Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Collaborative Media Streaming Services based on Content Networks

G.Fortino, C. Mastroianni, W. Russo

RT-ICAR-CS-06-08 **Dicembre 2006**

Consiglio Nazionale delle Ricerche – Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sedi di Cosenza (*Via P. Bucci 41C, 87036 Rende, Italy*), di Napoli (*Via P. Castellino 111, 80131 Napoli*) e Palermo (*Viale delle Scienze, Parco D'Orleans, 90128 Palermo*)– URL: *www.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Collaborative Media Streaming Services based on Content Networks

G.Fortino[1], C. Mastroianni[2], W. Russo[1]

Rapporto Tecnico N.: **RT-ICAR-CS-06-08**

Data: **Dicembre 2006**

---

[1] Università degli Studi della Calabria, DEIS, Via P. Bucci 41C, Rende (CS)
[2] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Cosenza, Via P. Bucci 41C, 87036 Rende(CS)

Collaborative Media Streaming Services based on Content Networks

Giancarlo Fortino[§], Carlo Mastroianni[*], Wilma Russo[§]
[§] Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Università della Calabria
Via P. Bucci cubo 41C, 87036 Rende (CS), Italy
e-mail: {g.fortino, w.russo}@unical.it
[*] ICAR-CNR (Italian National Research Council)
Via P. Bucci cubo 41C, 87036 Rende (CS), Italy
e-mail: mastroianni@icar.cnr.it

Abstract

Content networks (CNs) are being introduced to extend content distribution networks (CDNs) with content creation, modification, management, and placement mechanisms to support value-added services and modern applications. This paper presents a CN-based architecture supporting collaborative media streaming services which allow a synchronous group of users to select, watch and control a multimedia session. In particular a hierarchical control protocol for cooperative media streaming control is defined and analyzed through simulation.

1. Introduction

Content Distribution Networks (CDNs) have recently been proposed to improve the performance (response times, bandwidth, and accessibility) of Internet-based content delivery through coordinated content replication [9]. CDNs maintain geographically distributed clusters of surrogate servers placed at the network edge that store copies of identical content, so that users' requests can be satisfied by the optimal surrogates. CDNs have also been demonstrated to be an highly efficient solution to deliver media streaming services over the Internet ranging from TV broadcasts to video on-demand [1].

However, modern 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. To deal with such new requirements, the more general Content Networks (CNs) are being introduced [10].

CNs can effectively support collaborative media streaming services ranging from collaborative content creation to collaborative media streaming delivery and control. In particular, CNs can provide 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 paper introduces a CN-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 paper describes the Hierarchical COoperative COntrol Protocol (HCOCOP) which enables the collaborative media streaming control. HCOCOP is mapped on the hierarchical control structure which is formed and supported by the CN-based architecture when a collaborative playback session is started. The control structure is formed by a coordination server at the root level, one or more control servers at the intermediate level and clients at the leaf level.

HCOCOP was implemented and evaluated through discrete-event simulation and, in particular, the performance evaluation phase, which involves balanced and unbalanced topologies of clients in the control structure and the three different versions of HCOCOP (NoCoop, LocalCoop, and GlobalCoop), allows to analyze three significant performance indices (blocking probability, denial probability and server load) which characterize the protocol performance.

The remainder of the paper is organized as follows. Section 2 provides an overview of the architectures for for group-oriented playbacks and, in particular, describes the CN-based architecture. Section 3 describes HCOCOP whereas Section 4 proposes the performance evaluation of HCOCOP. Finally, conclusions are provided and the main directions for future research delineated.

2. Architectures for Collaborative Media Streaming

A collaborative playback session (CPS) is a networked multimedia session in which a

collaborative playback service is provided to a synchronous and explicitly-formed group of clients. Several systems have been designed and implemented to date to provide CPSs. Their architecture can be classified as centralized and CDN.

The MBone VCR on Demand [6], the MASH Rover [12] and the ViCRO^C [4] systems rely on a centralized server which provides group organization, IP-multicast-based media streaming delivery and streaming control functionalities to groups of clients. 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. Such architecture therefore suffer two main issues: performance bottleneck represented by the centralized server and real deployment on the Internet due to the scarce availability of IP-multicast.

The COMODIN system [2] provides the same functionalities of the centralized systems but relies on a CDN-based architecture which not only allows for overcoming the abovementioned issues but also increase efficiency of the media streaming control with respect to the centralized systems. In the following subsection the COMODIN architecture is described.

2.1. The COMODIN architecture

The COMODIN architecture [2] is structured into two planes (Figure 1): 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.

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 [7].
- 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.

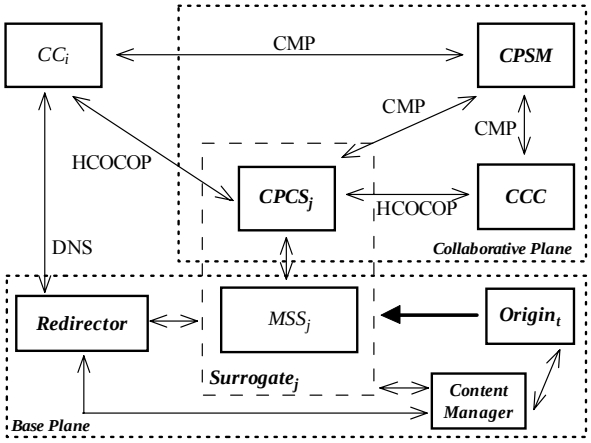

Figure 1. The COMODIN architecture.

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, consequently, is redirected to a CPCS.
5. *Join*. The CCs subscribers of the CPS join the CPS so becoming CPS members and, consequently, are redirected to their respective CPCSs.

6. *Execution*. The CPS is started by any member who issues the PLAY control request and evolves driven by a sequence of successive control requests (PAUSE, PLAY, SEEK).
7. *Termination*. The CPS can be terminated by its organizer CC.

The *Execution* phase, from the control point of view, is enabled by the HCOCOP which is defined in the next section.

3. HCOCOP

The Hierarchical COoperative COntrol Protocol (HCOCOP) is an extension of the COoperative COntrol Protocol (COCOP) [3] for CN-based media steaming architectures. It relies on the following characteristics: (i) random-based transmission policy of the streaming control requests; (ii) cooperation-based mechanism to reduce the transmission rate of likely unsuccessful control requests; (iii) soft state-based management of the control session state; (iv) FCFS policy for the acceptance of a control request.

In particular, HCOCOP allows for the selection of the control request sent by a client which will consequently change the state of a CPS (hereafter denoted as CPS_K). Figure 2 shows the control structure of CPS_K and the location of the automata which define the protocol. HCOCOP basically works as follows: if a client $C^{K,i}_x$ belonging to the subgroup G^K_i sends a control request (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 belonging to other subgroups G^K_J (with $j \neq i$) send a ClReq quasi-simultaneously. CCC_K accepts the first incoming ClReq, replies to all CPCSs, and discard other ClReqs for a given amount of time to regulate client interactivity and avoid session deadlocks. Possible conflicts generated by clients belonging to the same G^K_i are instead resolved by $CPCS^K_i$ which adopts the same policy as the policy adopted by the CCC_K.

HCOCOP can operate under three cooperation modes:

- global cooperation (*GlobalCoop*): the ClReq is forwarded downwards by the $CPCS^K_i$ to all other clients belonging to the subgroup G^K_i and by the CCC_K to all $CPCS^K_j$ ($j \neq i$) and then to all the clients of the other subgroups G^K_j (with $j \neq i$). Such mechanism allows clients to detect a ClReq sent by another client 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 other clients of the subgroup G^K_i;
- no cooperation (*NoCoop*): the ClReq is not forwarded to the other clients of the subgroup G^K_i nor to the other subgroups.

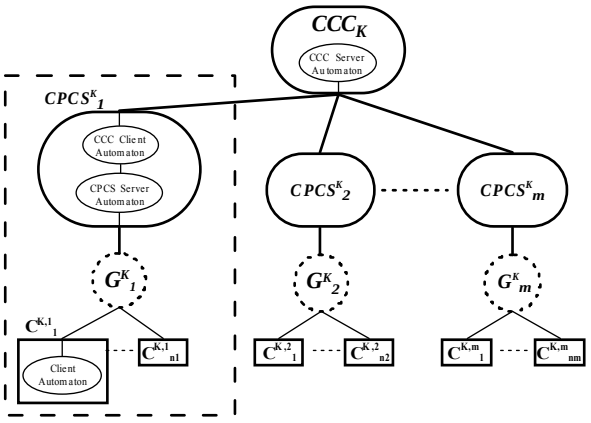

Figure 2. The control structure of CPS_K.

In the following the behavior of the HCOCOP automata is described.

The *Client Automaton* of the client $C^{K,i}_x$ generates a client request (ClReq) when the user issues a control request and its state is Ready, sends it to the the $CPCS^K_i$, and then passes into the RequestDone state. This state is also entered when the client $C^{K,i}_x$ in the Ready state senses a ClReq sent by another client belonging to the same subgroup (if *LocalCoop* is enabled) or to other subgroups (if *GlobalCoop* is enabled). In the RequestDone state, in which the automaton rests 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. When a Reply arrives it is processed and, to control the interactivity degree of the session, new user control requests are blocked until a given time T_{CC} elapses.

The *Server Automaton* of the $CPCS^K_i$ can receive a ClReq in the Ready state which makes it pass into the Synchro state. If the ClReq comes from the subgroup G^K_i (or *up* ClReq), such ClReq 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. If the ClReq comes from another subgroup (or *down* ClReq), such ClReq 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. Such delay is introduced both to make the clients aware of all changes in the session state, thus exploiting a soft-state like paradigm [11], and to regulate the group interactivity.

The *CCC Client Automaton* of the $CPCS^K_i$ when it receives a ClReq in the Ready state, performs the following operations: (i) if the ClReq is *up*, forwards it to the CCC Server Automaton, otherwise forwards the ClReq to its CPCS Server Automaton; ii) 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.

The *CCC Server Automaton*, when it receives a ClReq sent by the CCC Client Automaton of $CPCS^K_i$ in the Ready state, accepts such ClReq and forwards it to all the other CCC Client Automata, if global cooperation is enabled. A Reply is then sent to all the CCC Client 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.

4. Performance Evaluation

HCOCOP has been implemented in an event-driven, object-oriented simulation framework in order to evaluate its performance in CDN networks having different numbers of clients and different topologies. The aim of the simulation is to analyze the performance of HCOCOP and identify major benefits provided by the cooperation approach. In particular, the *NoCoop*, *LocalCoop* and *GlobalCoop* modes are analyzed and compared. Moreover, the performances are compared with those obtainable with a centralized architecture which exploits the basic COCOP protocol [3]. The centralized 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. 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; the automaton of the COCOP server process which resembles the CPCS Server Automaton of HCOCOP but does not have the Synchro state since there is no need to synchronize with other servers.

The performance indices used to analyze and compare the different protocols are defined in Table 1, 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 load of servers;

The denial probability Pden(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:

$$Pden(CPCS)+(1-Pden(CPCS))\cdot Pden(CCC)$$

where Pden(CPCS) and Pden(CCC) are the denial probabilities at the CPCS server and at the CCC server respectively. The server load indices are defined according to the CDN levels: 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.

Table 1. Performance indices of HCOCOP.

$N_{UsrReq(k)}$ $k\varepsilon\{CDN,Star\}$	Number of user requests issued in a CDN or Star	
$N_{UsrReqBlk(k)}$ $k\varepsilon\{CDN,Star\}$	Number of user requests blocked in a CDN or Star	
$N_{ClReq(i)}$ $i\varepsilon\{CPCS,CCC,Star\}$	Number of client requests delivered to the CDN servers (CPCS or CCC) or the Star server	
$N_{ClReqDis(i)}$ $i\varepsilon\{CPCS,CCC,Star\}$	Number of client requests discarded by the CDN servers (CPCS or CCC) or the Star server	
$N_{ClReqAcpt(i)}$ $i\varepsilon\{CPCS,CCC,Star\}$	Number of client requests accepted by the CDN servers (CPCS or CCC) or the Star server	
$Pblk(k)$ $k\varepsilon\{CDN,Star\}$	$\frac{N_{UsrReqBlk(k)}}{N_{UsrReq(k)}}$	*Blocking probability*: the probability that a user request is blocked in a CDN or Star
$Pden(i)$ $i\varepsilon\{CPCS,CCC,Star\}$	$\frac{N_{ClReqDis(i)}}{N_{ClReq(i)}}$	*Denial probability:* the probability that a client request is discarded by the CDN servers (CPCS or CCC) or the Star server
$SL(i)$ $i\varepsilon\{CPCS,CCC,Star\}$	$\frac{N_{ClReq(i)}}{T_{session}}$	*Server load:* the number of requests per second that a CDN server (CPCS or CCC) or the Star server receives

Main simulation parameters 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%).
- *User activity* (UA). It is characterized by MRIT (*Mean Request Interarrival Time*), i.e., the average inter-arrival time between two successive requests issued by the same user. MRIT is modelled according to a statistical model based on the *Gamma* probability distribution function [8]. User Activity can be 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 is varied within the range {10 s, 180 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 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. Normal distribution is chosen according to the considerations presented in [5]. The parameters of the delay model are set according to the values measured in a CDN testbed established across Italy and Spain [2]. 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 [3].

The simulation phase aims at evaluating the performance of the HCOCOP protocol in a simple CDN 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 [12].

First we will present results achieved in a CDN with a symmetric topology, than we will examine the behaviour of the protocol in an asymmetric topology and in an adaptive scenario in which a client is dynamically redirected from one subgroup to the other.

4.1. Performance of HCOCOP in a CDN with a symmetric topology

A first set of simulations have been performed in a symmetric CDN with 12 clients and 2 subgroups, with 6 clients per subgroups. In Figure 1, which shows the denial probability at the CPCS server, Pden(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 3), significantly decreases the denial probability. Benefits of cooperation are further enhanced under the *GlobalCoop* mode, since clients at the local CPCS are also able to detect a request issued by clients belonging to the other subgroup.

Figure 2 shows that the denial probability at the CCC server, Pden(CCC), is not appreciably modified by the cooperation approach. However, the values of overall denial probability, Pden(CDN), confirm the benefits of the cooperation approach, as shown in Figure 3. 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 clear in Figure 4.

Finally, the load of the Star server and of the CDN servers (CPCS and CCC) is reported in Figure 5. 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.

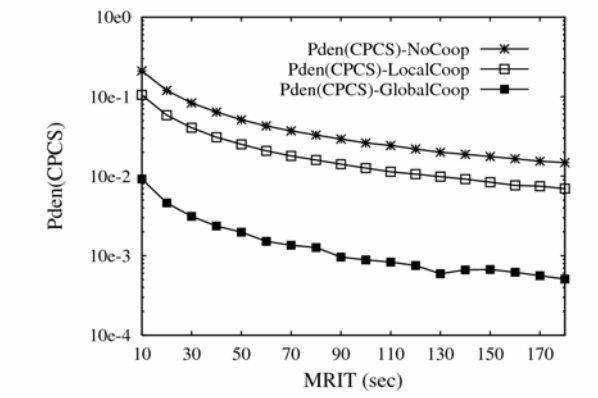

Figure 1. Denial probability at the CPCS servers. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

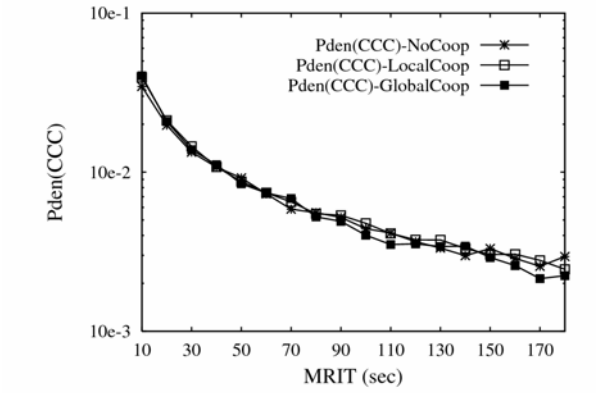

Figure 2. Denial probability at the CCC server. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

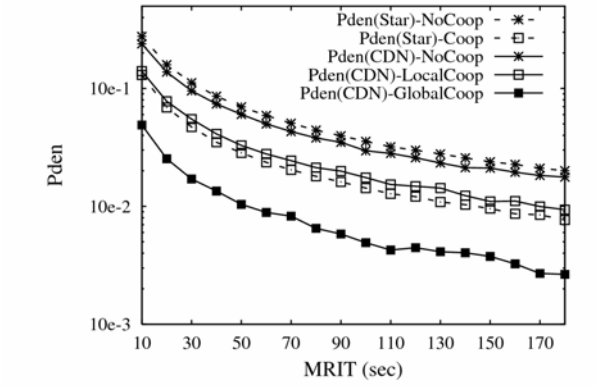

Figure 3. Overall denial probability. Comparison between CDN and Star architectures, with different protocol modes.

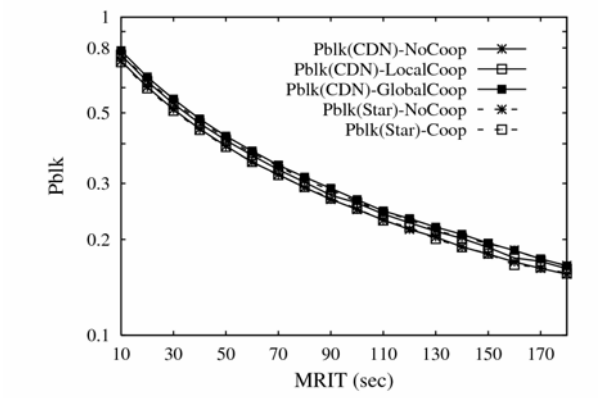

Figure 4. Blocking probability. Comparison between CDN and Star architectures, with different protocol modes.

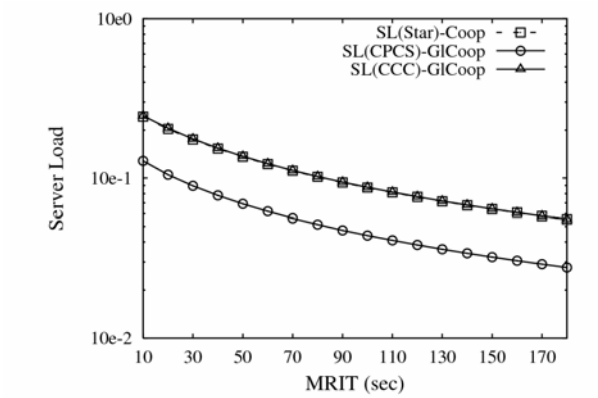

Figure 5. Server load. Comparison between CDN and Star architectures.

4.2. Asymmetric CDN topologies and effect of client redirection

A further set of simulation runs have been carried out to investigate the performance of a CDN in which clients are unevenly distributed among 2 CPCS servers. In particular, N=12 clients are allocated with 7 clients assigned to one CPCS and 5 to the other.

Figure 6 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 7-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. Conversely, no remarkable differences are noticed between the blocking probabilities experienced by clients of the 2 subgroups.

The server load for the asymmetric architecture is reported in Figure 7. The CCC server load is comparable to the Star server load. Furthermore, this figure shows that the load of a CPCS server is directly proportional the number of attached clients.

In a CDN, the *request-routing* algorithm is used to route a client request to an appropriate surrogate, which in our case corresponds to assign the client to a specific CPCS server and the related subgroup. In the case of adaptive routing [13], the surrogate can be dynamically chosen according to system conditions. Figure 8 shows the effect of dynamically moving 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 Figure A, the overall denial probability is decreased in the subgroup to which the client is redirected and is decreased in the other subgroup, whereas the denial probability related to the symmetric topology is in the middle. Conversely, if the initial configuration is the asymmetric one, the redirection of one client can be performed to obtain a better fairness among clients.

As opposed to the denial probability, blocking probability is hardly affected by client redirection. Furthermore, the CPCS server load, as mentioned in Section 4.1, is proportional to the number of clients, which means that client redirection can improve fairness also regarding to this performance index.

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

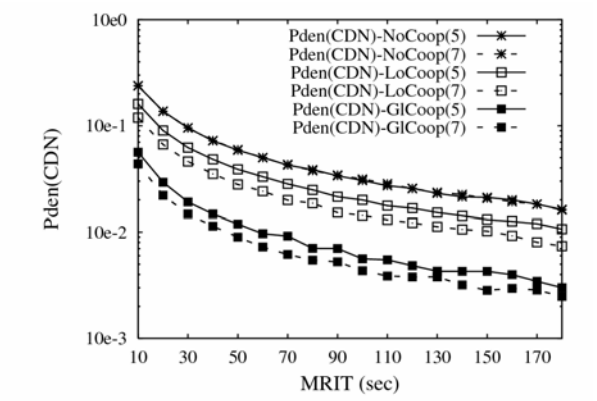

Figure 6. Overall denial probability in an asymmetric CDN architecture. Comparison among NoCoop, LocalCoop and GlobalCoop modes.

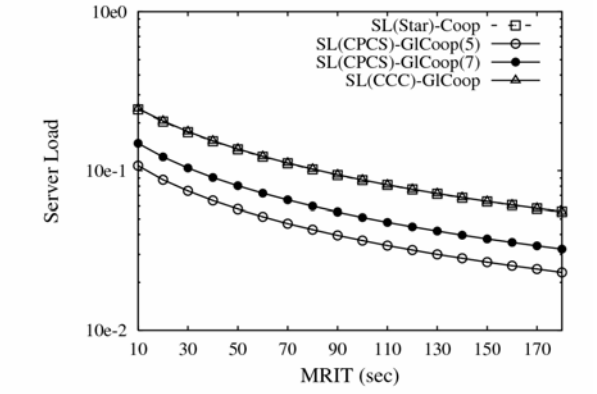

Figure 7. Server load in an asymmetric CDN architecture and in a Star architecture.

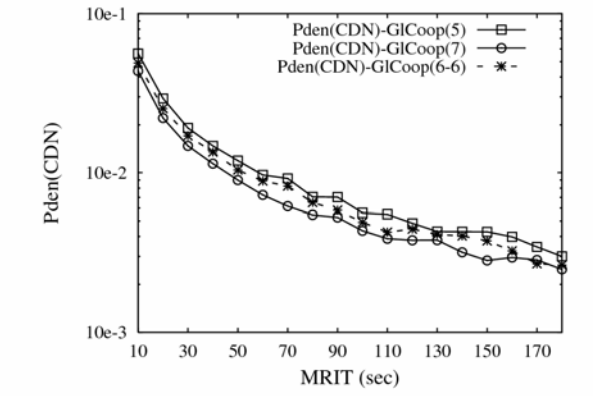

Figure 8. Effect of client redirection on the overall denial probability. Comparison among a symmetric architecture and an asymmetric one, obtained after redirecting a client from a subgroup to the other.

5. Conclusion

This paper presents a Content Network architecture that supports collaborative media streaming services and allows a synchronous group of users to select, watch and control a multimedia session. The control of the server is enabled through a hierarchical protocol whose performance was

evaluated through an event-based simulation. Results show that the hierarchical approach is highly efficient, when compared with the usually adopted centralized architecture, as denial probabilities are reduced while blocking probabilities and server load are 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, load of surrogates etc, can be exploited to tune the request routing algorithm, which is a pillar component of Content Networks.

6. References

[1] C.D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, C.J. Sreenan, and J.E. Van der Merwe, "Enhanced Streaming Services in a Content Distribution Network," *IEEE Internet Computing*, 5(4), pp 66-75, 2001.

[2] M. Esteve, G. Fortino, C. Mastroianni, C. Palau, and W. Russo, "CDN-supported Collaborative Media Streaming Control," *IEEE Multimedia*, 2007, to appear.

[3] G. Fortino, C. Mastroianni, and W. Russo, "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), pp. 823-839, 2005.

[4] G. Fortino and L. Nigro, "Collaborative Learning on-Demand on the Internet MBone," in *Usability Evaluation of Online Learning Programs*, Claude Ghaoui, Ed. Idea Group Publishing, Hershey (PA), USA, pp. 40-68, 2003.

[5] J.F. Gibbon and T.D.C. Little, "Use of Network Delay Estimation for Multimedia Data Retrieval", *IEEE Journal on Selected Areas in Communications* 14(7), pp. 1376-1387, 1996.

[6] W. Holfelder, "Interactive remote recording and playback of multicast videoconferences," *Proceedings of IDMS'97*, Darmstadt, Germany, September 1997.

[7] Molina, C.E. Palau, M. Esteve, I. Alonso, and V. Ruiz, "On Content Delivery Network Implementation," *Computer Communications*, 29(12), pp. 2396-2412, 2006.

[8] J. Padhye and J. Kurose, "Continuous Media Courseware Server: a Study of Client Interactions," *IEEE Internet Computing*, 3(2), pp. 65-72, 1999.

[9] G. Pallis and A. Vakali, "Insight and Perspectives for Content Delivery Networks," *Communications of ACM*, 49(1), pp. 101-106, 2006.

[10] T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, and G. Urvoy-Keller, "From content distribution networks to content networks – issues and challenges," *Computer Communications* 29, pp. 551-562, 2006.

[11] S. Raman and S. McCanne, "A model, analysis, and protocol framework for soft state-based communication," *ACM SIGCOMM Computer Communication Review*, 29(4), pp. 15-25, 1999.

[12] Schuett, S. Raman, Y. Chawathe, S. McCanne and R. Katz, "A Soft State Protocol for Accessing Multimedia Archives," *Proceedings of NOSDAV'98*, Cambridge, UK, July 1998.

[13] Wan. L. Wang, V. Pai and L. Petersen, "The effectiveness of request redirection on CDN robustness", *Proceedings of the 5th symposium on Operating systems design and implementation OSDI '02*, Boston, MA, USA, December 2002.