We propose a system called Comodin, based on a content distribution network (CDN), that provides collaborative media playback services. Specifically, the system enables Internet-based interactive media service for e-learning and e-entertainment, allowing an explicitly formed group of clients to view and cooperatively control a shared remote media playback.

Two big benefits of interactive collaborative multimedia services are the integration of media streaming with interactive group-based collaboration, as well as ease of service through the Internet. Another new benefit in the arena of Internet-based media service is cooperative playback service. This service provides an explicitly formed group of clients with cooperative viewing and control of a shared remote media playback.

In an effort to improve upon existing cooperative playback services, we propose a collaborative media on-demand platform, called Cooperative Media On-Demand on the InterNet (Comodin). This system efficiently provides collaborative playbacks on the current Internet infrastructure. It’s the first distributed multimedia system that’s based on a Content Distribution Network and provides interactive and collaborative multimedia services to a synchronous group of clients.

Initial considerations

When initially contemplating how to build Comodin, we considered the pros and cons of existing systems that attempt to fully or partially provide the collaborative streaming service (see the “Related Work” sidebar, page 62). An example is the MBone-based media-on-demand (MoD) systems, which basically provide single users with point-to-point transmission and control of selected media sessions and multiple users with point-to-multipoint transmission of selected media sessions. In the instance of multiple users, when a user wishes to control a media session that’s also being watched by other users, the following strategies can be adopted:

- Session splitting and proprietary control. The system clones the media session and transmits it to the user on an address that’s different from the original so that the user comes into control of the cloned session. The Interactive Multimedia Jukebox system exploits this strategy.

- Session control based on the initiator. Only the session initiator is qualified to control the session, whereas other users are merely passive viewers. The MBone VCR On-Demand system implements this strategy.

- Shared session control. Control actions, which any user can issue, induce session state changes affecting all session participants. The MASH Rover system (MARS) and the Cooperative Video Conference Recording On-demand system (Vicro) adopt this strategy.

The shared session control strategy is particularly interesting in that it enables collaborative playback sessions where a synchronous group of users collaboratively watches and controls a media playback. MARS—and particularly Vicro—are noteworthy examples of multimedia systems that have been built to fully support collaborative playback sessions. However, such systems are subject to the following drawbacks:

- limited efficiency and scalability of the media streaming delivery because of the media streaming server’s centralization (even if cluster based), and

- tight coupling to IP-multicast, which isn’t yet widely available on the current Internet infrastructure.

These drawbacks prevent the collaborative playback service from becoming a mainstream service on the current Internet. To overcome such drawbacks and provide an efficient use of and a ubiquitous access to collaborative playback services, we came up with the following solutions:
Exploiting an IP-based Content Distribution Network (CDN) for media streaming delivery. Recently, research focusing on Internet-based MoD has shifted from ad-hoc built systems to Streaming Content Distribution Networks (SCDNs) that can provide diversified media services more efficiently, ranging from TV broadcasts to video on demand (see the “Related Work” sidebar).

Exploiting IP multicasting where available. We can use IP multicasting in the core of the CDN and, where available, in the client access network.

Although the advantages of employing a CDN-based solution to improve media content delivery efficiency is widely confirmed, more complex coordination mechanisms are necessary for media streaming and streaming control with respect to centralized fully IP-multicast-based solutions. These mechanisms are to be introduced because media streaming servers are distributed across the Internet and close to potential users, which is typical of a CDN.

**Collaborative playback model**

Collaborative playback services let an explicitly formed group of clients cooperatively share both the viewing and control of a remote media playback. These services depend on the following main layers:

- organization, which allows for the explicit formation of a group of clients by clustering them in a collaborative playback session (CPS) and for CPS management;
- shared media and group-based control, which serves the same media session to group members and lets them control it through VCR-like control requests; and
- questioning, which lets group members synchronously exchange questions and answers.

Because the shared media and group-based control layer is the fundamental layer supporting CPSs, we defined, validated, and extensively analyzed (through discrete-event simulation) a model for collaborative playbacks based on a multicast client/server model. According to this model, a set of media clients (MCs), organized in a synchronous cooperative group (SCG) and logically linked to a media server (MS), can issue a control request to the MS which, upon acceptance of the request, replies to all MCs. The MS therefore holds the state of the playback session and changes it every time the MS accepts a control request.

Synchronizing the media playback viewing among all the MCs of a CPS is best-effort, since MCs could experience different delays from MS and, consequently, their viewings could be slightly desynchronized. Figure 1 shows the reference control schemas of the defined collabora-
Comodin is a novel multimedia system, based on a Streaming Content Distribution Network (SCDN), which provides collaborative playback sessions. With respect to the collaborative playback service it offers, Comodin relates to existing Internet Protocol (IP)-multicast-based cooperative playback systems; with respect to the media delivery infrastructure it employs, Comodin is related to research and commercial efforts on SCDNs.

**IP-multicast-based collaborative playback systems**

The Interactive Multimedia Jukebox is a system that implements a distributed video jukebox. IMJ uses the Web as a means to gather requests and present play schedules. Users can request playback, but have no means of interacting with the server to control the resulting schedule, cancel playback, or perform seek or pause operations. Control is performed only on buffered, per-client replicated data of the session. IMJ doesn’t provide real collaborative playback sessions because its control strategy follows the session-splitting and proprietary control format.

The MBone Video Conference Recording on Demand (MVoD) is a client–server system for interactive remote recording and playback of MBone sessions. The interaction between the system components is regulated by protocols for service access, stream control, session announcement, and interclient messaging. MVoD uses the control strategy of session control based on the initiator, so it only allows for restricted collaborative playback sessions.

The MASH Rover is a collaborative playback system that employs the Soft-State Archive Control Protocol, based on IP-multicast, which is used to remotely control media streams. The Vicro system is a full-fledged collaborative playback system that uses a multicast version of the Real-Time Streaming Protocol (RTSP) named Multicast Archive Control Protocol, or MACr, to allow for group-based control of media sessions. MASH Rover, Vicro, and Comodin all use the shared session control strategy; however, the former two are centered on a centralized control architecture based on an non-cooperative control protocol, whereas Comodin is centered on a hierarchical control architecture based on a cooperative control protocol.

**Streaming content distribution networks**

In this section, we summarize relevant research and commercial efforts on SCDNs, which are special-purpose content distribution networks designed to improve the streaming-based delivery of multimedia content to end users.

The PoRtal Infrastructure for Streaming Media (or Prism) is a research-oriented SCDN for distributing, storing, and delivering high-quality streaming media over IP networks. The Prism-based stored-TV (STV) service lets users select content based on the program name as well as the time it was aired. Content stored inside the network is accessible throughout the whole Prism infrastructure. The Base plane of Comodin adopts the aspect of session control.
architectural organization of Prism in the data, control, and management planes.

MarconiNet is a research-oriented architecture for IP-based radio and TV networks, built on standard Internet protocols including the Real-time Transport Protocol, RTSP, Service Announcement Protocol, and Session Description Protocol. 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 is a hybrid CDN/peer-to-peer network that 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 extendibility, 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.

The commercial SinoCDN system is a dedicated streaming CDN that uses the Intelligent Streaming Gateway, the CDN Manager, and the Media Domain Name System components, which constitute a comprehensive solution to provide content routing, delivery, and distribution of multimedia content. The Intelligent Streaming Gateway is based on cooperative caching and application-level multicast mechanisms.

Akamai (see http://www.akamai.com) is a worldwide commercial CDN provider which, after its fusion with Speedera, is currently the main CDN provider for video streaming delivered through a coordinated network of media proxies.

The Base plane, which is an open CDN platform primarily designed to deliver on-demand archived media streams. Thus, it's characterized by the following features: a request redirection algorithm based on monitored information, Java-based development targeting a multiplatform deployment; scalability to build small, medium, and large CDN systems; commercial, off-the-shelf technology; and the integration of the Darwin Streaming Media Server for the delivery of interactive audio–video streaming.

The Collaborative plane, which enhances the Base plane to provide collaborative playback services by adding components and protocols for group organization, group-based streaming control, and interactive questioning.

Moreover, we’ve implemented a fully Java-based multimedia application called the Comodin client as the user interface for the Comodin system.

Comodin relies on the collaborative playback model that was specifically customized for the CDN-based architecture within which the MS is implemented as a set of coordinated media streaming servers. The interfacing component between the Base and Collaborative planes is the media-streaming server (MSS), which has been enhanced to serve the same media stream to all the clients of the same CPS who can control it interactively.

**Base plane**

The Base plane architecture, shown in Figure 2 (next page), is comprised of three basic components:

- **Origin server.** This archives the media objects to be distributed by the CDN and delegates the uniform resource identifier (URI) name space for such objects to be distributed and delivered by the CDN system. The origin server distributes the delegated content to the CDN’s surrogates through the distribution network by means of bulk transfer mechanisms.

- **Surrogate.** This is a partial replica of the origin

References


server with the additional ability to temporarily store content and deliver it to clients through the access network. Surrogates can also share content—that is, the most popular content is more likely to be replicated among the surrogates.

**Client.** This is usually an individual PC that requests specific media content stored in the CDN. The request is issued through a Web browser while the viewing of the media object is supported by a QuickTime plug-in. We used a Domain Name System-based redirection to optimize CDN performance and eliminated the local DNS client cache of the clients so that new requests aren’t biased by previous DNS resolutions.

The surrogate component contains the following main modules:

- **Portal.** This is an HTTP-based server that provides access to the media content stored in the CDN and supports file transfer from origin servers to surrogates and among surrogates. The portal uses the $DB_{surgt}$ that contains a list of all available media objects and information for managing the CDN. Moreover, the portal can also redirect clients, who are already connected to a surrogate, to another surrogate.

- **MSS.** This is a Darwin streaming media server in charge of streaming requested media content to clients by using the Real-time Transport Protocol/Real-Time Control Protocol (RTP/RTCP) for media delivery and the Real-Time Streaming Protocol (RTSP) for media control. The MSS uses the media archive, which contains the media objects available at the surrogate.

- **Local monitor agent (LMA).** This is a Simple Network Management Protocol (SNMP) agent that periodically monitors the surrogate’s performance (for example, CPU and RAM usage, available disk space, and number of Transmission Control Protocol—or TCP—connections) and carries out on-demand network measurements for each given client.

The Base plane’s control mechanisms regulate the optimal surrogate selection, surrogate monitoring, and location and flow of content throughout the CDN. The main control compo-
ponents, shown in Figure 2, are the redirector and the content manager.

The redirector is the key component of the CDN as it selects the most adequate surrogate for each different client request. To perform the redirection task, the redirector uses two main modules:

- The global monitor agent (GMA), an SNMP-based server that queries the LMAs to obtain updated measurements of the surrogate load parameters and to estimate the network status (such as round-trip time—or RTT—and hops) with respect to the requesting client. The gathered data are stored in the database $DB_{Monitor}$.

- The redirection algorithm, which selects the optimal surrogate on the basis of the information gathered by the GMA plus that stored in the database $DB_{rdct}$. (See elsewhere for details about the developed redirection algorithm.)

The redirector is implicitly accessed through Java-based DNS servers (ad-hoc built) that are local to requesting clients. Such DNS servers accept client requests and forward them to the redirector. The redirector resolves them according to the redirection algorithm and responds to clients with the address of the most adequate surrogate serving the requested media content.

The content manager coordinates the media content storage with surrogates by determining

- the number of replicas of a media object,
- in which surrogate a new media object must be stored,
- evicting unpopular objects from the surrogates,
- transferring media objects from the origin servers to the surrogates,
- updating media objects in the surrogates when a new version is available to the origin servers, and
- transferring media objects among surrogates.

**Collaborative playback session manager.**
The CPSM is the component that provides the group organization functionality and also coordinates the other components of the collaborative plane. Group organization and component coordination are based on the CPS Management Protocol (CMP). In particular, group organization allows for the formation, (un)subscription, initiation, joining/leaving, and termination of CPSs. An organized CPS is identified by a unique identifier (or CPSId). Moreover, CPSM uses the CPS database ($DB_{CPS}$) to archive information about available media playbacks and organized CPSs.

**Collaborative playback control server.**
The CPCS, located in each surrogate and integrated with the MSS component, supports the remote control of the media streaming shared among the members of a CPS served by the same surrogate. To serve multiple CPSs, CPCS spawns a front end for each CPS whose members are served by the MSS. CPCS is based on the Multipolicy Collaborative Playback Control Protocol (MCPCP).9

**CPCS coordination channel.** The CCC is a component that coordinates the distributed...
CPCSs through the Coordination Channel Protocol (CCP). This CCC spawns a front end for each initiated CPS to coordinate the CPCS front ends serving the same CPS. Coordination among CPCS front ends is needed for deciding which CPCS front end is enabled to accept a control command, for synchronizing the CPCS front ends with respect to the media session time, and for allowing clients to join asynchronously.

**Comodin client.** The CC is a multithreaded Java-based multimedia application that interfaces the user with the Comodin system (see Figure 4). It consists of the following components:

- **MediaClient/MediaGUI,** which allows for the reception and presentation of RTP-based media streams that rely on the Java Media Framework.
- **ControlClient/ControlGUI,** which provide connection to the Comodin server side; browsing of the media archive; organization, (un)subscription, initiation, join/leave of a collaborative playback session; and collaborative control of a playback session.
- **CollabGUI,** which lets a user send and receive questions and filter incoming messages.

**CPS Management Protocol.** The CMP is a text-based stateful protocol designed upon RTSP. It supports the following interactions between pairs of components through appositely defined messages: <CC, CPSM>, <CPSM, CCC> and <CPCS, CPSM>.

The following messages characterize the interaction of the pair <CC, CPSM>:

- **DESCRIBE.** A client can receive either the list of available multimedia objects (request param=ALL) or information regarding a specific available multimedia object (request param=MPlid) through the DESCRIBE message.
- **ANNOUNCE.** A CPS is organized by an organizer client through the ANNOUNCE message that contains a description of the organized CPS based on the Session Description Protocol (SDP). Upon receiving the ANNOUNCE message, the CPSM indexes the session description through the CPSId and archives it in DB_CPS. The organizer client is automatically subscribed to the organized CPS.
- **CANCEL.** A client that has organized the CPS identified by CPSId can send a CANCEL message that contains the CPSId and the organizer’s authentication information.
- **DESCRIBE_CPS.** A client can receive either the list of the organized CPSs (request param=ALL) or information regarding a specific organized CPS (request param=CPSId) through the DESRIBE_CPS message.
- **SUBSCRIBE.** A client can subscribe to a given organized CPS, only before its initiation, through the SUBSCRIBE message that contains the CPSId of the CPS, and the subscriber’s authentication and personal information.
- **UNSUBSCRIBE.** A client can unsubscribe from a given organized CPS, only before its initiation, through the UNSUBSCRIBE message that contains the CPSId of the CPS and the subscriber’s authentication information.
- **INITIATE.** A CPS can only be initiated by its organizer client through the INITIATE message. Upon reception of INITIATE, the CPSM sends the CREATE message to the CCC and, after receiving the response, returns the URI of the media object to the organizer client.
- **JOIN.** A subscriber client can join an initiated CPS through the JOIN message that contains the CPSId and the client’s authentication information. Upon acceptance of JOIN, the CPSM returns the URI of the media object to the client (see the INITIATE message).
- **LEAVE.** A subscriber client can leave an initiated CPS through the LEAVE message that con-
tains the CPSId and the client’s authentication information.

The following messages characterize the interaction of the pair \(<\text{CPSM}, \text{CCC}>\):

- **CREATE.** Upon receiving the CREATE message, which contains the CPSId of an initiated CPS, CCC spawns a front end for the initiated CPS.

- **SESSIONSTATEINFO.** Every time the state of a CPS is changed, the CCC notifies the CPSM through the SESSIONSTATEINFO message.

- **GETSESSIONSTATE.** The CPSM can determine the current state of a CPS by sending the GETSESSIONSTATE message to CCC.

Finally, the interaction of the pair \(<\text{CPCS}, \text{CPSM}>\) is based on the LOOKUP message. When the joining client connects to the CPCS, the spawned front end of the CPCS sends the CPSM the LOOKUP message to obtain the URL of the CCC front end for the joined CPS.

**Multipolicy Collaborative Playback Control Protocol.** The interaction between CC and CPCS is based on the MCPCP that adopts a multipolicy-based approach for managing the control commands (for example, pause, play, seek, and stop) that are issued by clients to control the media playback. The management mechanism of the control commands is based on the following different policies according to each command’s semantics:

- **Random-based.** Group members can send a control command to the CPCS whenever they wish. The pause command is based on this policy.

- **Token-based.** A group member can transmit a control command only if they hold the token. The play and seek commands are based on this policy.

- **Voting-based.** A group member, who wishes to send a control command, triggers a voting procedure among the group members. If the voting procedure terminates successfully—that is, the majority of the clients accept the command—the command is forwarded. The stop command is based on this policy.

More details about the MCPCP can be found elsewhere.9

**Coordination Channel Protocol.** The interaction between CPCS and CCC is based on CCP, a text-based stateful protocol, which allows for the coordination of the CPCS front ends of the same CPS (hereby referred to as CPS\(_k\)). The defined messages of CCP are the following:

- **SYNCHROCC.** Before accepting a control command sent from a client connected to it, a CPCS front end of CPS\(_k\) sends the SYNCHROCC message, which contains the control command, to the CCC front end of CPS\(_k\). The CCC front end is able to resolve conflicts caused by the quasi-simultaneous transmission of control commands from clients belonging to CPS\(_k\) but connected to different CPCS front ends. The response to the SYNCHROCC message (SYNCHROCCREPLY) is multicast from the CCC front end to all of the CPCS front ends of CPS\(_k\).

- **SYNCHROMT.** To synchronize the start time of media streaming among the media-streaming servers of CPS\(_k\), the SYNCHROMT message is sent from each CPCS front end of CPS\(_k\) to the CCC front end when a session state change occurs. For example, when a seek command is accepted within CPS\(_k\), media streaming must quasi-simultaneously restart from the seek time at each media streaming server of CPS\(_k\). So, upon receiving the SYNCHROCCREPLY, the CPCS front ends of CPS\(_k\) send the SYNCHROMT message to the CCC to synchronize their media time clock to the reference media time clock of the CCC front end. Indeed, using this mechanism, the synchronization of the media streaming start time can be assured for the seek and play commands. To synchronize the viewings of all the CPS members after accepting a pause command, Comodin uses a client-based buffering technique.

- **SYNCHROAC.** To synchronize the state of a new client that joins CPS\(_k\) after the session has already begun, the SYNCHROAC message is sent from the CPCS front end of CPS\(_k\), to which the client is connected, to the CCC front end. This is necessary if, and only if, the creation of such a CPCS front end has been triggered by the late join of the new client.

**Performance evaluation**

The testbed on which the Comodin system is currently deployed is composed of three logical
subnetworks—the SATRD, LISDIP, and ICAR-CS intranets—connected through the Internet and respectively located at Sistemas y Aplicaciones de Tiempo Real Distribuido (SATRD), laboratory of the Universidad Politecnica de Valencia in Spain, the Laboratorio di Ingegneria dei Sistemi Distribuiti e Paralleli (LISDIP) of the University of Calabria in Italy, and the High-Performance Computing and Networks of the Italian National Research Council (ICAR/CNR) in Cosenza, Italy. Figure 5 shows only the components of the Collaborative plane with the following, specifically:

- The pairs <Surrogates A, Client Cluster A> and <Surrogate B, Client Cluster B>, which are respectively located at the ICAR/CNR and LISDIP intranets. Each Client Cluster is formed by a set of independent Comodin clients. Moreover, both the A and B surrogates contain media objects that can be selected to be streamed to a CPS.

- The CPSM and CCC servers located at the SATRD intranet.

We chose a symmetric topology (with respect to the CCC server) for the testbed: indeed, we measured comparable mean delays (about 64 milliseconds) between the Spanish SATRD intranet (where the CCC server is located) and both the Italian LISDIP and ICAR/CNR intranets (where the clients are located). Such symmetry enabled us to evaluate the performance of the Collaborative plane, in a case where all of the clients are equally distant from the CCC server.

We aimed for the performance analysis to evaluate a generic client’s ability to obtain control of the CPCS located at the local intranet and, afterwards, control of the CCC server located at the SATRD intranet. Performance evaluation focused on the efficacy of the random-based policy, because this is the base policy exploited by token and voting policies to grant the token and the right to issue a voting procedure, respectively.

If a client, who belongs to a CPS subgroup attached to a given CPCS, wins the contention of the CPCS with the other clients of the same subgroup, the request of such a client is accepted by the local CPCS and forwarded to the CCC server. The client actually gains control of the multimedia session only if his or her request also wins the CCC server’s possible contention; in this case, the CCC server changes the session state according to the client request and informs all the other clients about the change in session state through a multicast reply message.

We have analyzed the benefits and possible drawbacks of our proposal by comparing the performance of the CDN architecture with that of a one-server architecture, hereby referred to as Star, in which the media content is maintained and provided by a single MSS server located in Valencia with all the clients directly connected to it. The Star architecture is representative of existing collaborative playback architectures (see the “Related Work” sidebar) that have a centralized nature, as control messages are processed by a single server entity, and do not exploit cooperative control mechanisms.

Furthermore, for both Star and CDN architectures, we compared the cooperative (Coop) and non-cooperative (NoCoop) modes, to identify the benefits provided by the cooperative approach. Therefore, the analysis of the testbed performance not only allows for an evaluation of the two specific enhancements discussed in this article (CDN versus Star, Coop versus NoCoop), but also for appreciating the overall performance improvement that we can achieve by exploiting both the innovative features (with respect to a classical centralized control protocol that doesn’t use any cooperation mechanism).

We evaluated several performance indices, listed and defined in Table 1, for the testbed (see also Fortino et al.7). Here we only report results concerning the denial probability—that is, the
probability that a client request is not actually served, because either the local CPCS or the remote CCC server denies the request. Indeed, the denial probability is the most important performance index as users, aware that they aren’t always able to control the server, will be more tolerant of the inability to send a control request (the blocking event) than of the rejection of a forwarded control request (the denial event).

We used the following parameter settings in the testbed:

- The number of clients (NC) was varied from 2 to 24 so as to consider small-, medium- and large-sized groups of users. It’s worth noting that a cooperative playback session is, on average, comprised of 2 to 16 members. In the CDN architecture, clients were distributed among the two subnets (see Figure 5) with both even and uneven distributions.

- User activity was modeled according to a Gamma distribution of user requests. In particular, the average interarrival time between two successive requests issued by the same user (Mean Request Interarrival Time, or MRIT) was varied from $10$ to $180$ seconds with a step of $10$ seconds, to analyze the Cooperative plane in sessions with medium or high user activity.

- The server timer, used to control the server reactivity and the overall degree of system interactivity, was set to 3 seconds to allow for high interactivity; the client timers were set to the same value, to avoid the possible occurrence of deadlock situations, as described elsewhere.

Figure 6 shows the values of the denial probability, obtained in both the CDN and Star architectures, for different values of the user request rate and a client number of 16; within the CDN architecture, clients are evenly distributed (8 and 8) between the two Italian intranets. As all clients are equally distant from the server and experiencing similar performances, in Figure 6 we show the denial probability of a generic client. Figure 6 also compares the results of Coop and NoCoop, and clearly shows that in both the Star and CDN architectures the use of Coop decreases the denial probability.

Moreover, for each defined mode (Coop and NoCoop), the denial probability is lower in the CDN architecture than in the Star architecture. This outcome highlights that the distribution of media content among surrogates, besides improving the performance of media delivery, also improves performance in the collaborative plane. Finally, by comparing the Star-NoCoop curve with the CDN-Coop curve, it’s interesting to notice the significant reduction of the denial probability (between $50$ and $60$ percent) that’s achieved through the combined adoption of the CDN architecture and the cooperation mechanism.

Figure 7 shows the denial probability—with MRIT equal to $90$ seconds—obtained with the Star and CDN architectures for different numbers of clients (evenly distributed in the CDN architecture) and adopting both Coop and NoCoop.

<table>
<thead>
<tr>
<th>Table 1. Performance indices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
</tr>
<tr>
<td>Pblocking</td>
</tr>
<tr>
<td>Pdenial</td>
</tr>
<tr>
<td>L_net</td>
</tr>
<tr>
<td>L_ser</td>
</tr>
</tbody>
</table>
When NoCoop is used, adoption of the CDN architecture is advisable, and the corresponding improvement in performance actually increases with the number of clients.

However, the denial probability experienced with Coop is far lower, which confirms that Coop offers better performance. The comparison of the CDN and Star architectures when Coop is adopted demonstrates that the advantage provided by the CDN architecture is appreciable for medium- and large-sized groups. For small-sized groups (fewer than six clients), the overhead imposed by the CDN architecture is higher than obtainable benefits, thus making it more efficient to use the one-server architecture.

Figure 8 shows the results of an uneven distribution of clients. Here, the CDN architecture contains 16 clients, among which 10 are located on the LISDIP intranet and six are located on the ICAR/CNR intranet. The Star architecture also contains 16 users, all directly connected to the CCC server. Comparison with the results reported in Figure 6 shows that only the 10 clients on the LISDIP intranet experienced an improvement in performance with the CDN architecture, while the six clients located on the ICAR-CS intranet encountered a denial probability similar to that experienced by clients of the Star architecture.

Indeed, when a client belonging to the most-populated group gains control of the local CPCS, it has a higher chance of controlling the CCC server than clients belonging to the other group. This phenomenon occurs only when the cooperative mode is adopted and can be considered a beneficial outcome of the cooperation mechanism, because it favors the subnet in which cooperation among clients is better exploited (see also Fortino et al.). Therefore, clients should join existing groups because isolated clients or clients belonging to small groups are penalized with respect to the clients of larger groups.

Conclusions
In this article, we have described the Comodin system, a CDN-based distributed multimedia system that provides interactive and collaborative multimedia services to a group of users. Our evaluation of Comodin’s performance indicates that, with respect to current state-of-the-art solutions based on centralized servers, CDN-based solutions can be more effective. Indeed, Comodin not only improves the media content delivery—which is the main benefit provided by a CDN—it also increases the performance of the streaming control.

Ongoing research includes the design of a redirection mechanism centered around group awareness. This ability to redirect a client to a given CDN surrogate is also driven by the client’s membership in a given CPS.

Acknowledgments
This work was carried out in the bilateral research project IT-592/HI-0185, titled “Development of an IP-based Cooperative and Content-Enriched Media on-Demand System Supporting Distance Learning and Entertainment,” and partially funded by the Spanish and Italian governments in the framework of the Spain/Italy Integrated Actions program.

References
7. G. Fortino, C. Mastroianni, and W. Russo,
   “Cooperative Control of Multicast-Based Streaming
   On-Demand Systems,” Int’l J. Grid Computing:
   Theory, Methods, and Applications, vol. 21, no. 5,
8. B. Molina et al., “On Content Delivery Network
   12, 2006, pp. 2396-2412.
9. G. Fortino, C. Mastroianni, and W. Russo, “A Multi-
   Policy, Cooperative Playback Control Protocol,”
   Proc. 3rd IEEE Int’l Symp. Network Computing and
10. C.D. Cranor et al., “Enhanced Streaming Services in
    a Content Distribution Network,” IEEE Internet
11. G. Fortino et al., “Development and Validation of a
    Multicast Client/Server Model for Cooperative
    Control Sessions,” Proc. IASTED Comm. and Computer

Giancarlo Fortino is an assistant
professor of computer science at
the Department of Electronics,
Informatics, and Systems of the
University of Calabria, Italy. His
research interests include distrib-
uted computing, Internet com-
puting, multimedia systems, object- and agent-oriented
technology and systems, and applied software engi-
neering. Fortino received a Laurea and PhD in com-
puter engineering from the University of Calabria.

Wilma Russo is a professor of
computer science at the Depart-
ment of Electronics, Informatics,
and Systems at the University of
Calabria. Her research interests
include parallel and distributed
computing and systems, agent-
oriented systems, object-oriented technology, and
Internet computing. Russo received the Laurea degree
in physics from the University of Naples, Italy.

Carlos E. Palau is a associate pro-
fessor at the Escuela Tecnica Superior de Tele-
comunicacion at the Universidad Politecnica de Valencia, Spain.
His research interests include real-time systems, multimedia sys-
tems, and content distribution networks. Palau received
his MSc and PhD degrees in telecommunications engi-
neering, from the Universidad Politecnica de Valencia.

Manuel Esteve is a professor in
the Escuela Tecnica Superior de
Ingenieros de Telecomunicacion
at the Universidad Politecnica de
Valencia, Spain. His research
interests include real-time sys-
tems, multimedia systems, and
content distribution networks. Esteve received his MSc
and PhD degrees in telecommunications engineering,
from the Universidad Politecnica de Valencia.

Readers may contact Giancarlo Fortino at g.fortino@unical.it.

For further information on this or any other computing
topic, please visit our Digital Library at http://computer.org/publications/dlib.