MUMOC: an Active Infrastructure for Open Video Caching

Paolo Bellavista, Antonio Corradi, Luca Foschini
Dipartimento di Elettronica Informatica e Sistemistica - Università di Bologna
Viale Risorgimento, 2 – 40136 Bologna – ITALY
Phone: +39-051-2093001; Fax: +39-051-2093073
{pbellavista, acorradi, lfoschini}@deis.unibo.it

Abstract

Advances in networking and content delivery systems are enabling new challenging provisioning scenarios where a growing number of users access Video on Demand (VoD), possibly while moving among different points of attachment to the Internet and using different access terminals. This calls for novel middlewares capable of supporting personalized VoD access by dynamically activating intermediate nodes between VoD servers and clients. The paper proposes MUMOC, a dynamic and flexible overlay infrastructure for the distributed caching of both VoD prefixes and VoD metadata. To achieve openness and easy interoperability with legacy VoD services, MUMOC adopts standard XML-based formats, based on both Dublin Core and MPEG7, to represent VoD metadata. First experimental results show that MUMOC significantly reduces bandwidth utilization and user perceived delays. In addition, notwithstanding the dynamic building of the overlay network and the application-level approach, the introduced overhead is compatible with the strict requirements imposed by multimedia distribution over the best-effort Internet.

1. Introduction

The provisioning of multimedia data over the best-effort Internet is still a very challenging service scenario. Peer-to-peer file sharing applications such as Gnutella [1], Kazaa [2] and Pastry [3] have addressed the issue of storing large amounts of (mainly multimedia) data in a largely decentralized way, also by trying to provide online content preview facilities for downloaded data, e.g., Kazaa. These solutions have further widened the interest in multimedia data sharing over the Internet, up to the current scenarios where these applications produce most Internet traffic [4]. The problem is significantly more complex for Video on Demand (VoD) streaming applications over the best-effort Internet, where, for instance, users demand for low playback delays and for the maintenance of the negotiated quality during provisioning, notwithstanding the Internet latency and loss rates.

In addition, the diffusion of networked computing environments at home/office/open public spaces and the proliferation of wireless-enabled portable devices identify novel provisioning scenarios where a huge number of users are willing to have ubiquitous and continuous access to streaming services. Users demand the possibility not only to move among different Internet attachment points but also to change their access terminals, while having the service adapted to the runtime characteristics of their access environment and while preserving their session state.

All above issues motivate novel and highly flexible middleware-level approaches to support service provisioning [5]. Dynamic distributed infrastructures can support multimedia content dissemination and adaptation efficiently, with an active participation not only of service end-points but also of some intermediate nodes along the path between clients and servers. In other words, middleware should support active services, i.e., services resulting from the cooperation of interworking middleware/service components distributed along service paths, between clients and servers, and traversed by service flows [6]. In particular, active services can improve VoD streaming over the best-effort Internet, to primarily support *prefix caching*, i.e., the online caching of the initial part of VoD flows at intermediate traversed nodes, to allow *fast playback startup*, i.e., the reduction of user-perceived VoD startup delays in the case of prefix cache hit in the active infrastructure, and to achieve *interoperability*, by providing open and standard representations of the available VoD flows so to simplify the interworking with legacy VoD systems.

The paper presents **MUM** **O**pen **C**aching (MUMOC), an active infrastructure for distributed caching that significantly extends our **M**obile agent-based **U**biquitous multimedia **M**iddleware (MUM) with the possibility of caching multimedia prefixes and metadata descriptions (for both VoD prefixes and full VoD flows) in a highly interoperable way [6].

MUMOC automatically deploys its caching mid-

dleware components only where needed during provisioning, depending on client location. The caching of whole VoD flows would rapidly exhaust the storage capacity of proxy nodes; therefore, multimedia caches usually store only portions of VoD streams at proxies. Similarly, the primary idea in MUMOC is to support the distributed replication and storage of the first few seconds (*prefix*) of popular VoD contents at intermediate nodes along client-server paths. In particular, when a client requires a VoD flow, MUMOC routes the request to the server, intercepts the incoming VoD flow, and caches its prefix on a proxy node in the locality currently providing network connectivity to that client. In addition, MUMOC caches the metadata describing the available VoD contents: a distributed cache disseminates VoD metadata with the goal of reducing the response time for VoD retrieval. MUMOC metadata are represented in an interoperable XML-based format that combines the Dublin Core standard and the MPEG7 one, to increase the openness of the MUMOC solution and to simplify its integration with legacy VoD services.

2. MUM Active Middleware Overview

MUMOC provides open caching of VoD prefixes and metadata, and is integrated with the MUM active middleware for dynamic Quality of Service (QoS) tailoring and adaptation of multimedia flows. This section gives a brief overview of MUM design guidelines and functions to provide the needed background for fully understanding the MUMOC proposal. A more detailed description of MUM is out of the scope of the paper and can be found in [6].

The best-effort Internet lacks the infrastructure support for the effective delivery of information streams and much ongoing research is struggling to solve the problem [8]. Ubiquitous computing further complicates that scenario: possibly mobile users are willing to access their services ubiquitously, to switch their working sessions at runtime among terminals with very different capabilities, and to maintain their sessions open even while they are temporarily disconnected. All these issues call for a middleware-level support capable of dynamically extending the traditional Internet infrastructure, where and when needed depending on the client points of attachment, and has motivated MUM design and implementation.

A primary design guideline in MUM is to exploit proxy middleware components that can dynamically deploy along the client-server path, thus activating some intermediate nodes traversed by VoD flows (*active middleware*). MUM proxies participate in service delivery by playing the dual role of client and server, by passing in a pipeline the service data received from the previous middleware component to the next one along the service path. The interposition of proxies along distribution paths starts to be recognized as an effective solution for ubiquitous service provisioning over the next generation Internet [7, 9]. The MUMOC solution for VoD caching permits to maintain VoD prefixes and metadata locally to one activated intermediate node, and integrates with the dynamically deployed infrastructure of MUM proxies, as better described in the following.

The MUM middleware is implemented on top of our SOMA Mobile Agent (MA) platform [10]. The MA-based implementation permits to move both middleware components and session state at provision time, depending on the location of client requests of VoD flows. The MA programming paradigm is recognized as an effective solution for middleware supports in mobility-enabled provisioning environments because of the MA properties of mobility, asynchronicity, and autonomy [7, 11].

3. MUMOC Solution for Open Caching

Let us introduce the functions of the MUMOC middleware by sketching an actual VoD streaming usage scenario, depicted in Figure 1. Networked colleges and libraries in a university campus are interested in collaboratively providing a VoD streaming service with all the recorded lessons of the last semester. For instance, imagine that one CS210 lesson has been held in the morning in the engineering college and stored in the college VoD server. In the afternoon, Bob goes to the central library and would like to access, from its Wi-Fi laptop, the recorded morning lesson he missed. Bob's request produces the streaming of the VoD flow from the engineering college VoD server to the network locality that currently provides connectivity to Bob's laptop, step 1 in the figure. MUMOC not only enables Bob to rapidly find the VoD flow with the suitable QoS characteristics, but also exploits Bob's request to operate, as a side effect of VoD service provisioning, the prefix caching of the lesson at the central library proxy cache.

Suppose that later in the afternoon Alice goes to the engineering library with her Personal Digital Assistant (PDA) and asks for the same VoD because she has not well understood some points of the morning lesson. The MUMOC active middleware can browse metadata describing the disseminated VoD prefixes and the full VoD flows; MUMOC can discover a node close to the client, i.e., the central library, that holds a prefix of the requested VoD content, but with a frame size that is too large for Alice's limited PDA

display. In that case, MUMOC first commands the prefix downscaling and streaming from the central library proxy, and possibly activates the caching of the downscaled prefix at the engineering library node (step 2). Then, MUMOC downloads and downscales the rest of the flow (*suffix*) from the engineering college and forwards it to Alice's device (step 3).

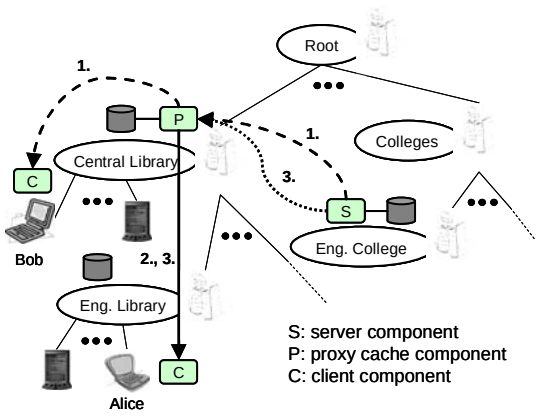

Figure 1. MUMOC at work in the university campus usage scenario

As exemplified by the usage scenario, MUMOC can assist multimedia delivery in integrated wired-wireless networks where wireless access points extends the accessibility of the traditional wired Internet infrastructure. The core of the MUMOC distribution network consists of a set of fixed hosts interconnected by wired broadband LANs; wired hosts compose an overlay network logically organized as a tree. Hosts are structured in location abstractions defined within the MUM framework: tree, leaves, and paths [6]. Client terminals, with possibly differentiated (and limited) local resources, access the core overlay either via wired or wireless connectivity, and can only play the role of leaves in the VoD distribution tree.

MUMOC aims at reducing server-proxy transmission costs (primarily bandwidth consumption and client-perceived delay for VoD startup) not only via prefix caching in the client vicinity, as sketched above in the example, but also via an original batching solution. Traditional batching techniques are server-side and batch consecutive requests received within a specified time interval [12]. MUMOC, instead, employs a proxy-based batching technique that takes advantage of cached VoD prefixes at proxy nodes. In the following, we will indicate this technique as *Suffix Batching (SBatch)*. [13] proposes a similar solution based on SBatch and proxies; it also shares with MUMOC the assumption that, in a general Internet service provisioning scenario, service paths are usually only unicast-enabled, in contrast with many other batching techniques that require the ubiquitous availability of multicast support [12].

SBatch activates when a proxy node receives a request for a prefix stored locally and, before the prefix streaming to that client ends, it receives requests from other clients located in the same locality. By referring to the university usage scenario, suppose that a first request for the CS210 lesson is received by the central library node. In case of cache hit, the prefix caching node in the client locality immediately starts streaming the VoD prefix to the client, and properly schedules suffix download from the server to avoid client-side visualization discontinuity when the prefix terminates. For successive requests of the same VoD flow from clients in the same locality, SBatch suggests providing those clients immediately with the locally cached prefix and anticipating the suffix transmission to them (on a separate channel) as soon as the suffix starts to be received at the proxy node. Let us rapidly note that this approach focuses on reducing transmission costs on the server-proxy path, by assuming that local proxy-client transmission costs are significantly lower, as it is in wired-wireless integrated networks as the one of the campus-wide streaming scenario. However, the SBatch solution proposed in [13] requires large storage capacity at client devices to buffer suffixes and is unsuitable for wired-wireless deployment environments. Section 4.2 details the original extensions that MUMOC proposes to the traditional SBatch strategy.

Finally, MUMOC implements a distributed and replicated VoD metadata repository both to provide high metadata availability and to reduce the client-perceived delay for the startup of the requested VoD flow. In fact, VoD metadata are crucial in the MUMOC middleware: for instance, anytime a client requires a multimedia presentation, MUMOC looks for the corresponding metadata in the distributed repository to determine where the VoD flow is available and with which QoS characteristics; VoD metadata are also exploited to guide service adaptation and to support session continuity [6, 7]. Given the frequent usage of VoD metadata in MUMOC operations, our middleware decides to perform metadata caching potentially at any wired node of the overlay infrastructure and to maintain a high replication degree for metadata of frequently requested VoD flows, as better explained in Section 4.2.

4. MUMOC Design and Implementation

This section first presents the architecture of the MUMOC active middleware and, then, describes its

primary design and implementation choices. Additional implementation details and the code of the MUMOC prototype are available at `http://lia.deis.unibo.it/Research/MUM/`

4.1. The MUMOC Architecture

The MUMOC middleware is organized in two main modules, local prefix cache and metadata browser, organized in two layers (mechanisms and facilities), as depicted in Figure 2.

Given a multimedia title, description, and possibly hints about the desired VoD quality, the Metadata Browser (MB) module returns the full metadata about the available multimedia presentations that match the searching criteria. The module offers functions for metadata querying, inserting, and deleting. Any MB component locally hosts a partial replica of XML-based metadata for the available VoD contents (see Section 4.4) and can coordinate with other distributed MB browser components to retrieve the needed VoD description. In particular, local MUMOC metadata repositories extend MUM repositories with new functions to store metadata according to standard, open, and extensible formats [6].

The local prefix cache module maintains prefixes at intermediate nodes along the client-server paths. At the mechanisms layer, the module consists of two different components devoted to the caching of VoD prefixes and to the SBatch enforcement.

Let us rapidly observe that MUMOC modules are implemented in terms of MAs and this implementation permits to achieve extreme flexibility and extensibility. For instance, as better detailed in the following section, MUMOC extends the base SBatch strategy to support also client devices with limited memory capabilities (insufficient for suffix buffering) by dynamically interposing an MA between the client and its prefix caching proxy; that MA, called SBatch Shadow Proxy (SBSP), acts over the fixed network on behalf of the limited client device and performs the needed suffix buffering in the client vicinity.

4.2. Content Cache

The content cache module offers functions for the online VoD prefix download, enforces VoD prefix replacement strategies, and implements SBatch and the different original extensions to it proposed in MUMOC. The caching-enabled proxy is the content cache core. MUMOC dynamically deploys it; there is one caching-enabled proxy for each served VoD flow to achieve per-flow adaptation and tailoring. The caching-enabled proxy exploits three main components, as shown in Figure 2: cache manager, cache writer, and transmission scheduler. In addition, the Proxy Agent, which is a middleware component part of the MUM architecture [6], realizes the necessary signaling between client, proxy, and server, e.g., during SBatch execution it sends to the caching-enabled proxy the endpoint of the communication channel used by the served client to receive the suffix.

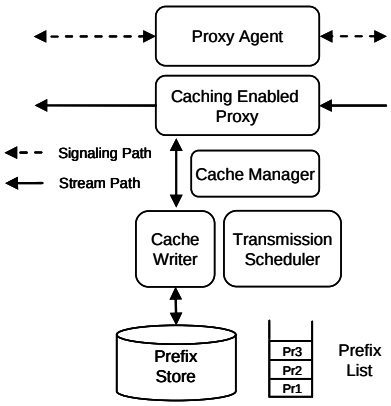

Figure 2. The architecture of the MUMOC content cache

The cache manager offers its APIs to the caching-enabled proxy. In particular, apart from methods for cache writer access, the cache manager exposes callbacks to notify the caching-enabled proxy of relevant events, such as the arrival of the suffix flow at the proxy. The transmission scheduler handles requests and schedules both server-proxy and proxy-client transmissions. The cache writer intercepts incoming VoD flows and saves them to the prefix store in the local file system. Let us note that MUMOC intercepts the VoD flow transmitted from the server to the client and does not require any additional transmission.

As demonstrated from both analytical and experimental results, prefix caching, especially when used in conjunction with reactive transmission schemes, reduces client perceived startup delay and bandwidth occupation [14]. Prefix length does not significantly influence the reduction of transmission costs [13]. Therefore, MUMOC chooses to adopt a fixed prefix length of 10 seconds, while the minimum cache slot at caching-enabled proxies is of 3 seconds. In fact, previous experiments accomplished within the MUM project have demonstrated that an interval of 3 seconds is sufficient to configure proxy-server service path within most common application scenarios [6]. About cache replacement, MUMOC implements the Proportional Priority strategy, a greedy algorithm assigning to each prefix a number of cache slots proportional to the product of the VoD size and its popularity. This strategy achieves sub-optimal results, but

requires very simple processing based on a very limited set of information [13].

SBatch requires client devices to open two communication channels with the proxy: one is used to get the prefix, while the other pre-fetches the suffix flow and buffers it locally at the client node. When the prefix visualization has ended, the client goes on with suffix download and starts employing the buffered VoD data to continue the multimedia visualization without interruptions. MUMOC proposes and implements a significant original extension to the above technique in order to support client devices with limited storage capabilities, as introduced by the example at the end of Section 4.1. In the case that the VoD flow request can be served by employing SBatch, MUMOC first deploys and activates the SBSP agent on a wired node in the network locality where the client is currently connected; then, MUMOC builds the extended service flow pipeline including, in this case, the server, the proxy, the SBSP agent, and the client; finally, it commands to start the prefix streaming. Then, when the suffix starts to be received at the proxy, the suffix is also forwarded to SBSP in charge of its buffering; before the end of the prefix transmission to the client, SBSP merges prefix and suffix in order to avoid perceivable interruptions of the VoD flow at the client. Note that this original extension to SBatch does not require client buffering at all and that the client device requires only one transmission channel towards SBSP.

Finally, let us observe that, when in ubiquitous computing middleware-based VoD transformations are needed along the client-server path, e.g., to dynamically downscale a VoD flow to fit the specific hardware/software characteristics of the served client terminal, it would be desirable to cache a prefix of the adapted VoD flow to allow fast playback startup in the case of new requests from similar client terminals. MUMOC recognizes such opportunity and is capable of locally caching also adapted prefixes, by exploiting the on-the-fly VoD transcoding functionality provided by MUM [6]. In addition, in that way MUMOC can take advantage of the cached adapted prefix to begin suffix adaptation in advance, thus smoothing the issues due to long service path configuration and complex VoD transcoding.

4.3. Metadata Browser

The distributed MUMOC Metadata Browser (MB) exploits the node organization in location abstractions. One MB component runs on each tree node and may communicate with other MBs on parent and children nodes, as depicted in Figure 3. Each node maintains metadata for all the VoD presentations stored in its sub-tree. For instance, metadata stored at A1 include all metadata replicated at nodes from B1 to Bn. As a consequence, the root node caches metadata for all the presentations stored in the system. Let us observe that this does not generate an excessive load for the root node because VoD metadata have very limited size, especially if compared with VoD prefixes. For instance, the size of a 10-second prefix of an MPEG4 VoD flow, frame size 350x240 and frame rate 14 frames/second, is about 800kB, while the associated metadata does not exceed 2kB.

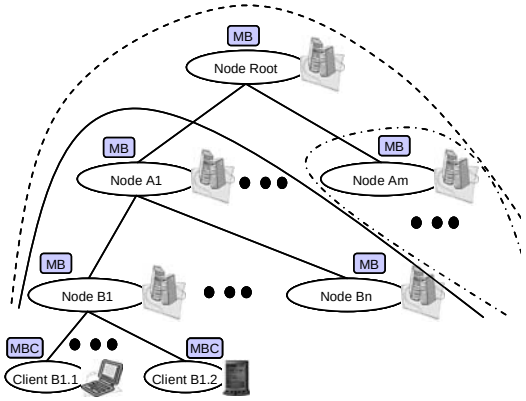

Figure 3. The MUMOC Metadata Browser

In addition, MUMOC stores the root node metadata by exploiting a Lightweight Directory Access Protocol (LDAP) service, which may be deployed over different distributed nodes in case of large number of available VoD flows and consequent large size of the corresponding metadata. MUMOC clients, instead, exploit a lightweight client, called Metadata Browser Client (MBC), simply to access disseminated VoD metadata, with no possibility of locally hosting a metadata repository, in order to minimize the utilization of usually limited client storage.

A MUMOC client looking for a specific VoD title initiates a metadata query. If the VoD title is not present at the metadata repository on the wired node where the client is currently attached, the query is propagated recursively up in the MUMOC tree until one MB component can respond. Thus, both network traffic and response time are reduced in case of metadata cache hit, and that contributes to accelerate playback startup.

When a VoD prefix elimination occurs, all corresponding VoD metadata must be discarded over the MB distributed implementation. The removal protocol proceeds as follows: first, VoD metadata are discarded from the node where the VoD prefix was stored; then, the delete request is forwarded up to the

root. In the case of insertion of new VoD metadata, MUMOC immediately propagates the metadata from the node where the VoD flow has been added to the LDAP-enabled root node. This increases VoD metadata availability, by making the new metadata rapidly visible in all the deployment environment.

In addition, MB offers functions to evaluate the convenience of caching a specified VoD prefix at one node. In particular, the browser controls if another VoD prefix with the same characteristics is already present in any proxy cache node reachable within *n* hops. For instance, if a request for a specified VoD prefix occurs at node B1 and *n=1*, the browser checks whether the associated prefix is already stored at nodes B1 and A1. Only if there is no already cached prefix, the VoD prefix is replicated and saved at B1. The VoD metadata in the MB distributed repository are represented according to a standard and interoperable format, which is illustrated in the next section.

4.4. Metadata Implementation

The high heterogeneity of multimedia formats, transport protocols, and storage solutions motivates the definition and adoption of standards that can describe multimedia contents independently of any specific technology and commercial product. For sake of openness and interoperability, MUMOC combines MPEG7 by Multimedia Picture Experts Group (MPEG) [15] and Dublin Core (DC) [16] representation formats to describe the VoD flows available in the distribution network. On the one hand, the DC adoption permits to be largely interoperable with a large set of Web library resources and to maintain backward compatibility with solutions we have developed in the past for museum-oriented information retrieval [7]. On the other hand, since DC is not sufficient to describe complex multimedia content, MUMOC conjugates it with the more recent multimedia-specific MPEG7 standard, similarly to some first research efforts in the area [17, 21].

DC is a widely adopted XML-based standard to maintain library and museum bibliographic metadata. Being DC developed when networked resources were mainly texts or still images, it does not support the representation of multimedia-specific metadata to describe possibly complex, articulated, and composed multimedia contents. For instance, DC does not distinguish still images from moving ones, and uses poor expressive Internet Media Types (MIME) to describe multimedia object formats. MPEG7, instead, provides a rich and complete set of items to describe audiovisual data. For instance, it defines several types of multimedia content (image, video, audio, audiovisual, and multimedia) and, for each type, supports additional descriptions of the exploited data format, such as medium type, file format, and type of coding. An exhaustive description of the MUMOC DC-extended schema is available at `http://lia.deis.unibo.it/Research/MUM/`

The adoption of open and interoperable metadata in MUMOC has a twofold objective: first, it simplifies the dynamic selection of the most suitable one among different versions of the same VoD content with different QoS levels, depending on user/terminal characteristics specified in terms of other standard profile metadata [18]; second, it facilitates the integration with legacy VoD systems and services, thus potentially accelerating the acceptance and the diffusion of the MUMOC middleware proposal.

5. Experimental Results

The section presents experimental results about MUMOC performance while executing a simple VoD lesson streaming service, built on top of the MUMOC caching middleware, in the actual deployment environment of our network lab. We have considered a usage scenario similar to the one presented in Section 3, where many students are willing to access VoD flows of registered lessons. Each participating node hosting the MUMOC middleware may publish VoD contents for downloading simply by registering them at the MUMOC MB. Client nodes only host a simple MUMOC-based application client that can look for the requested flow, command its delivery, and starts playing it as soon as possible during the download.

The used testbed consists of a set of Sun Blade 2000 workstations equipped with a 900 MHz processor and 1024MB RAM and connected by a 100 Mbps Ethernet LAN. The workstations are equipped with the SunOS 5.9 operating system, the Java Virtual Machine (JVM) version 1.4.2_03-b02, and exploit the Java Media Framework (JMF) Performance Pack for Solaris version 2.1.1e as the multimedia streaming library. Heterogeneous clients with more limited hardware/software capabilities are represented by Asus laptops exploiting IEEE 802.11b connectivity and equipped with Windows 2000, the same JVM version, and the JMF Performance Pack for Windows. The experiments presented in the following have mainly intended to measure user-perceived delays in different situations, in order to evaluate the feasibility of the approach and to quantitatively measure the benefits/overhead due to the MUMOC middleware and, in particular, to its exploitation of the JMF multimedia streaming library. All reported results are average values over a set of 100 runs.

In the first experiment, we have disabled all MUMOC caching functions to evaluate the configuration and activation time of our middleware components for metadata querying, VoD flow retrieval, and VoD streaming. The average value for the time interval between the metadata-based VoD flow request and the visualization start at the client has shown to be 824 ms in the case of nodes organized in a 3-level tree. This interval has demonstrated to be mainly composed by: a) video frame activation at the client = 179 ms, b) metadata query = 90 ms, c) RTP session configuration (including server endpoint signaling and RTP client creation) = 341 ms, d) player initialization = 121 ms, and e) rendering initialization = 72 ms. Moreover, with MUMOC disabled, that time interval has shown to linearly grow with the client-server distance (*N* in Table 1), with additional 179 ms for each traversed node in the MUMOC overlay network. The second experiment had the objectives of evaluating the MUMOC content cache activation time and the startup delay increase in the case of activated cache and cache miss. In this case, the MUMOC signaling and the RTP client/server activation has shown to be of 339 ms, while the CacheWriter creation requires 36 ms. In the third experiment, we have activated the distributed caching of MUMOC metadata, by experiencing a startup time increase of about 100 ms per hop in the distance from the requesting client node to the MB component that maintains matching metadata (*M* in the table). When MUMOC MB works, the fixed threshold measured for service path activation in the first experiment decreases to 780 ms, because part of that time was due to metadata browsing. Finally, we considered a fourth case with users exploiting access terminals with limited capabilities. The only specific performance figure of this case is the SBSP activation time, which has demonstrated to be 339 ms, 329 ms of which spent for additional MUMOC signaling and RTP client/server activation due to the added traversed middleware component.

Table 1 sums up the average times registered for user-perceived startup delays in the different situations. Most of the delay is directly related to the low performance of JMF libraries to create and initialize Java-based processor/player objects and to establish RTP sessions. On the contrary, the overhead introduced by the distributed coordination of the MUMOC middleware itself has demonstrated to be limited and acceptable in all the examined cases. The experimental results also confirm the feasibility of the SBSP interposition proposal: its added delay is largely compensated by the possibility to exploit SBatch also when serving access terminals with limited buffering capabilities, thus reducing bandwidth consumption.

Table 1. Average times for user-perceived startup delays

Infrastructure Configuration	Required time (msec)
Service path activation	179 x N + 824
Cache-miss	385
Metadata browser on and service path activation	100 x M + 179 x N + 780
SBSP activation	339
Metadata browser on and VoD lesson cached at one-hop reachable node	100 + 179 + 780 = 1084

Most important, as showed in the last row of Table 1, prefix cache hits significantly reduce the playback delays perceived by end users. In addition, also in the case of cache miss, the overhead introduced by the MUMOC prefix caching middleware is limited. Moreover, the MUMOC guideline of pervasively distributing network/processing workload at different localities in the distribution tree permits to limit the overhead and to achieve good scalability. In summary, first experimental results about the MUMOC performance have shown to be encouraging: the MUMOC active middleware imposes user-perceived delay of less than 3s in most common application scenarios (for usual N and M values), thus ensuring performance results at all compatible with the constraints of soft real-time VoD distribution at the usual Internet transmission rates.

6. Related Work

The distribution of multimedia content over the Internet significantly changes the usual Web caching scenario. Peer-to-peer file sharing and content networks, together with high bandwidth availability for end users, are imposing workloads never experienced before by the Internet infrastructure [4]. The exploitation of distributed caching has already shown its potential in improving service performance and client-experienced response times; several research activities have focused on proposing interesting and effective solutions for caching replacement, positioning, and distributed cache coordination [13, 19].

About the specific aspect of distributing replicas of VoD flow parts on a set of nodes, a few solutions have started to investigate the usage of VoD prefixes [13, 14] and of collaborative caching of VoD segments [20]. In particular, [14] proposes a caching solution at the frame granularity level, by showing the effectiveness of prefix exploitation. Recent work from the same research group has investigated how to achieve optimal prefix positioning in a server-based

batching architecture [13]. The collaborative caching solution presented in [20], instead, exploits a peer-to-peer organization to cache video segments over different peers. The MUMOC approach is similar to [20] from the points of view of cache distribution and application-level overlaying. However, MUMOC aims at disseminating replicas of VoD prefixes and metadata not only to minimize overall network traffic but also to reduce VoD starting delay and to improve scalability via decentralized local access.

Finally, nowadays there is a growing research and industrial interest in enhancing expressiveness and interoperability of multimedia content descriptions [21]. MUMOC exploits the results obtained in these researches by integrating DC and MPEG7 to provide an open extensible format for VoD metadata. To the best of our knowledge, the MUMOC middleware is original in implementing the caching of both VoD prefixes and metadata, and in integrating with an active overlay infrastructure for mobile access to VoD flows adapted on-the-fly.

7. Conclusions and Future Work

MUMOC realizes, in an open and interoperable way, an overlay network for distributed caching, also capable of reducing client-experienced response time via prefix-based techniques. The experimental evaluation of the MUMOC prototype has shown that, notwithstanding the flexible, MA-based, and application-level approach, the introduced overhead is widely compatible with the usual requirements of Internet VoD distribution.

These first encouraging results are stimulating further investigation and the extension of our current MUMOC prototype. In particular, we are working on fully supporting user roaming mobility, i.e., to guarantee that mobile users with Wi-Fi access terminals continue to receive the served VoD flow without noticeable interruptions even during the handoff between IEEE 802.11 cells. The primary idea is to exploit, on the one hand, original lightweight techniques to predict the next cells visited by mobile users and, on the other hand, proactive anticipated buffering in the predicted localities.

Acknowledgements

This work is partially supported by the Italian MIUR within the FIRB WEB-MINDS Project and by the Italian CNR within the Strategic IS-MANET Project.

References

[1] www.gnutella.org

[2] www.kazaa.org

[3] A. Rowstron, P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”. *ACM Conference on Distributed Systems Platforms*, 2001.

[4] S. Saroiu et alii, “An Analysis of Internet Content Delivery Systems”, *USENIX Operating Systems Design and Implementation*, 2002.

[5] H. J. Wang et alii, “ICEBERG: An Internet-core Network Architecture for Integrated Communications”, *IEEE Personal Communications*, Vol. 7, No. 4, 2000.

[6] P. Bellavista, A. Corradi, L. Foschini, “MUM: a Middleware for the Provisioning of Continuous Services to Mobile Users”, *IEEE International Symposium on Computers and Communications*, 2004.

[7] P. Bellavista, A. Corradi, C. Stefanelli, “The Ubiquitous Provisioning of Internet Services to Portable Devices”, *IEEE Pervasive Computing*, Vol. 1, No. 3, 2002.

[8] Y. Bai, M.R. Ito, “QoS Control for Video and Audio Communication in Conventional and Active Networks: Approaches and Comparison”, *IEEE Communication Surveys*, Vol.6, No.1, 2004.

[9] B. Zenel, D. Duchamp, “General Purpose Proxies: Solved and Unsolved Problems”, *IEEE Hot Topics in Operating Systems*, 1997.

[10] P. Bellavista, A. Corradi, C. Stefanelli, “Mobile Agent Middleware for Mobile Computing”, *IEEE Computer*, Vol. 34, No. 3, 2001.

[11] A. Fuggetta, G.P. Picco, G. Vigna, “Understanding Code Mobility”, *IEEE Transactions on Software Engineering*, Vol. 24, No. 5, 1998.

[12] H. Ma, K. G. Shin, “Multicast Video-on-Demand services”, *ACM Computer Communication Review* , Vol. 32, No. 1, 2002.

[13] W. Bing et alii, “Optimal proxy cache allocation for efficient streaming media distribution”, *IEEE INFOCOM*, 2002.

[14] S. Sen, J. Rexford, D. Towsley, “Proxy prefix caching for multimedia streams”, *IEEE INFOCOM*, 1999.

[15] http://archive.dstc.edu.au/mpeg7-ddl/

[16] http://dublincore.org/

[17] http://www.acmi.net.au/dctypeproposal/index.html

[18] http://www.w3.org/Mobile/CCPP

[19] W. Lau, M. Kumar, S. Venkatesh, “A Generalised Cost-aware Caching Scheme for Caching Continuous Media Objects in Best-effort Network Environments”, *IEEE International Workshop on Distributed Computing*, 2002.

[20] W.J. Jeon, K. Nahrstedt, “QoS-aware middleware support for collaborative multimedia streaming and caching service”, *Elsevier Microprocessors and Microsystems*, Vol. 27, No. 2, 2003.

[21] J. Hunter, “Enhancing the Semantic Interoperability of Multimedia through a Core Ontology”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 1, 2003.