

A Hierarchical Control Protocol for Group-Oriented Playbacks supported by Content Distribution Networks

Giancarlo Fortino¹, Carlo Mastroianni², Wilma Russo¹

¹ DEIS – University of Calabria, Via P. Bucci, 87036 Rende (CS), Italy

² Italian National Research Council – ICAR Institute, Via P. Bucci, 87036 Rende (CS), Italy

phone: +39.0984.494063, fax: +39.0984.494713

e-mail: {g.fortino, w.russo}@unical.it, mastroianni@icar.cnr.it

Corresponding author: Giancarlo Fortino (g.fortino@unical.it)

Abstract - Content Distribution Networks have recently been introduced as a more efficient alternative to centralized servers for the delivery of static content as well as media streaming services ranging from TV broadcasts to video on-demand. Content Distribution Networks can also efficiently provide collaborative playback service which allows a synchronous group of users to select, simultaneously watch altogether and share the control of a multimedia session. This paper presents the definition and analysis of the Hierarchical Cooperative Control Protocol (HCOCOP), which allows a synchronous group of users to share the control of the media streaming session provided by a Content Distribution Network. The analysis phase, which was supported by discrete-event simulation, was carried out to characterize the efficiency of the protocol on the basis of the following defined performance indices: blocking probability, denial probability, server load and network load. The performances obtained for architectures based on Content Distribution Networks are compared with those obtained for centralized architectures providing collaborative playbacks. The comparison shows that HCOCOP significantly improves performance of media streaming control.

Keywords - Collaborative Control Protocols, Content Distribution Networks, Playback Services, Performance Evaluation

1. Introduction

Content Distribution Networks (CDNs) have recently been proposed to improve the performance (e.g. response times, bandwidth, and accessibility) of Internet-based content delivery through coordinated content replication (Pallis and Vakali, 2006). CDNs maintain geographically distributed clusters of surrogate servers which are placed at the network edge and store copies of identical content, so that users' requests can be satisfied by the optimal surrogates. CDNs have also been demonstrated to be a highly efficient solution to deliver media streaming services over the Internet ranging from TV broadcasts to video on-demand (Cahill and Sreenan, 2006; Cranor et al., 2001; Dutta and Schulzrinne, 2004).

CDNs can therefore be used to provide, more efficiently than centralized media streaming architectures, an added-value media streaming service named *collaborative playback service* (Esteve et al., 2007; Fortino and Nigro, 2000; Schuett et al., 1998). Such a service allows an explicitly-formed group of clients to request, simultaneously watch and share the control of a streamed multimedia session. To date, the collaborative playback service has been implemented in centralized architectures that rely on IP-multicast-based delivery technology (Fortino and Nigro, 2000; Schuett et al., 1998; Almeroth and Ammar, 1998; Holfelder, 1997). Such architectures suffer the following main drawbacks which limit the possibility of exploiting the collaborative playback as a mainstream media service on the Internet: (i) single point of failure and bottleneck arising at the centralized and network-core-located media streaming server, and (ii) use of the IP-multicast which is not yet widely deployed on the Internet.

In the COMODIN (COoperative Media On-Demand on the InterNet) project (Esteve et al., 2007) a reference CDN-based architecture capable of supporting group-oriented playbacks has recently been defined and implemented. Such an architecture provides the three fundamental core services of collaborative playbacks: (i) *group formation and management*, which allows for the explicit formation and management of a synchronous group of clients that can then jointly set up the streaming of a media object; (ii) *media streaming delivery*, which streams the selected media object to all clients of the group; and (iii) *media streaming control*, which allows the clients of the group to share the control of the media streaming through the transmission of control requests.

This paper focuses on the definition and performance evaluation of control protocols which enable interactive and shared control of group-oriented and CDN-based media playbacks. In particular, the paper proposes the Hierarchical COoperative COntrol Protocol (HCOCOP) which is an extension of the COoperative COntrol Protocol (COCOP) for CDN-based media steaming architectures (Fortino et al., 2005).

HCOCOP relies on the following characteristics: (i) random-based transmission policy of the streaming control requests; (ii) a cooperation-based mechanism to reduce the transmission rate of control requests which would likely be discarded; (iii) soft state-based management of the control session state (Schuett et al., 1998); (iv) FCFS-like policy for the acceptance of a control request. In particular, HCOCOP uses a random-based transmission protocol to increase client/CDN interactivity by avoiding the exploitation of explicit synchronization mechanisms (e.g. floor control (Dommel and Garcia-Luna-Aceves, 1999)) as these introduce additional delays in the session control. HCOCOP is mapped on the tree-based control structure which is formed and supported by a CDN reference architecture when a collaborative playback session is started. Such a control structure has three components which are hierarchically arranged into three levels (root, intermediate, leaf): one *control request coordinator* node at the root level, one or more *media server controller* nodes at the intermediate level and two or more *client* nodes at the leaf level. Each *client* node is attached to one *media server controller* node which is, in turn, attached to the *control request coordinator* node. When a *client* node is able to generate a control request, it sends such request to the related *media server controller* node, which discards it if another control request has been accepted, but otherwise coordinates with the other *media server controller* nodes through the *control request coordinator* node.

Three different operational modes of HCOCOP are defined: *LocalCoop*, *GlobalCoop*, and *NoCoop*. In *LocalCoop* the control request sent by a *client* node at the intermediate level is routed by the *media server controller* node down to all the other attached clients so that they can sense the control request and block the transmission of their own control requests as these would probably be discarded. In *GlobalCoop* the same mechanism is implemented at the root level: the control request routed by a *media server controller* node to the *control request coordinator* node is also routed down to all the other *media server controller* nodes and then to all the clients involved in the session. In *NoCoop* the routing-down mechanism of control requests is disabled.

HCOCOP was implemented and evaluated through event-based simulation. The performance evaluation

phase, which involves symmetric and asymmetric topologies of the control structure and the three different modes of HCOCOP, allows for the analysis of four significant performance indices which characterize protocol performance:

- 1) *blocking probability*, the probability that a user request is blocked by the *client* node;
- 2) *denial probability*, the probability that a control request is discarded either by the *media server controller* node or by the *control request coordinator* node;
- 3) *server load*, the number of protocol messages per second that are received by a *media server controller* node and by the *control request coordinator* node;
- 4) *network load*, the number of protocol messages per second that circulate in the hierarchical logical network.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 provides an overview of the CDN-based reference architecture for group-oriented playbacks. Section 4 describes, in detail, HCOCOP through state machines. In Section 5, a performance evaluation of HCOCOP and its comparison with centralized architectures are presented. Finally, Section 6 concludes by providing final remarks, and by listing some of the main directions for future research.

2. Related Work

Since our work aims at integrating collaborative playback protocols in CDN architectures, in this section we will introduce the state-of-the-art of the collaborative playback protocols which are currently available in centralized architectures and, subsequently, the state-of-the-art regarding the use of CDNs for media streaming.

With reference to the centralized architectures for collaborative playback control, the following works have investigated the development of a full-fledged collaborative playback service on the Internet: (Schuett et al., 1998) proposed the Soft State Archive Control protocol (SSAC) which is the cooperative playback protocol of the MASH Rover system, a multicast-based client/server system for remote playback developed at the University of Berkeley; (Fortino and Nigro, 2000; Fortino and Nigro, 2003) presented the Multicast Archive Control Protocol (MAC π) which is the cooperative playback protocol of ViCRO^C, a multicast-based client/server system for remote playback and recording developed at the University of Calabria. Both protocols have similar

characteristics with respect to:

- *Service architecture.* Both the MASH Rover system and the ViCRO^C system are based on a centralized server providing media streaming and streaming control
- *Multicasting.* The transmission and reception of control requests and replies are based on IP multicast (Crowcroft et al., 1999);
- *Coordination among clients.* Submission of control commands is based on a random-based policy without explicit or implicit coordination (Dommel and Garcia-Luna-Aceves, 1999);
- *Session state management.* The server holds the session state and changes it each time a control request is accepted. The server announces the new state to all the members according to a soft-state paradigm (Raman and McCanne, 1999);
- *Distributed playback synchronization.* The media streaming synchronization among the group members is driven by the server without any resynchronization mechanism at the client side as the media time is set by the server when it accepts a control request.

To improve performance of MAC π and SSAC, the COoperative Control protocol (COCOP) introduced an implicit coordination mechanism among clients which allows a client to refrain from issuing a control request after sensing that another client has issued a control request (Fortino et al., 2005).

MAC π , SSAC and COCOP are supported by a control architecture, hereafter named “Star”, which is centralized and consists of a single control server to which all members of the collaborative playback session are attached.

With reference to CDN architectures for media streaming delivery, or “streaming CDNs” (SCDNs), they have been introduced to improve the performance of media streaming on the Internet. The state-of-the-art of the SCDNs provides significant results for client redirection mechanisms, multimedia data management, and media streaming delivery. Below, some well-known research efforts regarding SCDNs are summarized.

PRISM (PoRtal Infrastructure for Streaming Media) is a research-oriented SCDN for distributing, storing and delivering high quality streaming media over IP networks (Cranor et al., 2001). The PRISM-based stored-TV (STV) service allows users to select content based on program name as well as the time it was aired. Content stored inside the network is accessible throughout the whole PRISM infrastructure. MARCONINet (Dutta and

Schulzrinne, 2004) is a research-oriented architecture for IP-based radio and TV networks, built on standard Internet protocols including RTP, RTSP, SAP, and SDP. It allows for the building of virtual radio networks, similar to traditional AM/FM radio and TV networks. The research-oriented Video CDN (VCDN) architecture (Cahill and Sreenan, 2006) is a hybrid CDN-P2P network which exploits the dynamic nature of a P2P architecture while providing a CDN service atop this overlay network. This design choice is tailored to overcome the limitation of CDN extensibility, while also minimizing the amount of overall resources required to serve a given client request pattern. In VCDN, ISP servers can advertise their willingness to partake in the system by acting as peers.

An interesting preliminary result pertaining to the integration of collaborative playback protocols and SCDN architectures is represented by the COMODIN (COoperative Media On-Demand on the InterNet) system (Esteve et al., 2007). COMODIN is an SCDN which provides both basic media streaming services through a Base plane and, differently from the other SCDNs, cooperative playback services through a Collaborative plane. The Base plane of COMODIN (Molina et al., 2006) adopts the architectural organization of PRISM in the data, control and management planes. The Collaborative plane introduces additional components to fully support the collaborative playback service.

This paper defines and analyzes a new protocol for streaming control in SCDN, named HCOCOP, which is a key component of the COMODIN reference architecture and can be incorporated into any hierarchical SCDN architecture. HCOCOP (Hierarchical Cooperative Control Protocol), which is an extension of COCOP targeted to CDN architectures, is able to further improve performance of COCOP as demonstrated in Section 5.

3. The reference CDN-based Architecture providing Collaborative Playbacks

The reference CDN-based architecture for group-oriented playbacks is organized into two planes: the *Base* plane, which consists of a streaming CDN (SCDN) (Molina et al., 2006) providing on-demand media streaming services, and the *Collaborative* plane which provides the collaborative playback service (Esteve et al., 2007).

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

- The *Origin*, which archives the media objects to be distributed by the CDN and delegates the URI name

space for such objects which are distributed and delivered by the CDN system. The Origin distributes the content to the surrogates of the CDN through the distribution network by means of bulk transfer mechanisms.

- The *Surrogate*, which is a partial replica of the Origin and is provided with the additional ability to temporarily store and deliver content to clients through the access network by using the Media Streaming Server (MSS) component, a networked component of the Base plane capable of streaming a media session to a subgroup of clients.
- The *Client*, which is usually an individual PC and requests specific media content stored in the CDN.

Moreover, the *Base* plane is equipped with a redirection system capable of redirecting clients to the optimal surrogate (Molina et al., 2006).

The architecture of the *Collaborative* plane (see Fig. 1) introduces the following additional components to provide the collaborative playback service:

- The *Collaborative Playback Session Manager* (CPSM), which coordinates the other components of the collaborative plane and provides the *group formation and management* core service based on the collaborative playback session management protocol (CMP). In particular, the *group formation and management* core service allows for the formation, (un)subscription, initiation, joining/leaving, and termination of *collaborative playback sessions* (CPSs). A CPS is a networked multimedia session in which a collaborative playback service is provided to a synchronous and explicitly-formed group of clients. Moreover, the CPSM archives information about the available media playbacks and the organized CPSs, i.e. CPSs organized by a client request and identified by a unique identifier (or CPSId) assigned by the CPSM.
- The *Collaborative Playback Control Server* (CPCS), which is located in each Surrogate and integrated with the *Base* plane MSS component. The CPCS supports the remote control of the media streaming shared among the members of a CPS served by the same Surrogate.
- The *CPCS Coordination Channel* (CCC), which coordinates the distributed CPCSs through the coordination channel protocol (CCP). The CCC spawns a front-end for each initiated CPS, i.e. a CPS started by its organizer client, to coordinate the CPCS front-ends serving the same CPS. Coordination among CPCS front-ends is needed for (i) deciding which CPCS front-end is enabled to accept a control request, (ii)

synchronizing the CPCS front-ends with respect to the media session time and (iii) allowing clients to join asynchronously.

- The *Collaborative Client (CC)*, which is a networked multimedia application which interfaces the user with the CDN-based collaborative playback service.

The CCC, CPCS and CC components interact according to the HCOCOP to provide the media streaming control. For the sake of clarity, the acronyms of the *Collaborative* plane components and protocols are reported in Table 1.

A CPS can be set up and run according to the following base scenario organized in sequential phases:

1. *Organization*. An organizing 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 (e.g. email, instant messaging, SIP-based tools, Mbone-based Session Directory (Crowcroft et al., 1999)).
3. *Subscription*. CCs connect to CPSM and subscribe to the CPS, either after invitation or spontaneously.
4. *Initiation*. The organizer CC connects to the CPSM, requests the initiation of the CPS and is finally redirected to an assigned CPCS.
5. *Join*. The CC subscribers of the CPS join the CPS to become CPS members and are consequently redirected to their assigned CPCSs.
6. *Execution*. The CPS is started by any member issuing the PLAY control request and evolves through a sequence of successive control requests (e.g., PAUSE, PLAY, SEEK).
7. *Termination*. The CPS can be terminated by its organizer CC.

In the following subsection we first provide a formal definition of the CPS and then detail the hierarchical control structure which is exploited by HCOCOP.

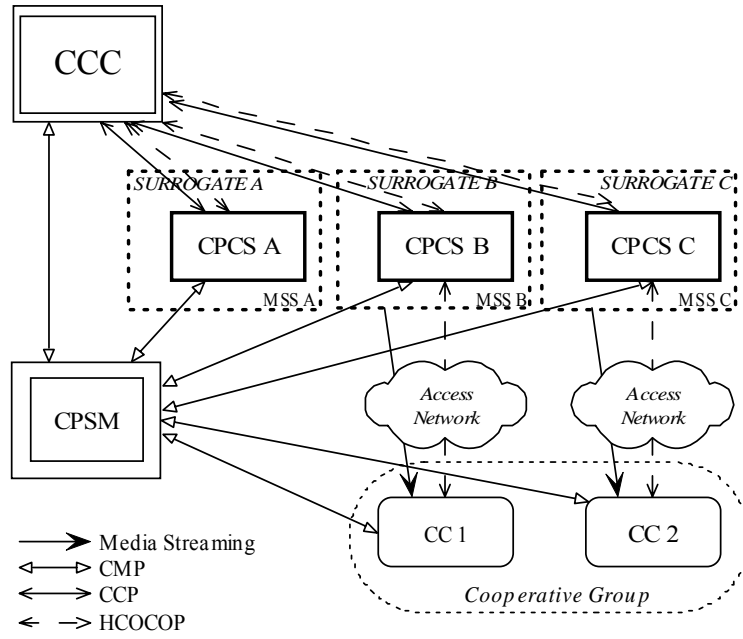


Fig. 1. The *Collaborative* plane of the reference CDN-based architecture.

Table 1. Acronyms of the Collaborative plane components and protocols.

<i>Acronym</i>	<i>Component Name</i>
CPSM	Collaborative Playback Session Manager
CPCS	Collaborative Playback Control Server
CCC	CPCS Coordination Channel
CC	Collaborative Client
CMP	CPS Management Protocol
CCP	Coordination Channel Protocol
HCOCOP	Hierarchical Cooperative Control Protocol

3.1. Control Structure of a Collaborative Playback Session

A CPS enabled by the previously introduced CDN-based architecture can be defined as the tuple:

$$\langle G, SMF, SCF, SMC, CF, MS, CS \rangle$$

where:

- G is the synchronous group of CCs;

- SMF is the set of media streaming front-ends of the MSSs spawn for the CPS;
- SCF is the set of streaming control front-ends of the CPCSs spawn for the CPS;
- $SMC=SMF\times SCF$ is the set of pairs composed of a front-end of the MSS and the corresponding front-end of the CPCS which controls it;
- CF is the front-end of the CCC spawn for the CPS;
- $MS\subseteq(SMF\times G)$ is the media delivery structure on which media streams are transmitted, i.e. the media communication relation which links each client of G to its respective MSS front-end in SMF .
- $CS\subseteq(SCF\times G)\cup(CF\times SCF)$ is the hierarchical control structure on which control messages are transmitted, i.e. the control communication relation which links each client of G (*leaf level*) to its respective CPCS front-end in SCF (*intermediate level*) and each CPCS front-end in SCF to CF (*root level*).

With respect to the CS , G is partitioned into a number of subgroups each attached to a given CPCS front-end in SCF . The formation of the subgroups is dynamic and occurs when a client of G joins the CPS. Every client of G is redirected by the CDN to its optimal surrogate, i.e. the surrogate which can better serve it. The dynamic subgroups formation relies on this redirection mechanism (Molina et al., 2006).

Fig. 2 shows the hierarchical control structure (CS) for an initiated CPS (hereafter called CPS_K) in which m subgroups $\{G^k_1, \dots, G^k_m\}$, where $G^k_i=\{CC^{k,i}_1, \dots, CC^{k,i}_{n_i}\}$, are formed and respectively attached to the m corresponding CPCS front-ends $\{CPCS^k_1, \dots, CPCS^k_m\}$ which are, in turn, attached to the CCC_K .

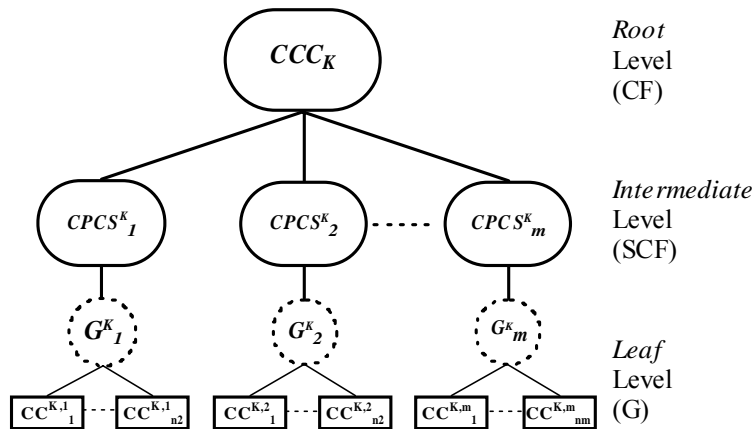


Fig. 2. Hierarchical Control Structure of a Collaborative Playback Session.

4. The Hierarchical Cooperative Control Protocol

HCOCOP (Hierarchical Cooperative Control Protocol), which enables the control of collaborative playback sessions in hierarchical SCDNs, relies on the following characteristics:

- *random-based transmission policy of the control requests.* A control request can be sent without explicitly acquiring the rights to send it. This policy increases service interactivity by avoiding the exploitation of explicit synchronization mechanisms (e.g. floor control (Dommel and Garcia-Luna-Aceves, 1999)) as these introduce additional delays in the session control;
- *FCFS-like policy for the acceptance of a control request.* The first incoming request is accepted while the others discarded for a given time interval. In fact, in a collaborative playback session, control requests are not queued due to their tight coupling with the on-going playback session, since queued control requests could have no meaning when the playback session changes.
- *cooperation-based mechanism* A client blocks itself when it senses that a control request has been issued by another client so to avoid the transmission of control requests which would likely be discarded;
- *soft state-based management of the control session state.* The session state changes according to the soft-state paradigm (Schuett et al., 1998; Raman and McCanne, 1999).

The full-fledged version of HCOCOP works as follows. Given a CPS (hereafter called CPS_K), a control request (CIReq) generated by a client x ($CC^{K,i}_x$) belonging to the subgroup G^K_i , if accepted at the *intermediate* level of the hierarchy, is forwarded *upwards* to the *root* level. In particular, a CIReq is accepted at the *intermediate* level (by the $CPCS^K_i$) and subsequently at the *root* level (by the CCC_K) if the servers at those levels are not serving other CIReqs. A CIReq which cannot be accepted at any level will not be served.

A Reply issued by the CCC_K upon the acceptance of a CIReq is forwarded *downwards* to all $CPCS^K_w$ and then by $CPCS^K_w$ to the clients of the respective subgroups, in order to enable such clients to process it. After processing the Reply, clients can issue new control requests. Delay timers are introduced to make clients aware of changes in the session state and to guarantee the consistency of HCOCOP as detailed in Section 4.2.

HCOCOP supports an implicit *cooperation* mechanism among clients which enables a client to sense CIReqs issued by other clients and induces it to refrain from issuing new CIReqs, which would probably not be served

but only increase the session load. The implicit *cooperation* mechanism of HCOCOP can operate according to the following modes which depend on how a ClReq generated by a client of the subgroup G_i^K is handled:

- global cooperation (*GlobalCoop*): the ClReq is forwarded downwards by the $CPCS_i^K$ to all other clients belonging to the subgroup G_i^K and by the CCC_K to all $CPCS_j^K$ ($j \neq i$) and then to all the clients of the other subgroups G_j^K (with $j \neq i$);
- local cooperation (*LocalCoop*): the ClReq is only forwarded downwards by the $CPCS_i^K$ to all other clients of the subgroup G_i^K ;
- no cooperation (*NoCoop*): the ClReq is not forwarded to the other clients of the subgroup G_i^K nor to the other subgroups.

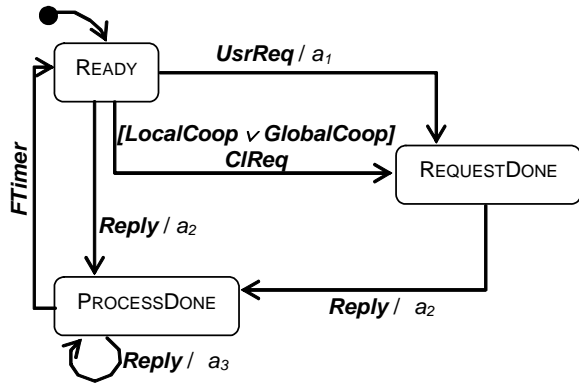
In the following, HCOCOP is described through an automata-based approach.

4.1. HCOCOP Automata

Four automata are defined to describe HCOCOP: (i) the CPCS Client Automaton, which runs in the CC; (ii-iii) the CPCS Server Automaton and the CCC Client Automaton which run in the CPCS; (iv) the CCC Server Automaton which runs in the CCC.

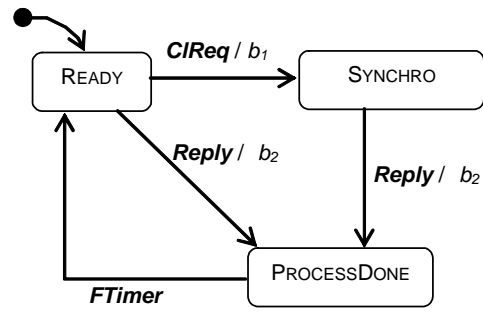
The cooperation mechanism which the pair CPCS Client Automaton and CPCS Server Automaton implements enables cooperation on a subgroup basis (or *LocalCoop* mode), i.e. among the clients belonging to the same subgroup. To enable *GlobalCoop* mode such a cooperation mechanism is also used by the pair CCC Client Automaton and CCC Server Automaton. No cooperation mechanism is exploited under the *NoCoop* mode.

The following subsections detail the HCOCOP automata reported in Fig. 3 whereas the flow of the HCOCOP control messages is shown in Fig. 4.



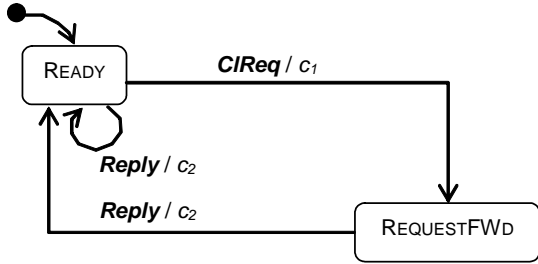
a_1 : sendToCPCSServer(CIReq);
 a_2 : process(Reply); setTimer(FTimer, T_{CC});
 a_3 : process(Reply); resetTimer(FTimer, T_{CC});

(a)



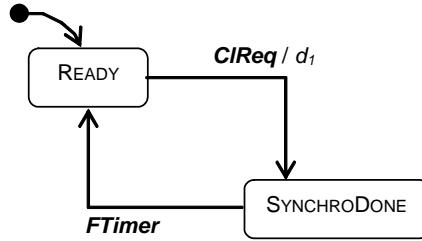
b_1 : if (CIReq.type==up) {
 if (LocalCoop || GlobalCoop)
 forwardToCPCSClients(CIReq);
 passToCCCClient(CIReq);
 }
 else forwardToCPCSClients(CIReq);
 b_2 : process(Reply);
 sendToCPCSClients(Reply);
 setTimer(FTimer, T_{CPCS});

(b)



c_1 : if (CIReq.type==down)
 forwardToCPCSServer(CIReq);
 else forwardToCCCServer(CIReq);
 c_2 : passToCPCSServer(Reply);

(c)



d_1 : if (GlobalCoop) forwardToCCClients(CIReq);
 process(CIReq);
 sendToCCClients(Reply);
 setTimer(FTimer, T_{CCC});

(d)

Fig. 3. The HCOCOP Automata: (a) CPCS Client Automaton, (b) CPCS Server Automaton, (c) CCC Client Automaton, (d) CCC Server Automaton.

CPCS Client Automaton

The CPCS client automaton is reported in Fig. 3(a). When the user issues a control request (UsrReq) and the state is Ready, the CPCS Client Automaton of the client $CC^{K,i}_x$ generates a client request (a CIReq of type *up*), sends it to the $CPCS^K_i$, and passes into the RequestDone state. This state is also entered when the client $CC^{K,i}_x$ in the Ready state senses a CIReq sent by another client belonging to the same subgroup (if *LocalCoop* holds) or to whatever subgroup (if *GlobalCoop* holds). In the RequestDone state, in which the automaton rests until a Reply is received, additional CIReqs sent by other clients are ignored and the client $CC^{K,i}_x$ is disabled from generating

new control requests to limit the session load (specifically the load of network, servers, and clients).

In particular the generation of new CIReqs should be avoided because: (i) they are meaningless if the same client has already generated a CIReq and is waiting for a Reply, and (ii) they will not be probably served.

When a Reply arrives, no matter the Automaton state, it is processed and, to control the interactivity degree of the session, new user requests are blocked until a given time T_{CC} , set up by a timer (FTimer), elapses. The setting of the timer is detailed in Section 4.2.

CPCS Server Automaton

The CPCS server automaton is reported in Fig. 3(b). As shown in Fig. 4, a client request (CIReq) can be sent to the CPCS Server Automaton of $CPCS^K_i$ either from the CPCS Client Automaton of the client $CC^{K,i}_x$ (a CIReq of type *up*) or from the CCC Client Automaton of $CPCS^K_i$ (a CIReq of type *down*). In the Ready state the reception of a CIReq, which leads the CPCS Server Automaton to the Synchro state, is processed depending on its type: (i) an *up* CIReq is passed to the CCC Client Automaton of $CPCS^K_i$ and, if local or global cooperation is enabled, forwarded to the other CPCS clients of the subgroup G^K_i ; (ii) a *down* CIReq, which can be received only if global cooperation is enabled, is forwarded to all the CPCS Clients of the subgroup G^K_i . In the Synchro or Ready states, upon receiving a Reply from the CCC Client Automaton, the CPCS Server Automaton processes the Reply and forwards it to all the clients of the subgroup G^K_i . Afterwards it enters the ProcessDone state wherein it rests until a given time T_{CPCS} elapses. In particular a timer FTimer, whose setting is detailed in Section 4.2, is introduced both to make the clients aware of all changes in the session state, thus exploiting a soft-state like paradigm (Fortino et al., 2005; Raman and McCanne, 1999), and to regulate the group interactivity.

CCC Client Automaton

The CCC client automaton is reported in Fig. 3(c). A client request (CIReq) can be sent to the CCC Client Automaton of $CPCS^K_i$ either from its CPCS Server Automaton (a CIReq of type *up*) or from the CCC Server Automaton (a CIReq of type *down*). In the Ready state, upon receiving a CIReq, the CCC Client Automaton: 1) if the CIReq type is *up*, forwards it to the CCC Server Automaton, otherwise forwards the CIReq to its CPCS

Server Automaton; 2) passes into the RequestFwd state wherein it waits for a Reply sent by the CCC Server Automaton. Upon receiving the Reply, the CCC Client Automaton passes into the Ready state after forwarding the Reply to its CPCS Server Automaton.

CCC Server Automaton

The CCC server automaton is reported in Fig. 3(d). A ClReq sent by the CCC Client Automaton of $CPCS^K_i$ and received in the Ready state is forwarded to all the other CCC Client Automata, if global cooperation is enabled, and accepted by the CCC Server Automaton. A Reply is then sent to all the CCC Client Automata and the Automaton passes into the SynchroDone state wherein it rests until a given time T_{CCC} elapses. In particular a timer FTimer, whose setting is detailed in Section 4.2, is introduced to assure the consistency of HCOCOP.

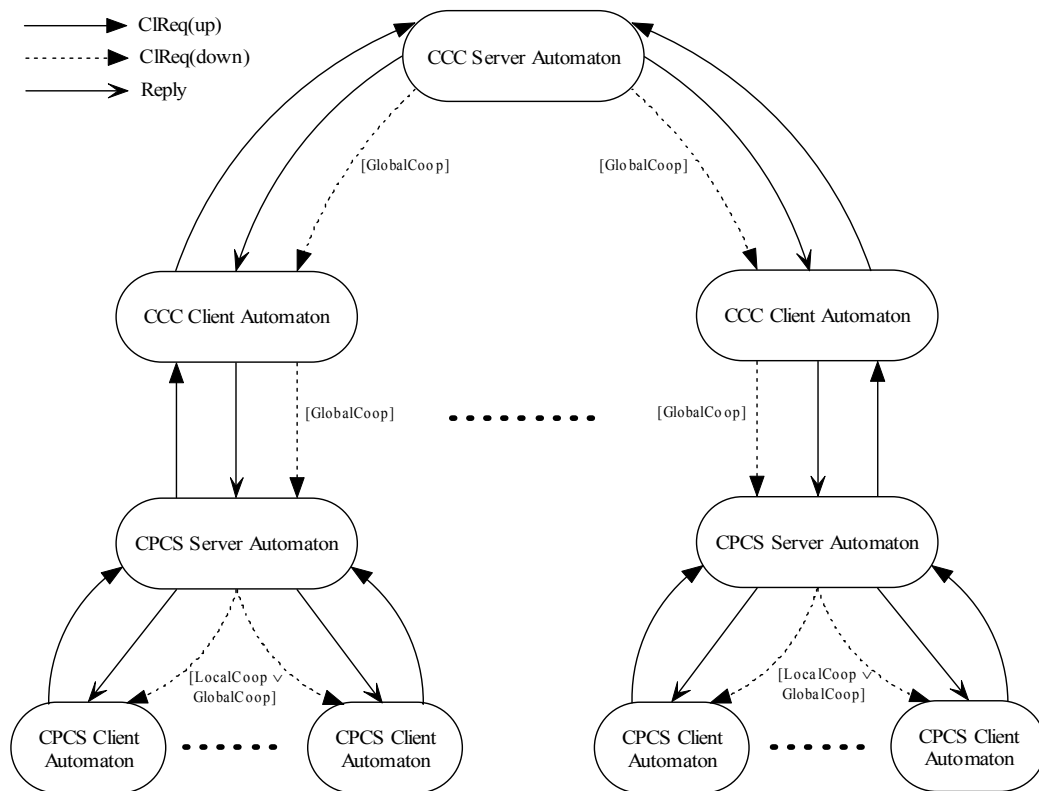


Fig. 4. Flow of the HCOCOP Control Messages.

4.2 Relationships among client/server timers

Timers T_{CC} , T_{CPCS} and T_{CCC} introduced to make clients aware of all changes in session state and to regulate the group interactivity, are set as discussed below to assure the consistency of HCOCOP and, in particular, to prevent session deadlock.

The time sequence diagram depicted in Fig. 5 exemplifies an interaction between two generic clients (C_1 and C_2) and their related server (S). In particular, the client/server pair can be one of the following: (i) $\{C_1=CC_1, C_2=CC_2\}$ and $S=CPCS$; (ii) $\{C_1=CPCS_1, C_2=CPCS_2\}$ and $S=CCC$.

The time sequence diagram highlights a session deadlock due to wrong timer settings at the server (T_S) and client (T_C) sides. In fact, as the server is unable to process a ClReq and produce a Reply, all the clients (not only the one that sent the ClReq) are blocked to wait for a Reply that would never come. To overcome this problem, T_S and T_C must be correctly set. A client block can occur when:

$$T_C < T_S - RTT_{C/S} - T_{PROC},$$

where $RTT_{C/S}$ is the c/s round trip time and T_{PROC} is the server elaboration time.

Such session deadlock can be avoided by taking into account the worst case (where $RTT_{C/S}$ and T_{PROC} are negligible) and thus setting T_C equal to T_S for all clients.

According to this outcome, to avoid session deadlocks the following constraint must hold: $T_{CC}=T_{CPCS}=T_{CCC}$. The constraint is based on the consideration that CCC is the server of the CPCS which is the server of the CC; this implies that the timer of the first server (i.e. the CCC) must be set before and the other timers are set accordingly.

The simulation analysis permits the evaluation of the performance improvements that can be achieved with the two enhancements discussed in this paper (CDN vs. Star, and the use of cooperation) with respect to a classical centralized control protocol that does not adopt any cooperation mechanisms.

The performance indices used to analyze and compare the different protocols are defined in Table 2, which also gives details on how such indices are calculated.

The analysis aims at evaluating:

- the capability of a generic client to obtain control of the collaborative playback session in terms of the denial probability (i.e. the probability that a client request is not served) and blocking probability (i.e. the probability that a user request is blocked by the client);
- the server load;
- the network system load.

In the CDN, the denial probability $P_{den}(CDN)$ is defined as the probability that a client request is discarded at either the *intermediate* or the *root* level of the CDN and is calculated as:

$$P_{den}(CDN) = P_{den}(CPCS) + (1 - P_{den}(CPCS)) \cdot P_{den}(CCC)$$

where $P_{den}(CPCS)$ and $P_{den}(CCC)$ are the denial probabilities at the CPCS server and at the CCC server respectively (see Table 2).

In the CDN, the server and network load indices are defined according to the CDN levels as follows:

- $SL(CPCS)$ and $SL(CCC)$ are the server loads (number of requests received per second) measured at a CPCS server and at the CCC server, respectively.
- The overall network load is calculated as

$NL(CDN) = NL(CPCS) + NL(CCC)$, where: (i) $NL(CPCS)$ is equal to $\sum_j(NL(CPCS_j))$, and $NL(CPCS_j)$ is the number of messages exchanged among all clients of the j -th subgroup and the j -th CPCS server; (ii) $NL(CCC)$ is the frequency of messages exchanged among all CPCS servers and the CCC server.

5.1. Simulation Parameters

Main simulation parameters (see Table 3) and their settings are as follows:

- *Duration of the session* (T_{Session}). For each simulation run 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%).
- *Number of clients* (N). In the performed simulations N is varied from 2 to 16 as cooperative control sessions are mainly intended for small/medium sized groups of users (Fortino and Nigro, 2000; Schuett et al., 1998).
- *Mean Request Interarrival Time* (MRIT). This is the average interarrival time between two successive requests issued by the same user.
- *User activity* (UA). This is modelled according to a statistical model based on the *Gamma* probability distribution function which has a shape parameter equal to 2 (Padhye and Kurose, 1999). It is characterized by MRIT and can be classified as very low ($\text{MRIT} \geq 15\text{m}$), low ($10\text{m} \leq \text{MRIT} < 15\text{m}$), medium ($5\text{m} \leq \text{MRIT} < 10\text{m}$), high ($120\text{s} \leq \text{MRIT} < 5\text{m}$) and very high ($\text{MRIT} < 120\text{s}$). To enable the complete evaluation of HCOCOP in sessions with high to very high user activity, the value of MRIT is varied within the range $\{10 \text{ s}, 180 \text{ s}\}$.
- The *delay* between two adjacent nodes (δ). δ is set according to the following link delay model:

$$\delta_i = K_f \delta_m + N(K_v \delta_m, \sqrt{K_v \delta_m})$$

$$K_f + K_v = 1 \quad K_f, K_v \geq 0$$

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 weights 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. Normal distribution is chosen according to the considerations presented in (Gibbon and Little, 1996). The parameters of the delay model are set according to the values measured in a CDN testbed established across Italy and Spain (Esteve et al., 2007). In particular, δ_m is set to 3 ms for the links between a client and the local 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 Star configuration.

- The *server processing delay* (T_{proc}). This 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 cooperative session. T_{proc} is set to 200 ms.

- The *server timers* (T_{CCC} , T_{CPCS} , T_{CC}). T_{CCC} and T_{CPCS} , used to control servers' reactivity and the overall degree of system interactivity, are both set to 3.0s, as is the client timer T_{CC} , to avoid deadlock situations, as shown in Section 4.2. The timer values are chosen on the basis of the performance analysis of the ViCRO^C system (Fortino and Nigro, 2003) and the COMODIN system (Esteve et al., 2007).

Table 2. Performance indices of HCOCOP.

$N_{usrReq(k)}$ $ke(CDN, Star)$	Number of user requests issued in a CDN or Star	
$N_{usrReqBlk(k)}$ $ke(CDN, Star)$	Number of user requests blocked in a CDN or Star	
$N_{clReq(i)}$ $ie(CPCS, CCC, Star)$	Number of client requests delivered to the CDN servers (CPCS or CCC) or the Star server	
$N_{clReqDis(i)}$ $ie(CPCS, CCC, Star)$	Number of client requests discarded by the CDN servers (CPCS or CCC) or the Star server	
$N_{clReqAcept(i)}$ $ie(CPCS, CCC, Star)$	Number of client requests accepted by the CDN servers (CPCS or CCC) or the Star server	
$P_{blk(k)}$ $ke(CDN, Star)$	$\frac{N_{usrReqBlk(k)}}{N_{usrReq(k)}}$	<i>Blocking probability:</i> the probability that a user request is blocked in a CDN or Star
$P_{den(i)}$ $ie(CPCS, CCC, Star)$	$\frac{N_{clReqDis(i)}}{N_{clReq(i)}}$	<i>Denial probability:</i> the probability that a client request is discarded by the CDN servers (CPCS or CCC) or the Star server
$SL(i)$ $ie(CPCS, CCC, Star)$	$\frac{N_{clReq(i)}}{T_{session}}$	<i>Server load:</i> the number of requests per second that a CDN server (CPCS or CCC) or the Star server receives
$NL(i)$ $ie(CPCS, CCC, Star)$	<i>Network load:</i> the number of messages per second that circulate between all subgroups and their related CPCSs, between all CPCSs and their related CCC, and in the Star network	
$P_{den(CDN)}$	$P_{den(CDN)} = P_{den(CPCS)} + (1 - P_{den(CPCS)}) \cdot P_{den(CCC)}$	Probability that a client request is discarded at either the <i>intermediate</i> or the <i>root</i> level of the CDN
$NL(CDN)$	$NL(CDN) = NL(CPCS) + NL(CCC)$	Overall network load

5.2. The Discrete-Event Simulator

The discrete-event simulator, written in C++, is fully object-oriented and designed with the following objects:

1. Client, CPCS Server, CCC Client and CCC Server Nodes, which model the client and server processes,

according to the automata shown in Fig. 3;

2. User Agent, which generates new requests on behalf of the user. The request generation process follows the Gamma distribution, as discussed in Section 5.1;
3. Event, which embodies a message exchanged among Client and Server objects. Upon event reception, a Node handles the event according to its automaton;
4. Event Scheduler/Dispatcher, which manages events, stores them in a queue ordered by message delivery times, and dispatches them to destination nodes.

Table 3. Simulation parameters of HCOOP.

$T_{session}$	Duration of the session
N	Number of clients
MRIT	Mean Request Interarrival Time
UA	User Activity
δ	Link delay between two adjacent nodes
T_{proc}	Processing delay on a server
T_{cc}	Client timer
T_{cpcs}	CPCS Server timer
T_{ccc}	CCC Server timer

5.3. Simulation Results

The simulation phase aims at evaluating a wide range of CDN topologies, i.e. symmetric and asymmetric topologies having different numbers of clients and CPCS servers, and different sizes of local subgroups.

The following presents some simulation results which are achieved in four scenarios designed to evaluate the main features of HCOCOP and in particular, the benefits of cooperation in different CDN architectures. In particular the scenarios are:

- *Scenario 1:* symmetric CDN topologies in which a fixed number of clients are equally distributed among 2 subgroups served by the respective CPCS servers.
- *Scenario 2:* symmetric CDN topologies in which 2 subgroups and a variable number of clients, equally distributed among the subgroups, are considered.
- *Scenario 3:* asymmetric CDN topologies in which a fixed number of clients are unevenly distributed between 2 subgroups.
- *Scenario 4:* symmetric CDN topologies in which a given number of clients are equally distributed among a

variable number of subgroups.

For all the mentioned scenarios, the HCOCOP performances obtained are compared with the performances obtained for the Star with the same number of clients as the CDN topologies.

Scenario 1: symmetric topology with N clients and 2 CPCS servers

Figures 6-12 are related to a symmetric CDN having N=12 clients and 2 subgroups, with 6 clients per subgroup. In Fig. 6, which shows the denial probability at the CPCS server, $P_{den}(CPCS)$, the benefits of the cooperation modes are evident. The *LocalCoop* mode which disables new client requests when the client senses a request issued by another client of the same subgroup (as described in Section 4), significantly decreases the denial probability. Benefits of cooperation are more evident in the *GlobalCoop* mode as clients at the local CPCS server experience even less denial probability since they are also able to detect a request issued by clients belonging to the other subgroup.

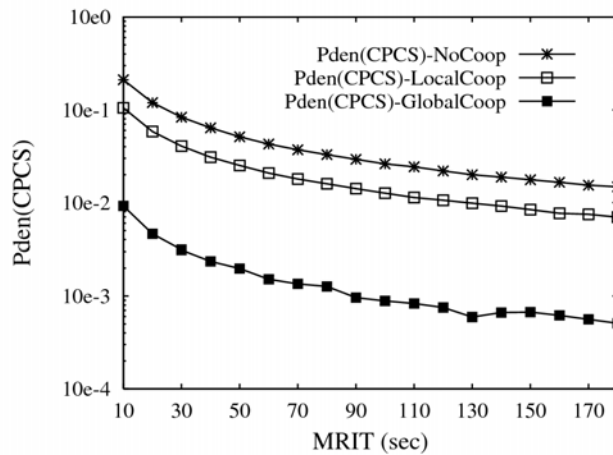


Fig. 6. Denial probability $P_{den}(CPCS)$ vs. the MRIT. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

Fig. 7 shows that the denial probability at the CCC server, $P_{den}(CCC)$ is not appreciably modified by the cooperation approach. However, the denial probability $P_{den}(CDN)$, which depends on $P_{den}(CPCS)$ and $P_{den}(CCC)$ as reported in Table 2, confirms the benefits of the cooperation approach, as shown in Fig. 8. The

same figure also compares the denial probabilities experienced in the CDN and Star architectures. 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. More importantly, denial probabilities obtained in the CDN with *GlobalCoop* are far lower than all other cases.

While the cooperation approach and the adoption of the CDN cause a significant decrease in denial probability they have little impact on the blocking probability, as shown in Fig. 9. However, the collaboration modes imply an additional cost, in terms of network and server load. In fact, with the cooperation modes, a number of additional messages circulate between the leaf and intermediate level in order to inform clients about requests issued by other clients of the same subgroup (if *LocalCoop* mode is enabled) or by other clients belonging to other subgroups (if *GlobalCoop* mode is enabled). The weight of such messages can be observed in Fig. 10, which reports the message load $NL(CPCS_j)$ measured within one of the two subgroups.

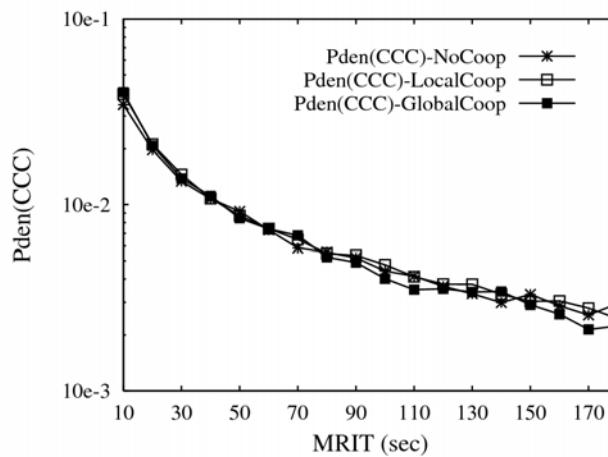


Fig. 7. Denial probability $Pden(CCC)$ vs. the MRIT. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

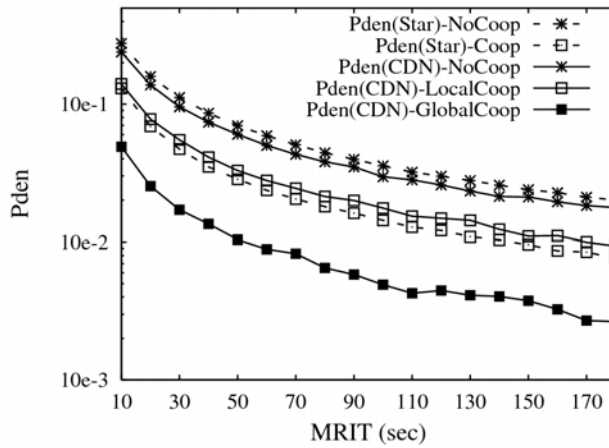


Fig. 8. Overall denial probability vs. the MRIT. Comparison between CDN and Star architectures.

As shown in Fig. 11, the network load between the intermediate and root levels of the CDN increases only if *GlobalCoop* mode is enabled, due to the additional messages (client requests) sent by the CCC server to the CPCS servers.

Fig. 12 shows that the cooperation mechanism causes an increase in the overall network load $NL(CDN)$.

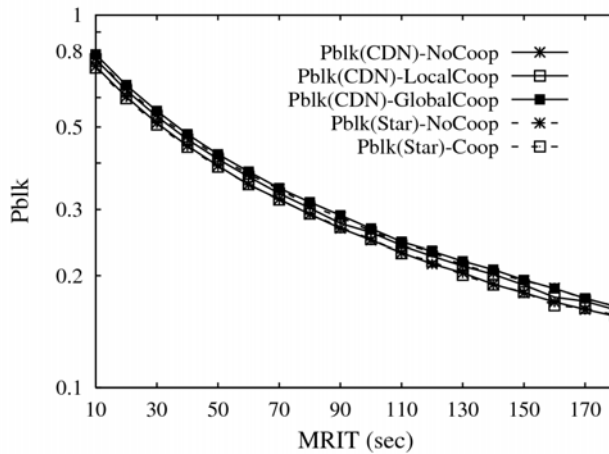


Fig. 9. Blocking probability Pblk vs. the MRIT. Comparison between CDN and Star architectures.

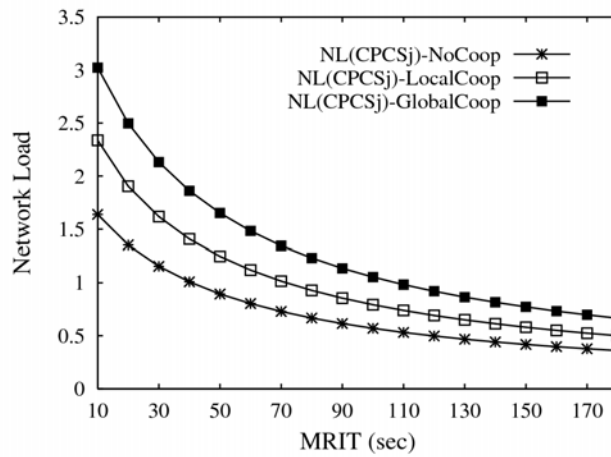


Fig. 10. Network load in a subgroup of the CDN architecture vs. the MRIT. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

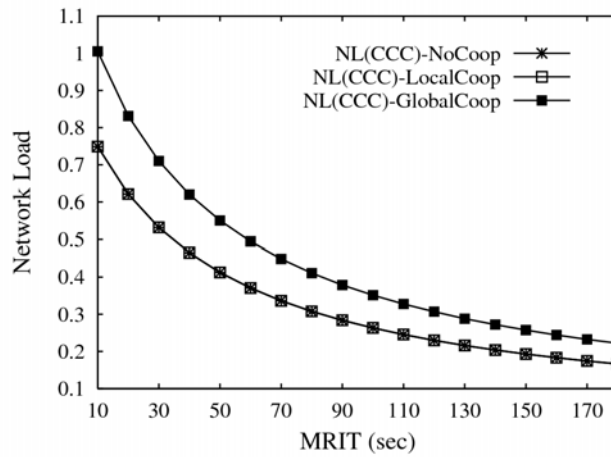


Fig. 11. Network load in the high layer of the CDN architecture vs. the MRIT. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

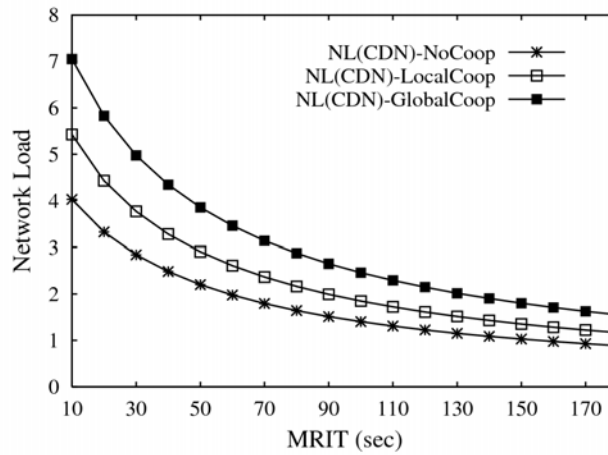


Fig. 12. Overall network load of the CDN architecture vs. the MRIT. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

Scenario 2: symmetric topology with 2 CPCS servers and a variable number of clients

As shown in Fig. 13, the decrease in denial probability discussed in scenario 1 is also evident in symmetric CDN topologies with 2 groups and different numbers of clients, ranging from 4 to 16. Denial probabilities increase with the number of clients, as a larger number of requests are generated. It is worth noting that the denial probabilities obtained in the CDN with *GlobalCoop* are always lower than 0.01, which is an acceptable value for a client. Blocking probabilities also increase with the number of clients but are not significantly affected neither by the adopted cooperation mode nor by the type of control architecture (CDN or Star).

The load of the Star server and of the CDN servers (CPCS and CCC) is reported in Fig. 14. Results are obtained in the CDN with the *GlobalCoop* mode and in the Star with the *Coop* mode. It can be noted that the adoption of the CDN does not imply an additional load for servers since the load at the CCC server is comparable with the load at the Star server, whereas the load at the CPCS servers is much lower. Finally, the network load is increased by the cooperation modes, as discussed for Scenario 1.

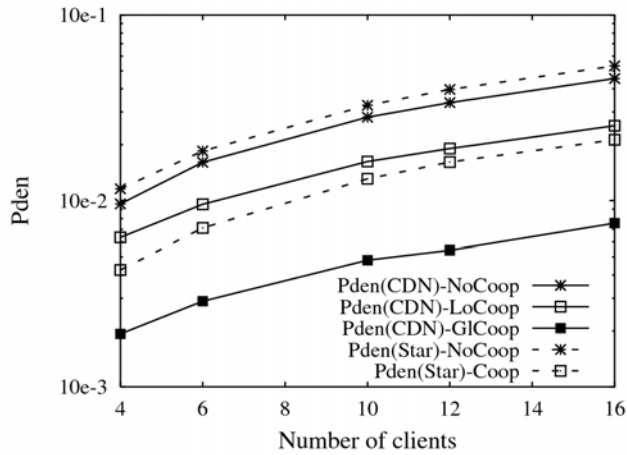


Fig. 13. Overall denial probability in CDN and Star architectures vs. the number of clients with MRIT=90.

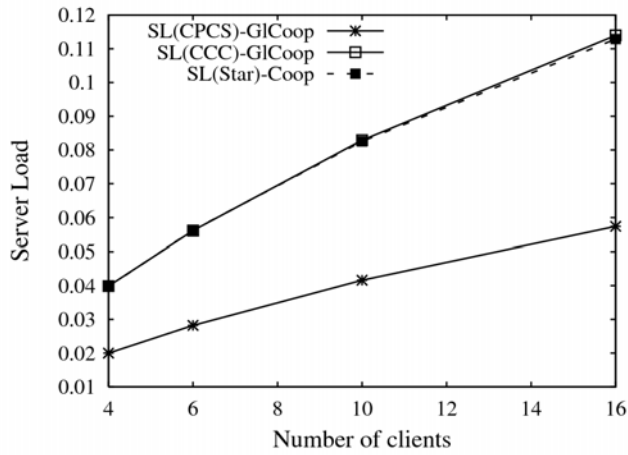


Fig. 14. Load of servers in CDN and Star architectures vs. the number of clients with MRIT=90.

Scenario 3: asymmetric topology with N clients unevenly distributed between 2 CPCS servers

A set of simulation runs are carried out to investigate the performance of the CDN in which clients are unevenly distributed among two CPCS servers. In particular, $N=12$ clients are allocated with 8 clients assigned to one CPCS and 4 to the other. Fig. 15 reports the overall denial probability experienced by the clients belonging to the two subgroups under all three operational modes. Comparison shows that, with *NoCoop* no difference in denial probability is found between the two subgroups, whereas with *LocalCoop* and *GlobalCoop*, the clients that belong to the most numerous subgroup are favored. Indeed, when a client belonging to the 8-client subgroup gains control of the local CPCS server, it has a higher chance of controlling the CCC server than

a client belonging to the other subgroup. This phenomenon can be considered a beneficial outcome of the cooperation mechanism which favors the most numerous subgroup. Therefore clients should be redirected to existing subgroups because isolated clients or clients belonging to very small subgroups are penalized with respect to the clients of larger subgroups.

The server load for the asymmetric architecture is reported in Fig. 16. The CCC server load is comparable to the Star server load, as also shown in the scenario 2. Furthermore, this figure shows that the load of a CPCS server increases linearly with the number of attached clients. In fact the load of the server which serves the subgroup of 8 clients, denoted as $SL(CPCS)-GI\text{Coop}(8)$, is about twice the load sustained by the server of the other subgroup, $SL(CPCS)-GI\text{Coop}(4)$.

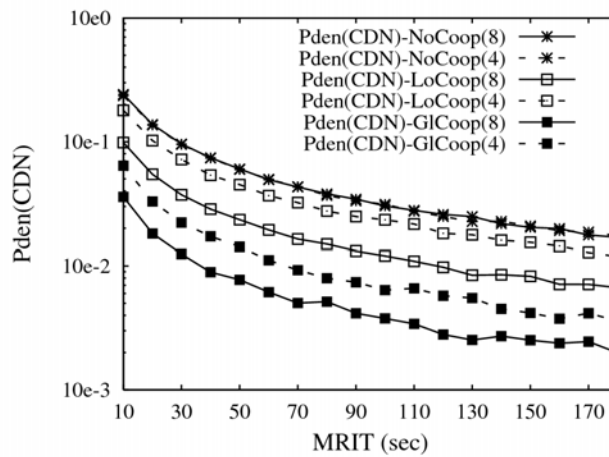


Fig. 15. Overall denial probability vs. the MRIT in an asymmetric CDN architecture.

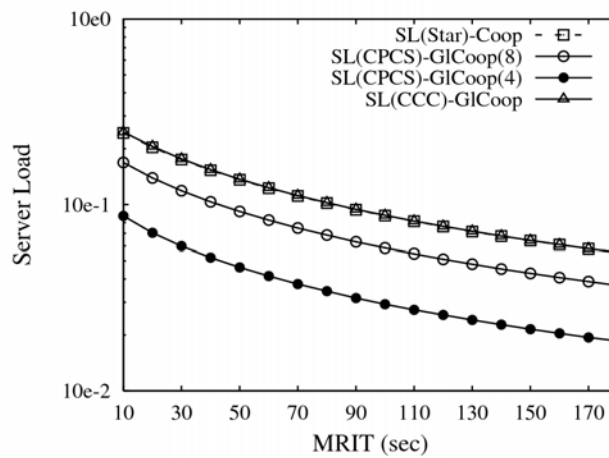


Fig. 16. Load of servers in an asymmetric CDN architecture and in a Star architecture.

Scenario 4: symmetric topology with N clients distributed among a variable number of CPCS servers.

This scenario is used to evaluate the performance of several CDN architectures with the same number of clients and a variable number of CPCS servers. Specifically, $N=12$ clients are distributed in equal numbers among 2, 3, 4 and 6 CPCS servers. The denial probabilities obtained with *GlobalCoop* are depicted in Fig. 17, in which the label “mS” is related to the topology with m subgroups, with m equal to 2, 3, 4 or 6. Interestingly, denial probability decreases when the size of the subgroups increases, thus confirming that cooperation favors the aggregation of clients whereas isolated clients are penalized. Fig. 17 also compares denial probabilities obtained with CDN and Star architectures. In particular, the CDN architecture with the *GlobalCoop* mode offers a significant advantage with respect to the Star architecture operating with the *NoCoop* mode.

Fig. 18 shows that the server load of a generic CPCS increases with the number of attached clients. Finally, Fig. 19 shows that the network load increases when the clients are distributed among a larger number of CDN subgroups.

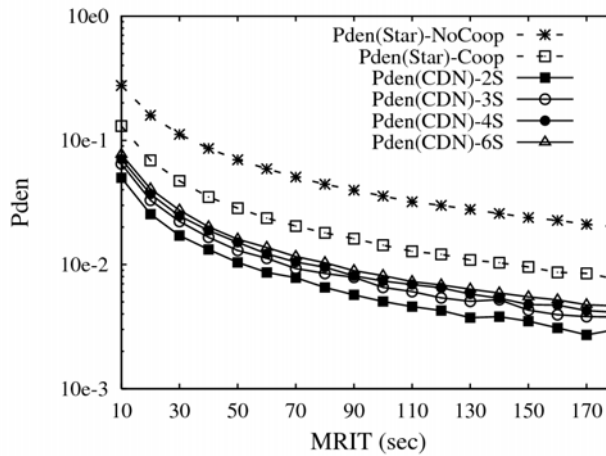


Fig. 17. Overall denial probability vs. the MRIT. Comparison between a Star architecture with 12 clients and CDN architectures having a fixed number of clients (12) and different number of CPCS servers (2, 3, 4, 6).

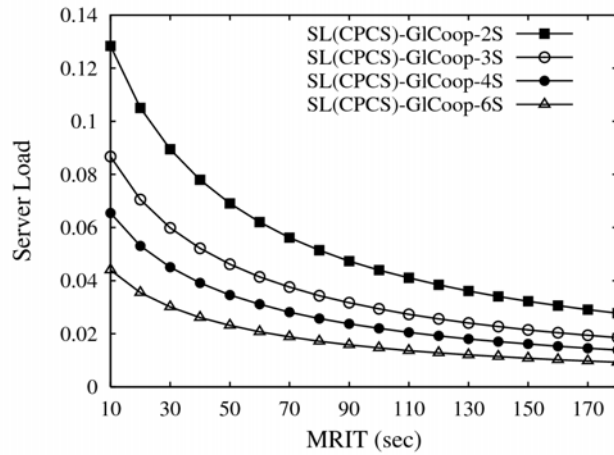


Fig. 18. Processing load of CPCS servers vs. the MRIT, in CDN architectures having a fixed number of clients (12) and different number of CPCS servers (2, 3, 4, 6).

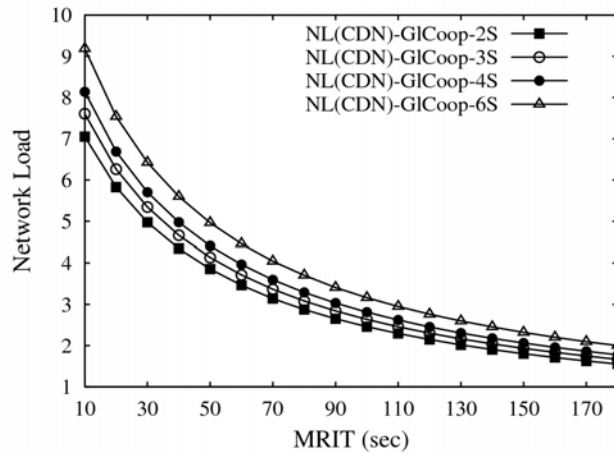


Fig. 19. Overall network load vs. the MRIT, in CDN architectures having a fixed number of clients (12) and different number of CPCS servers (2, 3, 4, 6).

Final remarks

Analysis was focused to evaluate the efficiency of HCOCOP on the basis of the defined performance indices: denial probability, blocking probability, server load and network load. Denial probability is the performance index that best characterizes the performance of HCOCOP. In particular, in a shared and cooperatively controlled playback session, server rejection of an issued control request is always considered a more unpleasant event than the inability to issue such request, as, while users are completely aware that they are not always able

to control the server, they expect that a control request, once issued, is likely to be accepted. Therefore, in collaborative playback sessions, denial probability is more important than blocking probability.

The analysis of simulation results indicates that the CDN architecture is more effective than the Star architecture and that cooperation significantly decreases the denial probability both in the CDN and Star architectures.

Moreover, cooperation in the CDN decreases denial probability more than in the Star architecture as a client is notified of any control requests issued by another client of the same subgroup (*LocalCoop*), through the exchange of short distance messages, as well as by clients of other subgroups (*GlobalCoop*). Furthermore, the use of the *GlobalCoop* mode exploited by HCOCOP achieves much lower values of denial probability than those obtained in the CDN with local cooperation or no cooperation and in the Star with and without cooperation. This significant decrease in denial probability due to the cooperation modes is obtained at the cost of a slight increase in the network load (as additional CIREqs circulate in the network), without, however, a significant increase in the load of servers and in the blocking probability (see Scenario 1).

Simulations performed with varying numbers of clients (see Scenario 2) further show that the use of CDN and the exploitation of cooperation modes significantly decrease the denial probability, even in applications that are not well scalable for their intrinsic characteristics (one single server must be concurrently controlled by several clients).

If the overall number of CDN clients is fixed (see Scenario 3) the clients that belong to the most numerous subgroup have more chances of controlling the server if one of the cooperation modes is enabled. These findings could be exploited to redirect clients to existing subgroups because isolated clients or clients belonging to very small subgroups are penalized with respect to the clients of larger subgroups.

Finally the benefits achieved through the exploitation of a CDN architecture are higher if clients are distributed among a few subgroups (see Scenario 4). Indeed the more subgroups, the closer the CDN topology is to the Star topology, since a Star may be seen as a CDN in which every client forms a 1-client subgroup. Hence, for a fixed number of clients, the denial probability increases with the number of subgroups.

5. Conclusions

Content Distribution Networks can not only efficiently support media streaming delivery but also feature the collaborative playback service, allowing for a synchronous group of users to select, simultaneously watch and share the control of a multimedia session.

This paper has presented the modeling and analysis of an application-level control protocol, Hierarchical Cooperative Control Protocol (HCOCOP), which makes it possible to control the media streaming of a collaborative playback session provided by a CDN. HCOCOP relies on the integration of a competitive access mechanism with a cooperation-based policy, avoiding, therefore, the adoption of higher-level mechanisms (e.g. floor control) to increase client-CDN interactivity. Performance evaluation of HCOCOP was carried out on symmetric and asymmetric CDN topologies by customizing and using an event-driven simulation framework. The analysis of the results shows that the global cooperation mechanism of HCOCOP significantly decreases the denial probability in the CDN architecture. Furthermore, it was demonstrated that the CDN architecture is more convenient than centralized architectures also from the point of view of media streaming control.

On the basis of these findings, current research efforts are underway to define and analyze a redirection system for CDNs providing collaborative playback sessions which is able to redirect clients to surrogates in such a way as to optimize not only media streaming delivery but also media streaming control.

References

- Almeroth KC, Ammar MH. The Interactive Multimedia Jukebox (IMJ): a new paradigm for the on-demand delivery of audio/video. Proceedings of the 7th International World Wide Web Conference (WWW7), 1998.
- Cahill J, Sreenan CJ. An Efficient Resource Management System for a Streaming Media Distribution Network. International Journal of Interactive Technology and Smart Education, 3(1), p. 31-44, February 2006.
- Cranor CD, Green M, Kalmanek C, Shur D, Sibal S, Sreenan CJ, Van der Merwe JE. Enhanced Streaming Services in a Content Distribution Network. IEEE Internet Computing, 5(4), p. 66-75, 2001.
- Crowcroft J, Handley M, Wakeman I. Internetworking Multimedia, Morgan Kaufmann Pub., San Francisco (USA), 1999.

- Dommel HP, Garcia-Luna-Aceves JJ. Group Coordination Support for synchronous Internet Collaboration, *IEEE Internet Computing*, 3(2), p. 74-80, March/April 1999.
- Dutta A, Schulzrinne H. MarconiNET: overlay mobile content architecture. *IEEE Communications*, 42(2), p. 64-75, February 2004.
- Esteve M, Fortino G, Mastroianni C, Palau C, Russo W. CDN-supported Collaborative Media Streaming Control. *IEEE Multimedia*, 14(2), p. 60--71, April 2007.
- Fortino G, Nigro L. A cooperative playback system for on-demand multimedia sessions over Internet. *Proceedings of IEEE Conference on Multimedia and Expo (ICME'00)*, New York, USA, August 2000.
- Fortino G, Nigro L. Collaborative Learning on-Demand on the Internet Mbone, *Usability Evaluation of Online Learning Programs*, Claude Ghaoui, Ed. Idea Group Publishing, Hershey (PA), USA, p. 40-68, 2003.
- Fortino G, Mastroianni C, Russo W. Cooperative Control of Multicast-based Streaming On-Demand Systems. *Future Generation Computer Systems*, 21(5), p. 823-839, 2005.
- Gibbon JF, Little TDC. Use of Network Delay Estimation for Multimedia Data Retrieval, *IEEE Journal on Selected Areas in Communications*, 14(7), p. 1376-1387, 1996.
- Holfelder W. Interactive remote recording and playback of multicast videoconferences. *Proceedings of IDMS'97*, Darmstadt, Germany, September 1997.
- Molina B, Palau CE, Esteve M, Alonso I, Ruiz V. On Content Delivery Network Implementation, *Computer Communications*, 29(12), p. 2396-2412, 2006.
- Padhye J, Kurose J. Continuous Media Courseware Server: a Study of Client Interactions, *IEEE Internet Computing*, 3(2), p. 65-72, 1999.
- Pallis G, Vakali A. Insight and Perspectives for Content Delivery Networks. *Communications of ACM*, 49(1), p. 101-106, 2006.
- Raman S, McCanne S. A model, analysis, and protocol framework for soft state-based communication, *ACM SIGCOMM Computer Communication Review*, 29(4), p. 15-25, 1999.
- Schuett A, Raman S, Chawathe Y, McCanne S, Katz R. A Soft State Protocol for Accessing Multimedia Archives. *Proceedings of NOSDAV'98*, Cambridge, UK, July 1998.

Giancarlo Fortino is an Associate Professor of Computer Science at the Department of Electronics, Informatics and Systems of the University of Calabria, Cosenza, Italy. He received the Laurea degree in Computer Engineering and the PhD in Computer Engineering at the University of Calabria (Italy) in 1995 and 2000, respectively. His research interests include Distributed Computing, Agent Oriented Software Engineering, Content Distribution Networks, Internet Computing, Distributed Multimedia Systems.

Carlo Mastroianni is a Researcher at the Institute of High Performance Computing and Networks of the Italian National Research Council (ICAR-CNR) in Cosenza, Italy. He received the Laurea degree in Computer Engineering and the PhD in Computer Engineering at the University of Calabria (Italy) in 1995 and 1999, respectively. His research interests include Grid Computing, Peer-to-Peer Networks, Content Distribution Networks, Multi Agent Systems.

Wilma Russo is a Professor of Computer Science at the Department of Electronics, Informatics and Systems (DEIS) of the University of Calabria, Cosenza, Italy. She received the Laurea degree in Physics at the University of Naples, Italy. Her research interests include Parallel and Distributed Computing and Systems, Agent Oriented Software Engineering, Internet Computing, Content Distribution Networks.