sendToCPCSServer(ClReq);
ac2: process(Reply); setTimer(FTimer, TCC);
ac3: process(Reply); resetTimer(FTimer, TCC);
```

```
ac1:if (ClReq.type IS upward){
     if (LocalCoop OR GlobalCoop)
      forwardToCPCSClients(ClReq);
     forwardToCCCServer(ClReq);
    }
    else forwardToCPCSClients(ClReq);
ac2:process(Reply);
    sendToCPCSClients(Reply);
    setTimer(FTimer, TCPCS);
```

```
ac1:if (GlobalCoop) forwardToCPCSServers(ClReq);
    process(ClReq);
    sendToCPCSServers(Reply);
    setTimer(FTimer, TCCC);
```

Fig. 12.7 Automata of the HCOCOP protocol: (**a**) client automaton; (**b**) CPCS automaton; (**c**) CCC server automaton

it enters the ProcessDone state wherein it rests until a given time T_{CPCS} elapses. Such delay is introduced both to make the clients aware of all changes in the session state, thus exploiting a soft-state like paradigm [12], and to regulate the group interactivity.

The *CCC Server Automaton* (see Fig. 12.7c), when receives a ClReq sent by the CPCS Automaton of $CPCS^K{}_i$ in the Ready state, accepts such ClReq and forwards it to all the other CPCS automata, if global cooperation is enabled. A Reply is then sent to all the CPCS automata and the CCC Server Automaton passes into the SynchroDone state wherein it rests until a given time T_{CCC} elapses. Such delay is introduced to assure the consistency of HCOCOP.

12.5 Simulation-Based Analysis of HCOCOP

This section is focused on the analysis of the HCOCOP performance in order to demonstrate the major benefits provided by the cooperation approach in a CDN-based architecture. In this regard, an object-oriented discrete event simulation framework [6] is exploited to implement HCOCOP and evaluate its performance in CDN-based architectures having different numbers of clients and different topologies. The HCOCOP performance is also compared to the performance of non-cooperative and cooperative protocols in a Star-based architecture to show that the use of a CDN can actually improve streaming control efficiency with respect to Star-based architectures.

12.5.1 Performance Indices

A cooperative playback control protocol must assure the consistency of the cooperative playback session (CPS) and, at the same time, must give users the ability to change the playback status. The definition of the performance indices, that refer to the relevant features which characterize a cooperative playback control protocol, takes into account (1) the handling of a user request for issuing a control command and (2) the handling of an issued control command; in particular, as discussed in Sect. 12.4:

(1) The Client Automaton enables or disables user requests that can therefore be forwarded as ClReq or blocked. In the ProcessDone state user requests are blocked to assure consistency of the CPS whereas in the RequestDone state user requests are blocked according to the cooperation mechanism to give priority to other already issued user requests.
(2) A non blocked user request is first forwarded as ClReq by the Client Automaton, to its reference CPCS server and, if accepted, it is then forwarded by this CPCS server to the CCC server which could accept it or not.

On this basis two performance indices are defined: the *blocking probability* (P_{BLK}) and the *denial probability* (P_{DEN}). The former is defined according to point (1) as the probability that a user request is *blocked* by the client process. The latter is defined according to point (2) since there is probability that a ClReq can be discarded (or *denied*) by the CDN. In particular, different denial probabilities are defined: (i) $P_{DEN}(CPCS)$, which is defined as the probability that a ClReq is discarded by the reference CPCS server; (ii) $P_{DEN}(CCC)$, which is defined as the probability that a ClReq is discarded by the CCC server; (iii) $P_{DEN}(CDN)$, which is also referred as the overall denial probability, is defined as the probability that a client request is discarded at either the *CPCS* Server or the *CCC* Server of the CDN, and is calculated as $P_{DEN}(CPCS) + (1 - P_{DEN}(CPCS))P_{DEN}(CCC)$.

The denial probability should be as low as possible since the server rejection of a client request is always considered a very unpleasant event for the user who generated the control command. In fact, although a user is completely aware that he/she is not always able to control the server, when a request is forwarded to the network, it is very likely that the user will expect to get his/her request accepted. The blocking probability should also be acceptably low since it characterizes the user inability to issue a control request. However, the denial probability is more critical than the blocking probability since users are generally more tolerant of the inability to send a control request than the rejection of a forwarded control request.

12.5.2 The Simulation Parameters

In this section we describe the parameters of the simulation framework and their setting in order to define a realistic simulation scenario which enables the evaluation of HCOCOP on CDN-based control architectures (Sect. 12.4).

Firstly, we consider more general aspects such as the duration of each simulation session and the degree of user activity; then, we focus on those parameters that allow us to characterize CDN-based control architectures (link delays, processing delays, and timers) and Star-based architectures used for comparison purposes.

12.5.2.1 General Aspect Parameters

For each simulation run the duration of the simulation session $T_{SESSION}$ is set to an amount of time that allows for deriving performance values of a pre-determined statistical relevance (i.e. with at least a 0.95 probability that the statistical error is below 5%).

The average inter-arrival time between two successive requests issued by the same user (User Activity) is characterized by the *Mean Request Interarrival Time* (MRIT) which is modeled according to a statistical model based on the *Gamma* probability distribution function [11]. In particular, User Activity is classified as very low (MRIT>=15m), low (10m<=MRIT<15m), medium (5m<=MRIT<10m), high (120s<=MRIT<5m) and very high (MRIT<120s). To enable the complete evaluation of HCOCOP in sessions with high to very high user activity, the value of MRIT was varied within the range {10 s, 180 s}.

12.5.2.2 CDN Parameters

The *delay* between two adjacent nodes (δ) is defined according to the following link delay model:

$$\delta_i = K_f\delta_m + N(K_v\delta_m, \sqrt{K_v\delta_m})$$
$$K_f + K_v = 1 \qquad K_f, K_v \geq 0$$

where δ_m is the mean delay and δ_i is the instantaneous delay for a given message. δ_i is the sum of a fixed part and a variable part, and the values of K_f and K_v are the relative weighs of the two parts, with K_f set to 0.7. The variable part of δ_i is generated by a normal random variable whose mean and variance are set to $K_v\delta_m$. The distribution of the normal variable is truncated at $-K_f\delta_m$ in order to assure that δ_i cannot assume negative values. The normal distribution is chosen according to the considerations presented by Gibbon et al. [8]. The parameters of the delay model are set according to the values measured in a CDN testbed established across Italy and Spain [7]. In particular, δ_m is set to 3 ms for the links between a client and its reference CPCS server, and to 61 ms for the links between a CPCS server and the

CCC server. For a fair comparison, δ_m between clients and the server is set to 64 ms in the considered Star-based architecture.

The *server processing delay* (T_{PROC}) is the amount of time taken by a CDN server (CPCS or CCC) or the Star server to serve an accepted request and accordingly change the state of the CPS. T_{PROC} is set to 200 ms.

The *server timers* (T_{CCC}, T_{CPCS}, T_{CC}) are used to control the reactivity of servers (T_{CCC} and T_{CPCS}) and the overall degree of system interactivity. They are both set to 3.0s, as is the client timer T_{CC}; this setting avoids deadlock situations, as shown by Fortino et al. [6].

12.5.3 Operational Modes of HCOCOP

In this section the *NoCoop*, *LocalCoop* and *GlobalCoop* operational modes defined in Sect. 12.4 are analyzed and compared. Moreover, the performances are compared with those obtainable with a Star-based architecture which exploits the COCOP protocol [6]. The Star-based architecture employed, hereby referred to as "Star", is representative of existing collaborative playback architectures which have a centralized nature, as control messages are processed by a single server entity (see Sect. 12.2). The COCOP protocol also operates in two different modes, *cooperative* (*Coop*) and *non-cooperative* (*NoCoop*), and is defined by two automata: the automaton of the COCOP client process, which is similar to the Client Automaton of HCOCOP (see Sect. 12.4), and the automaton of the COCOP server process which resembles the CPCS Automaton of HCOCOP but does not have the Synchro state since there is no need to synchronize with other servers. Moreover, the control protocols employed by the Star-based systems (MASH Rover and ViCRO^C, see Sect. 12.3) are similar to COCOP operating in the NoCoop mode that can be considered an archetypal implementation of those protocols and can be effectively used for comparison purposes.

12.5.4 Performance Evaluation

The simulation phase aims at evaluating the performance of the HCOCOP protocol in a simple CDN-based architecture with two subgroups and 12 clients, which is a quite large number for a cooperative playback session, since such sessions are mainly intended for small/medium sized groups of users [13].

We first present results achieved in a CDN-based architecture with a symmetric topology; then, we examine the behavior of HCOCOP in an asymmetric topology and in an adaptive scenario in which a client is dynamically redirected from one CPCS server to the other.

12.5.4.1 CDN with Symmetric Topology

A first set of simulations have been carried out in a symmetric CDN with 12 clients and 2 CPCS servers. Due the symmetry of this topology, 6 clients are assigned to each CPCS, as shown in Fig. 12.8.

Figure 12.9 shows the denial probability at the CPCS server, $P_{DEN}(CPCS)$. From the figure the benefits of the cooperation modes are evident. As described in Sect. 12.4, the *LocalCoop* mode disables users to issue control commands when the client process senses a request issued by another client attached to the same CPCS server. The use of this mode significantly decreases the denial probability with respect to the *NoCoop* mode. Benefits of cooperation are further enhanced under the *GlobalCoop* mode, since clients attached to a given CPCS server are also able to detect a request issued by clients attached to other CPCS servers.

Figure 12.10 shows that the denial probability at the CCC server, $P_{DEN}(CCC)$, is not appreciably modified by the cooperation approach. However, due to the improvement at the local group level, the values of overall denial probability, $P_{DEN}(CDN)$, are much lower when the global cooperation mode is exploited (Fig. 12.11). The denial probabilities experienced in the CDN and centralized (or Star) architectures are also compared in Fig. 12.11. Denial probabilities obtained in the CDN with *LocalCoop* and *NoCoop* modes are comparable with the denial probabilities achieved in the Star with the corresponding *Coop* and *NoCoop* modes. However, the denial probabilities obtained in the CDN with *GlobalCoop* are far lower than all other cases. Therefore, the use of CDN architectures, combined with cooperation mechanisms, can actually lead to a remarkable improvement in the ability of a client to control the server.

Even if the denial probability is the main performance index, it is important to verify if this improvement is obtained at the expense of the blocking probability. Figure 12.12 shows that the blocking probability is not significantly affected either by the cooperation mode (no cooperation, local or global cooperation) or by the type of architecture (Star or CDN).

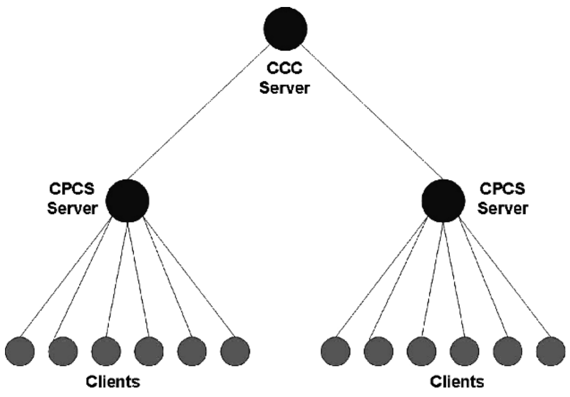


Fig. 12.8 Symmetric CDN architecture with 2 subgroups and 6 clients per subgroup

Fig. 12.9 Denial probability at the CPCS servers: comparison between NoCoop, LocalCoop and GlobalCoop operational modes

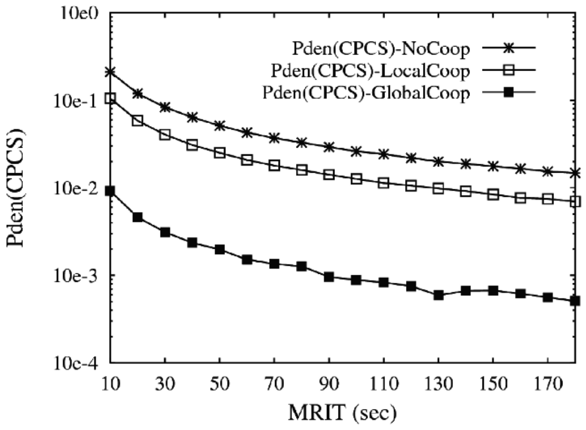
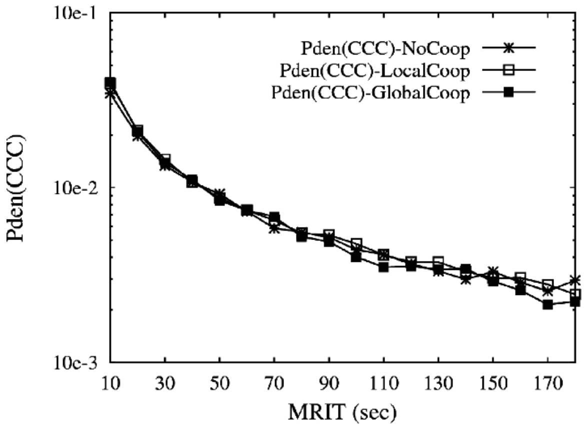


Fig. 12.10 Denial probability at the CCC server: comparison among NoCoop, LocalCoop and GlobalCoop operational modes



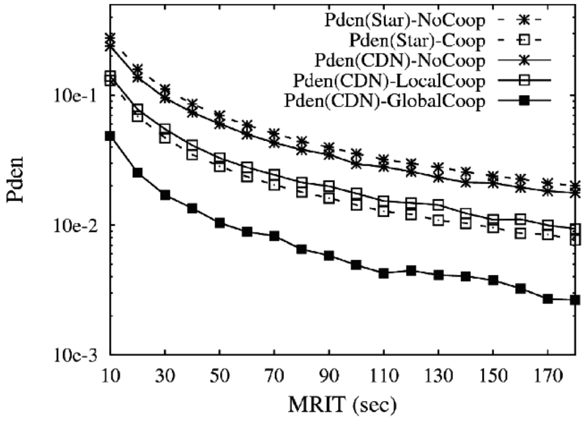


Fig. 12.11 Overall denial probability: comparison between CDN and Star-based architectures under cooperative and non cooperative operational modes

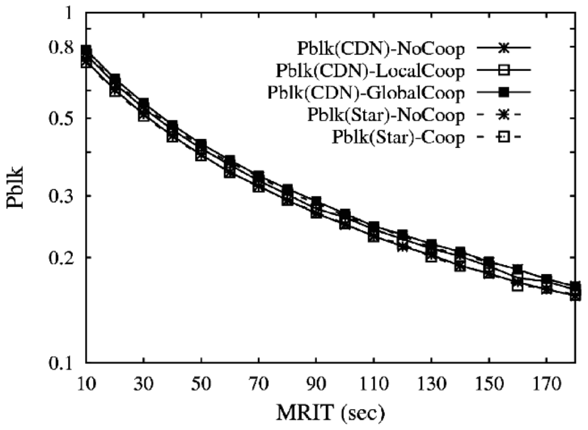


Fig. 12.12 Blocking probability: comparison between CDN and Star-based architectures under cooperative and non cooperative operational modes

12.5.4.2 Asymmetric CDN Topologies and Dynamic Client Redirection

A further set of simulation runs have been carried out to investigate the HCOCOP performance in a CDN architecture in which 12 clients are asymmetrically distributed among 2 CPCS servers (see Fig. 12.13). In particular, 7 clients are allocated to one server and 5 to the other. This topology may be obtained starting from the previously examined symmetric topology, in case that one of the clients is moved (or “redirected”) from one server to the other.

To better understand this phenomenon, it must be recalled that in a CDN a *request-routing* algorithm is employed to route a client request to an appropriate surrogate, which in our case corresponds to assigning the client to a specific CPCS server. In case of *adaptive request routing* [15], the surrogate can be dynamically changed according to CDN conditions, which in our case is under examination. The implications of this event are discussed in the following.

Figure 12.14 reports the overall denial probability experienced by the clients belonging to the two subgroups under cooperative and non cooperative modes.

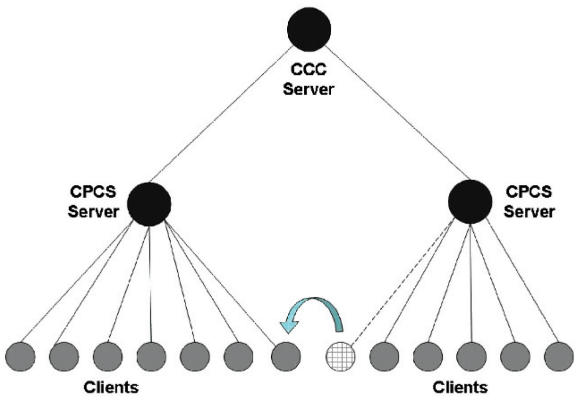


Fig. 12.13 Asymmetric CDN architecture with 2 subgroups, one with 7 clients and the other with 5 clients

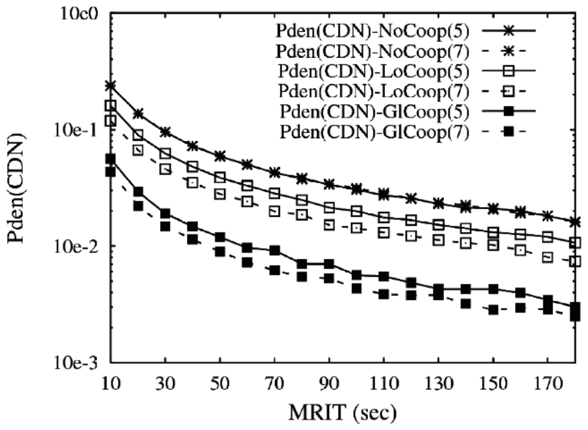


Fig. 12.14 Overall denial probability in an asymmetric CDN architecture: comparison among NoCoop, LocalCoop and GlobalCoop operational modes

Comparison shows that, with *NoCoop*, no difference in denial probability is found between the two subgroups. On the other hand, under cooperative modes, *LocalCoop* and *GlobalCoop*, the clients that belong to the most numerous subgroups have more chances to control the session state. This phenomenon can be considered a beneficial outcome of the cooperation mechanism; indeed the aggregation of clients in the same subgroup can improve the performance of all the participants of the subgroup. In particular, this phenomenon can be explained as follows. At the local level, the cooperative mechanism allows clients to perceive the requests that are generated by other clients. Therefore, as the number of clients in the same subgroup increases, it becomes easier to avoid issuing the requests that will be probably discarded at the local CPCS server. This benefit balances the drawback that comes from the fact that the level of local concurrency increases with the number of clients. On the other hand, once a client belonging to the larger subgroup gains the control of the local CPCS server, it has a higher chance of controlling the CCC server than a client belonging to the other subgroup. In fact the larger subgroup forwards a higher number of requests to the CCC server; therefore, these requests undergo a lower level of concurrency at the upper CDN level than the requests forwarded by the smaller subgroup. According to this outcome, clients can be profitably redirected to existing subgroups whereas isolated clients or clients belonging to very small subgroups can be penalized.

Moreover, no remarkable differences have been noticed between the blocking probabilities experienced by clients of the 2 subgroups.

Figure 12.15 focuses on the effect of the dynamic redirection of one client from a subgroup to the other, thus passing from a symmetric topology, with 6 clients per subgroup, to an asymmetric one, with 7 and 5 clients per subgroup. As a confirmation of the results shown in Fig. 12.14, the overall denial probability decreases in the subgroup to which the client is redirected and increases in the other subgroup, whereas the denial probability related to the symmetric topology is in the middle. This can also be seen the other way round: if the initial configuration is the asymmetric one, the redirection of a client can be performed to achieve a symmetric topology and this way obtain a better fairness among clients.

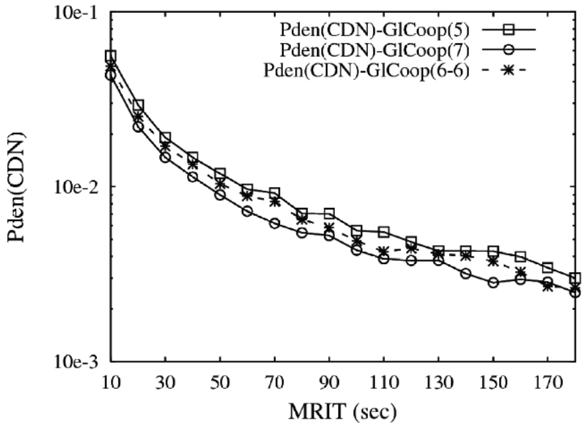


Fig. 12.15 Effect of client redirection on the overall denial probability: comparison between a symmetric CDN architecture and an asymmetric one resulting after a client redirection from one CPCS server to the other

As opposed to the denial probability, the blocking probability is hardly affected by client redirection.

In conclusion, the purpose of improving the fairness properties of the CDN architecture, with respect to denial probability, can be one of the rationales that drive the request routing algorithm, along with other usual parameters such as network proximity, client-server latency, and load of surrogates. The combination of such parameters is currently investigated with the purpose of defining a routing algorithm that improves not only data delivery, but also the effectiveness of the session control protocols.

12.6 Visionary Thoughts for Practitioners

The actual development and deployment of CPSs supported by CDN-based architectures provides the possibility to offer collaborative playback services on the current Internet infrastructure. It can also be enabled for several important application domains ranging from e-Learning to e-Entertainment.

A CDN-based CPS can efficiently support the Collaborative Learning on-Demand (CLoD) e-Learning paradigm [4], a virtual collaborative learning method which enables a self-tutored and interactive learning process where a small group of remotely dislocated students requests, watches, and controls a playback of a lecture and exchanges questions. CLoD borrows some of the ideas of the Tutored Video Instruction (TVI) and Distributed Tutored Video Instruction (DTVI) learning methodologies and tools [14] in which a small group of students driven by a tutor goes over a videotape of a lecture. DTVI is a fully virtual version of TVI, in which each student has a networked computer equipped with audio (microphone and headset) and video (camera) facilities to communicate within a group. TVI and DTVI have proven real effectiveness in that the students involved in their experimentation have been shown to outperform students who physically attended the lectures. The main difference between CLoD and DTVI is that CLoD does not assume the presence of a tutor which guides students to construct knowledge. In fact, while in DTVI only the tutor has control of the videoconference recorder (VCR), in CLoD each participant of the playback session uses a shared VCR remote controller in a sort of group-supervised collective tutoring.

CDN-based CPSs can also feature e-Entertainment applications such as the Virtual Theaters which are distributed virtual environments where people avatars (virtual alter egos of people) meet and plan to cooperatively watch and control a movie by exchanging comments or chatting with each others.

12.7 Future Research Directions

The CDN-based architecture proposed in this chapter is currently being enhanced to increase service effectiveness and efficiency. In particular, the defined HCOCOP currently does not differentiate among control commands; however associating different handling policies to different control commands can result in a more effective control of a cooperative playback session. A multi-policy playback control protocol for Star-based architectures has been proposed by Fortino et al. [5] where the authors have defined three policies (random-based, token-based and voting-based) and respectively associated them to the control commands Pause, Play/Seek and Stop according to their semantics. The handling of the Pause control command requires being highly interactive so that it can be effectively supported by the provided random-based policy of HCOCOP. The handling of the Play/Seek control commands can be supported by a token-based mechanism which allows the token holder to issue the control command. Finally, the handling of the Stop control command, as its acceptance would cause the CPS to be terminated, should be done according a majority criterion so that a voting-based policy can be effectively exploited.

The COMODIN system provides a best-effort media streaming synchronization among the group members of a CPS. Currently synchronization mechanisms at the CDN or at the client site are not offered, which would guarantee a synchronized view of the multimedia session to all clients of the group. Research efforts are under way to define a synchronization mechanism driven by the CDN which will provide more than best effort synchronization of the multimedia playback view without burdening the clients.

12.8 Conclusions

This chapter has presented a novel CDN-based architecture that supports collaborative media streaming services and allows an explicitly-formed synchronous group of users to select, watch, and cooperatively control a multimedia session. The control of the playback session is enabled by HCOCOP whose performance was evaluated through discrete event simulation. Results have shown that the hierarchical CDN-based approach is highly efficient, when compared with the usually adopted Start-based architecture, as denial probability is reduced while blocking probability is not significantly affected. Another interesting outcome is that in asymmetric topologies the clients that are assigned to more numerous groups are better served than isolated clients or clients belonging to very small subgroups. This phenomenon, if combined with other parameters such as network proximity, client-server latency, and load of surrogates can be exploited to tune the request routing algorithm, which is one of the major components of a CDN.

References

1. Cranor, C. D., Green, M., Kalmanek, C., Shur, D., Sibal, S., Sreenan, C. J., Van der Merwe, J. E. (2001) Enhanced Streaming Services in a Content Distribution Network. *IEEE Internet Computing*, 5(4):66–75.
2. Crowcroft, J., Handley, M., Wakeman, I. (1999) *Internetworking Multimedia*. Morgan Kaufmann Pub, San Francisco, USA.
3. Dommel, H. P., Garcia-Luna-Aceves, J. J. (1999) Group Coordination Support for synchronous Internet Collaboration. *IEEE Internet Computing*, 3(2):74–80.
4. Fortino, G., Nigro, L. (2003) Collaborative Learning on-Demand on the Internet MBone. In: Ghaoui C (ed) *Usability Evaluation of Online Learning Programs*. Idea Group Publishing, Hershey (PA), USA, pp 40–68.
5. Fortino, G., Mastroianni, C., Russo, W. (2004) A Multi-Policy, Cooperative Playback Control Protocol. In *Proc. of the 3rd IEEE Int'l Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, pp 297–302.
6. Fortino, G., Mastroianni, C., Russo, W. (2005) Cooperative Control of Multicast-based Streaming On-Demand Systems. *Future Generation Computer Systems, The International Journal of Grid Computing: Theory, Methods and Applications* 21(5):823–839.
7. Fortino, G., Russo, W., Mastroianni, C., Palau, C., Esteve, M. (2007) CDN-supported Collaborative Media Streaming Control. *IEEE Multimedia*, 14(2):60–71.
8. Gibbon, J. F., Little, T. D. C. (1996) Use of Network Delay Estimation for Multimedia Data Retrieval. *IEEE Journal on Selected Areas in Communications*, 14(7):1376–1387.
9. Holfelder, W. (1998) Interactive remote recording and playback of multicast videoconferences. *Computer Communications* 21(15):1285–1294.
10. Molina, B., Palau C. E., Esteve, M., Alonso, I., Ruiz, V. (2006) On Content Delivery Network Implementation. *Computer Communications*, 29(12):2396–2412.
11. Padhye, J., Kurose, J. (1999) Continuous Media Courseware Server: a Study of Client Interactions. *IEEE Internet Computing*, 3(2):65–72.
12. Raman, S., McCanne, S. (1999) A model, analysis, and protocol framework for soft state-based communication. *ACM SIGCOMM Computer Communication Review*, 29(4):15–25.
13. Schuett, A., Raman, S., Chawathe, Y., McCanne, S., Katz, R. (1998) A Soft State Protocol for Accessing Multimedia Archives. In *Proc. of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, pp. 29–39.
14. Sipusic, M. J., Pannoni, R. L., Smith, R.B., Dutra, J., Gibbons, J. F., Sutherland, W.R. (1999) Virtual collaborative learning: a comparison between face-to-face Tutored Video Instruction (TVI) and Distributed Tutored Video Instruction (DTVI). (Technical Report N. SMLI TR-99-72 by Sun Microsystems Laboratories, Palo Alto, CA, USA).
15. Wang, L., Pai, V., Petersen, L., (2002) The effectiveness of request redirection on CDN robustness. *ACM SIGOPS Operating Systems Review*, 36:345–360.