Dynamic Adaptation in Distributed Multimedia Applications

Oussama Layaida, Daniel Hagimont

To cite this version:

Oussama Layaida, Daniel Hagimont. Dynamic Adaptation in Distributed Multimedia Applications. [Research Report] RT-0266, INRIA. 2002, pp.19. inria-00069908

HAL Id: inria-00069908
https://hal.inria.fr/inria-00069908

Submitted on 19 May 2006

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Dynamic Adaptation in Distributed Multimedia Applications

Oussama Layaida and Daniel Hagimont

N° 0266

August 2002

THÈME 1

Rapport technique

ISSN 0249-0803

Dynamic Adaptation in Distributed Multimedia Applications

Oussama Layaida and Daniel Hagimont

Thème 1 — Réseaux et systèmes
Projet Sardes

Rapport technique n° 0266 — August 2002 — 19 pages

Abstract: In this report, we present a proxy-based framework for dynamic adaptation in distributed multimedia applications. This proxy can be configured to perform adaptations of multimedia data in real time. Proxy adaptations can be used to manage QoS according to the resources of the underlying environment (capacity of the user terminal or of the network), or to personalize (or extend) the functions of the application. We present the motivations for this approach, the implementation of the framework (based on the Microsoft DirectShow environment) and a performance evaluation that demonstrates the effectiveness of the approach.

Key-words: Multimedia, Content Adaptation, Quality of Service (Qos), Transcoding.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Adaptation dynamique dans les applications multimédia réparties

Résumé : Ce rapport présente une architecture d'adaptation dynamique dans les applications multimédia réparties, basée sur l'utilisation de noeuds intermédiaire dans le réseau. Un proxy peut être configuré pour adapter les données multimédia en temps réel. Ces adaptations peuvent viser la gestion de la qualité de service selon la disponibilité des ressources de lénvironnement sous-jacent (les capacités des terminaux ou celles du réseau) ou pour personnaliser (ou étendre) les fonctions de l'application. Nous présentons les motivations de cette approche, l'implémentation de l'architecture et une évaluation de performance qui démontre la faisabilité de cette approche.

Mots-clés : Multimédia, Adaptation, Qualité de service (Qos),Transcodage.

1 Introduction

Last years have seen the emergence of a multitude of mobile equipments connected to the Internet (telephones, personal assistants, etc.). These equipments are very heterogeneous in different aspects. They can be connected to the Internet through different networks, such as GPRS, Ethernet, modems or infrared cards. They can also have different processing capacities, screen sizes and different abilities to handle specific data encoding (especially multimedia data). Moreover, the Internet Protocol provides a best effort service by routing packets independently and its performance is very variable. As a result, it is very difficult to make hypotheses on the environment in which a distributed application will be executed. Thus, it becomes necessary to adapt a distributed application according to the characteristics of the execution environment. This need of adaptation according to the capacities of the environment is especially important for distributed multimedia applications, which are characterized by an intensive use of host and network resources.

On another side, distributed multimedia applications generally rely on standard software installed on both sides of the network: a multimedia data server (the provider which makes multimedia data available to users) and a multimedia client (the consumer which displays the multimedia data). Application used (Web servers and video players) are basic components which can hardly include the sophisticated algorithms that deal with the constrained capacities of the environment. These algorithms should be integrated in the distributed application independently from these client and server basic components.

We propose a proxy-based framework which allows plugging adaptations on a proxy machine (an intermediate node between the client which receives and the server which distributes the multimedia data) in order to adapt a distributed multimedia application. We are particularly interested in adaptations which aim at transforming the content of a video stream. The main motivation is to take into account the constraints of the environment, but also to extend the functions of the application, without imposing any modification on standard softwares installed on both sides of the network (the client terminal and the multimedia server).

We describe the design of our framework and the application architecture to which it is applied. We motivate this design through several adaptation scenarios. We present an implementation of the proposed framework under the Microsoft DirectShow environment. Our experiments with this framework show that it is possible to perform adaptations of multimedia data dynamically (on the fly), and that it improves resources management when they are limited.

The rest of this report is organized as follows. Section 2 presents our design and describes the adaptation scenarios that we consider. The implementation of the framework is presented in section 3. Section 4 gives the results of our performance evaluation. After a comparison to related works in section 5, we conclude in section 6.

2 Motivations

2.1 Architeture

The architecture of the application that we consider is based on the client-server model. A client plays a video on a terminal, the content of the video being supplied by a server through a communication network. As illustrated in Figure 1, the operator of the communication network generally manages intermediate nodes, called proxies, on which additional treatments (adaptations) can be performed. An important aspect of this architecture is that adaptations can be deployed on proxy sites traversed by the video stream, without opening the client and server sites to these adaptations.

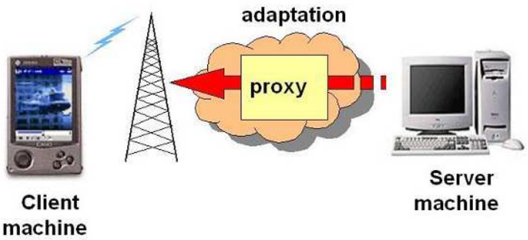

Figure 1. Architecture of the application

The management of proxy sites is a means to take into account two levels of network performance. The first level, which corresponds to the backbone of the network, interconnects servers and proxies with a large bandwidth and a relatively good reliability. The second level interconnects proxies and terminals with a more limited and variable bandwidth (for example a wireless network connection) and a more relative reliability.

Proxies also allow to distinguish the machines on which standard software packages are installed (the client and the server) from machines on which we perform adaptations. In our framework, client machines are provided with a standard video displaying software that we do not want to modify. The servers supply some video contents and we do not want to modify the software packages of the servers or to require that the servers manage the contents of videos in various versions or formats. Proxies allow to perform adaptations which are dynamically deployed and configured, without reconsidering the software packages which are installed on the client and server sites.

2.2 Adaptation scenarios

In this section, we describe several adaptation scenarios that are made possible thanks to this proxy-based architecture. Some of these scenarios have been implemented in order to validate the framework, in particular to make the performance measurements reported in section 4.

2.2.1 Performance improvement

Distributed multimedia applications require efficient data transfers with QoS guarantees. They also require strong CPU capacities to display video data. In a heterogeneous environment, it is not always possible to benefit from the required resources to execute multimedia applications. Adapting these applications under variable resources constraints is a means to maintain an optimal quality level. In our approach, a client communicates exclusively with a proxy, a machine located between the client and the server, which performs adaptations in order to maintain quality.

The expected benefits from these adaptations are, on one hand, to move computation (data transformation) from the client site to the proxy, and on the other hand to reduce the volume of data transferred to the client. In both cases, the principle of the adaptation is to modify on the fly the transferred video. These video modifications can have a significant impact on the client's CPU load, or on the network load. Below, we consider different examples which illustrate this motivation.

A multimedia presentation can be displayed on different types of terminals, which can vary from a powerful workstation to a PDA (Personal Digital Assistant). These terminals differ in terms of processor and display capacities (size and quality). Displaying on a PDA a video encoded for a workstation is often inefficient because the PDA has a much smaller screen. The PDA must resize the video on the fly in order to display it, which requires more CPU than available on the PDA. In this case, a proxy can be configured to perform an adaptation of the size of the video, since the proxy has much more CPU capacities than the PDA. Notice that some low quality encoding formats require much less computation on the terminal. Some terminals may also be optimized for decoding particular video formats. It can then be interesting to convert (on the proxy) the video in the format best supported by the terminal.

The second example illustrates the adaptation to the variation of the network bandwidth. The video can be transferred on networks having different and variable bandwidths. A possible adaptation is to degrade the quality of the video transmitted to the client. For example, the adaptation on the proxy can pass the video in black and white or decrease its quality factor. This adaptation will reduce the volume of data to be transmitted between the proxy and the client.

An important problem with mobile devices is the intrinsic power limitations. Unlike desktop PCs with permanent power supply, mobile computers such as PDAs have limited power autonomy. Together with the LCD display, the CPU is responsible for a large part of the power consumption. Several research works have studied the power consumption on mobile devices running multimedia applications [10][26][14]. Analysis and measurements have demonstrated existing dependencies between consumption and the size, color, depth and encoding format of the video. By adapting the video on the proxy, the power consumption is reduced, which naturally augments the autonomy of the device.

Notice that some adaptations may cumulate several advantages. For instance, consider a multimedia session played on a PDA with a black-and-white screen and transmitted through a low bandwidth network. In this case, it is not necessary to send color information of the video or to include high-resolution pictures since the device is not able to exploit them. Sending these data is not only useless but also worse since it requires additional CPU resources on the PDA to remove this color information and it would waste network bandwidth.

2.2.2 Treatment of temporary failures

Mobile devices often use wireless Hertzian networks which are characterized by a reduced and variable bandwidth, and by more frequent temporary failures (interferences, disconnections). The treatment and the compensation of these failures is a privileged application domain for proxy-based adaptations. Let us consider the case of a user who watches the TV news on its PDA. The TV news program is made available by a TV broadcast company on its Web server.

The temporary failure that we want to take into account here is the failure of the connection between the terminal and the proxy, for example when it is a Hertzian connection. In such a case, the user would like not to have to restart the video from the beginning, but rather to resume the video session from the interrupt point. To supply such a service (without reloading the whole video on the PDA), it would be necessary to have an adapted interface on the server, which would allow resuming the transfer at the point where it was interrupted. This service can be implemented as an adaptation on the proxy machine. The proxy can keep the connection with the server and buffer the video stream the time that the connection with the client is restored, or it can reload the video if the failure lasts longer and transmit it to the client from the interrupt point.

In the case the temporary failure that we want to take into account is a connection failure between the proxy and the server, the proxy adaptation can implement a pre-fetching policy which loads the video on the proxy in advance. This way, the proxy will be able to continue emitting the video towards the client, the time that the proxy reconnects to the server to obtain the rest of the video. These failure handling policies cannot be implanted on mobile equipment. They can be dynamically installed and configured on the proxy sites without requiring modifications of software packages used on client and server machines.

2.2.3 Application's function extension

The third adaptation class that we consider concerns the extension of the service implemented by a distributed multimedia application. Such extensions can aim at integrating an additional function into the presentation of a video, as for example an announcement in subtitle when an event occurs such as the arrival of an E-mail. Mobile devices (such as

PDAs) are generally not multi-threaded and can hardly handle the arrival of an E-mail while the PDA is presenting a video loaded through the network. In our framework, an adaptation can be configured on the proxy site in order to react to an event arrival of an E-mail. The reaction to this event on the proxy consists in inlaying in the video (in subtitle) the name of the E-mail sender as well as the title of the E-mail for example.

Another additional function, less nice, is the inlay of commercials, which can be for example added by the telecommunication operator according to the profile of the user, independently from the server and the presented video.

3 Framework implementation

This section describes the implementation of our framework. We present the architecture and the most important components of the system. The architecture of our proxy is depicted in Figure 2.

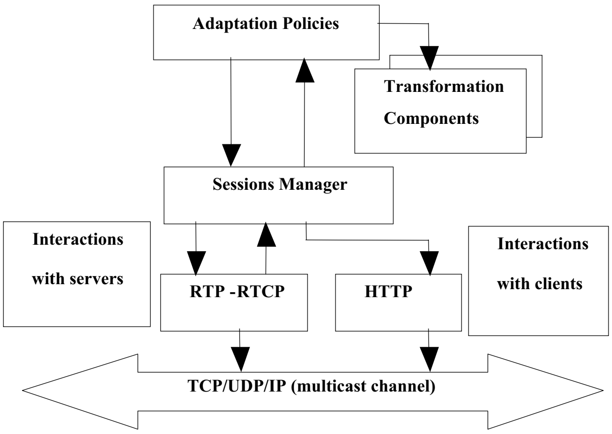

Figure 2. Main Architecture

In our architecture network communications, coders, decoders and adaptations mechanisms are implemented as modular software components. We use the Microsoft DirectShow environment [12] as a basis to implement these components. DirectShow relies on the COM technology and provides an extensible object model for the control and processing of time dependant multimedia streams. The architecture consists of modular components called filters and their assembly in a filter graph. Filters are used to encapsulate basic tasks related to multimedia data processing and they provide interfaces which grant access to internal properties. A filter graph defines links between these filters and allows controlling the flow of data through the graph. In our architecture, a graph implements a session which receives an input stream from a server and generates an adapted stream towards clients. Figure 3 shows an example of a session which consists of a video stream received from a server with the RTP protocol. The video is encoded in MPEG. The session decodes the video, resizes it from CIF to QCIF, encodes it in H.261 and sends the adapted video to a client.

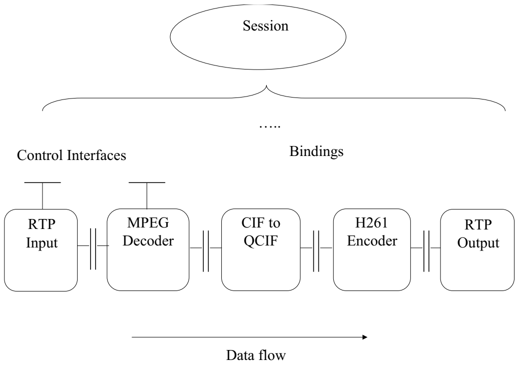

Figure 3. Session example

At the network layer, HTTP and RTP-RTCP are implemented over TCP, UDP and IP-multicast to transport application data. The primary reason to support both protocols is that they are widely used today to transmit video data over the Internet. The session manager maintains all information about active sessions. When it receives a client connection request, it creates all required components and interconnects them to instantiate the session. For real time sessions, it handles clients RTCP reports to monitor remote client information and reception statistics and it enforces the appropriate adaptation policy under variable resource availability via component interfaces. As shown in Figure 4, the input stream is decoded into an intermediate representation, which can then be transformed and delivered to the encoder, which produces an adapted stream in output. Our adaptations can be applied at different levels in the data path. First at the decoder level, by choosing the appropriate decoder component and the intermediate format in which uncompressed data will be represented. Transformations on the content of the video are performed in this intermediate format. Support for multiple intermediate formats allows us to make an optimized configuration to perform these effects (for example, resizing an image in YUY2 format is faster than in RGB format). At the intermediate level, the data can be transformed in various ways. The first type of transformation that we consider is the video size modification (spatial scaling with the nearest neighbor/and bilinear algorithm) required for small devices. This operation is also necessary for some transcoding operation, for example MPEG video may be transmitted in any size, while H.261 requires predefined sizes such as CIF and QCIF. The second type of transformation is the color space-scaling which reduces the number of entries in the color space, for example from 24 to 12 bits, gray-scale or black-and-white. Finally, transformation may aim at integrating a new service in the multimedia application, such as a dynamic insertion of commercial advertisements or subtitles in the video. At the encoder level, the output stream is configured and controlled to offer the best-suited data rate that matches network and receiver's states and capacities. The target data rate is obtained by modifying the rate of encoded frames or by degrading the encoding quality.

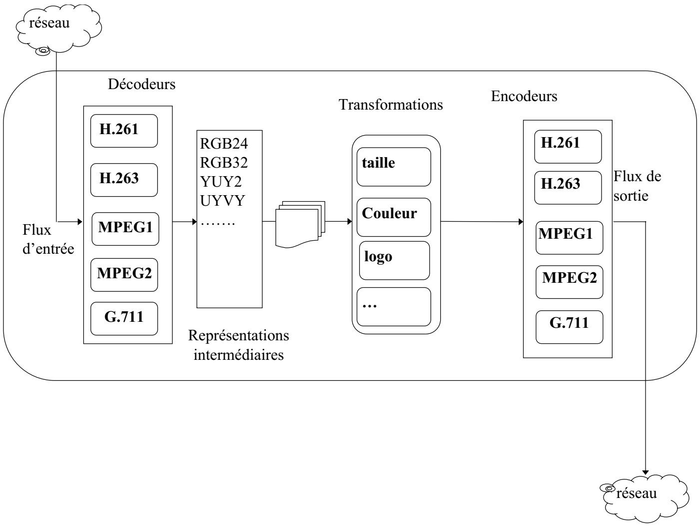

Figure 4. Video transcoding scheme

4 Performance evaluation

In this section, we present a performance evaluation of the framework that we implemented. We evaluate the ability of the framework to perform on the fly content adaptations, and the benefits that we can expect from these adaptations. The adaptations used for this evaluation are mainly related to QoS management because they allow measuring the adaptation benefits.

For these measurements, we used a platform based on two PC workstations interconnected through an Ethernet network and a PDA device (iPAQ) connected to one of the PC through a WaveLan (802.11) network connection. One PC machine hosts the video server application and the other PC hosts the proxy application. The PDA runs a software for presenting the video. Our evaluation demonstrate the benefits of adapting the video on a proxy site, in order to take into account the limited processing and display capabilities of the PDA. Table 1 gives the device types and connection characteristics of all entities in our experiments.

Device	CPU	Memory	Connection
Servers (PC)	700 MHz	256 MB	LAN 100 Mbps
Proxy (PC)	700 MHz	256 MB	LAN 100 Mbps
Clients (PDA IPaq)	200 MHz	32 MB	802.11 2 Mbps

Table 1. Evaluation environment

The applications used on the server and the client are standard software packages which are widely used for multimedia streaming over the Internet. In the different configurations we used in this evaluation, the server runs a Web server to deliver MPEG video or the VIC [15] video conferencing tool. On the client side, we used PocketTV [18] for MPEG1 and VVP [23] for H.261.

4.1 Adaptation needs for clients

Before conducting our experiments with on the fly adaptation on the proxy site, we first evaluated the needs for adaptation. To do so, we measured the real time processing capacity of the mobile device using different video sizes and encoding formats. The measurements concern the displayed frame rate and the energy consumption. Because the battery driver on the PDA (IPaq running Microsoft® Windows CE 3.0) notifies the battery level only when it changes by 10/100, we were not be able to measure the real power consumption rate at a fine grain. To observe power consumption, we displayed a video for a long period (up to one hour) and we measured the average period at which the battery level changes. We used the same video content with different sizes and encoding formats, all encoded at the same frame rate and data rate (25 images/seconds, 250 Kbits/s). Table 2 gives for each case the maximum displayed frame rate and the average power consumption.

Format	Size	Maximum frame rate (fps)	Power consumption
MPEG1	640*480	3.8	every 12 minutes
MPEG1	320*240	9.2	15 minutes
MPEG1	176*144	21.4	19 minutes
H.261	176*144	24.5	24 minutes

Table 2. Performance of PDA with differen formats and sizes

We observe that the PDA can hardly display a large size video because it has to decode and resize the video on the fly. However when the size decreases, the frame rate increases accordingly, in particular for the size QCIF 176x144, which is probably the best suited size for handheld devices such as PDAs. We also observe that power consumption strongly depends on the size of video and encoding format.

4.2 Adaptation benefits

In order to evaluate the adaptation benefits, we performed an experiment which consists in comparing the client performance under two conditions. In the first configuration, the server runs the VIC video conferencing tool which generates a video stream in H.261-CIF (352*288) at 25 frames/seconds and the best quality factor. The video stream is directly transmitted to the PDA through the 2 MB/s wireless link. In the second configuration, the same server transmits the video stream to the proxy PC (through our 100 MB/s Ethernet network) which resizes the video to QCIF (176*144) and re-encodes it at a lower quality (85%). To evaluate the benefits from this adaptation on the proxy machine, we measured the loss rate, data rate and displayed frame rate, all the measurements being performed on the client machine (the PDA).

Results are reported on Figures 5, 6 and 7. They show that the adaptation significantly improves client performance and network resource utilization. Figure 5 shows that when the video is adapted, the client can better process it at a very good frame rate. Without adaptation, the client cannot display the video at the original frame rate, which results in a poor quality presentation. This is confirmed on Figure 6 which reports the measured loss rate. Packet loss can be caused by network congestion or processor overload. Lost packets due to processor overload are packets which are delivered to the client but are discarded as they could not be processed in time (and therefore the displayed frame rate decreases as shown in figure5). These packet drops also result in wasted network resources as these packets are discarded after reaching the client. With the proxy adaptation, network resources are saved (Figure 7) because resizing the video reduces considerably the volume of emitted data. This reduction can also be obtained by quality degradation or by decreasing the frame rate, which would be beneficial when network resources are limited.

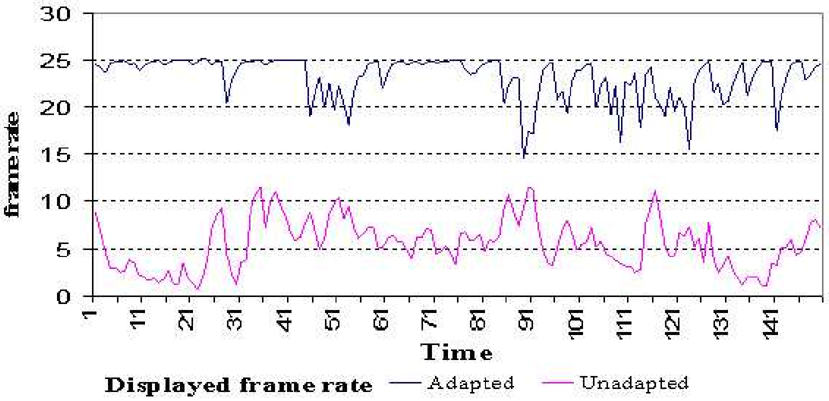

Figure 5. Impact of adaptation on the frame rate

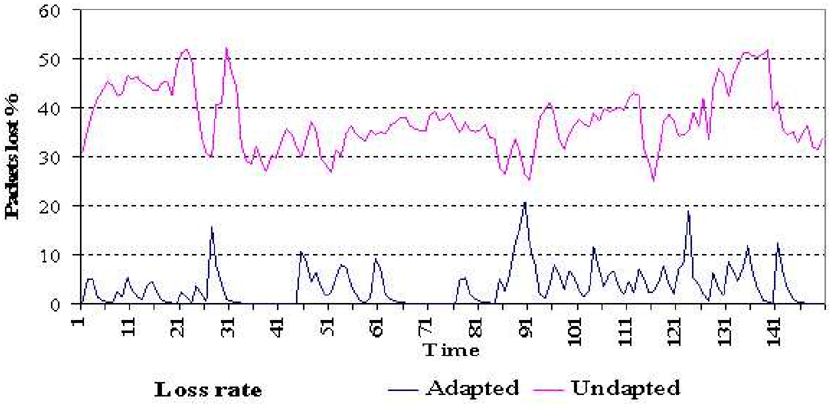

Figure 6. Impact of adaptation on the loss rate

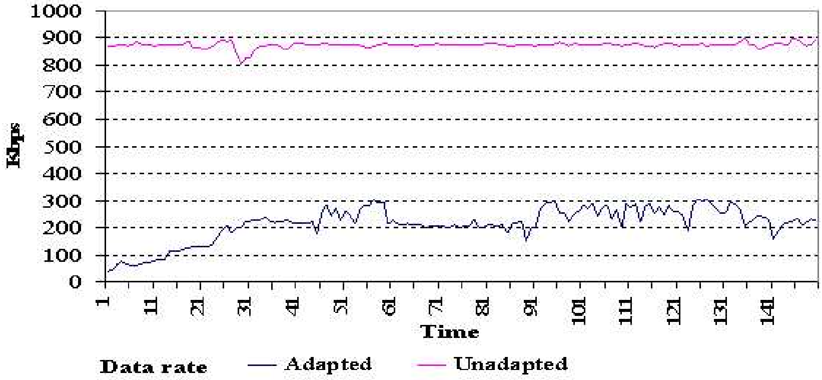

Figure 7. Impact of adaptation on the data rate

4.3 Performance of the proxy

One important hypothesis of our framework is the ability of the proxy to perform video adaptation on the fly. The main considerations are the real time processing capabilities of the proxy machine and its capacity to maintain multiple concurrent sessions. In the previous scenarios, we used a single effect in the proxy to adapt the size and encoding parameters of the video. As previously mentioned, several effects can be added in our proxy framework to implement different adaptations on the same video stream. Then, when using multiple codecs in a session, it is important that the proxy processes data at the same rate at which it receives it from the server, in order to maintain real time communication between the server and the client. To evaluate the influence of these effects on the real time performance of the proxy, we configured a more complex session which receives an MPEG video, resizes it, insert subtitling and changes the color to black and finally, encodes the resulting video in H.261 with the same parameters as the previous experiment. Figure 8 shows the displayed frame rate on client side.

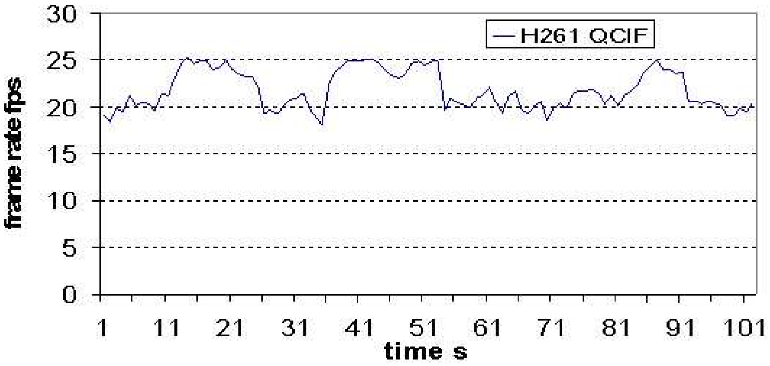

Figure 8. Impact of the addition of effects

We observe on Figure 8 that the addition of codecs in a session does not degrade the quality of the video perceived by the client; the proxy maintains its transmission rate at the same level.

In our architecture, several sessions may be opened on the proxy at the same time. So, another characteristic of the proxy is its ability to support the load of several parallel sessions. To evaluate this, we measured the processor load on the proxy in function of the number of sessions (Figure 9). Each session is composed of an MPEG decoder, resize effect and H.261 encoder. This figure shows that a session consumes about 13 % of the processor CPU resources. We also observe a peak when the proxy creates a new session. This peak corresponds to the creation and the configuration of the codecs which are used in this session (communication, coding, decoding and transformations). These results seem acceptable as we used a medium range PC to execute our proxy software. Using RTP and multicast, the proxy will be able to create separate sessions, each serving a homogenous group of clients with very high quality.

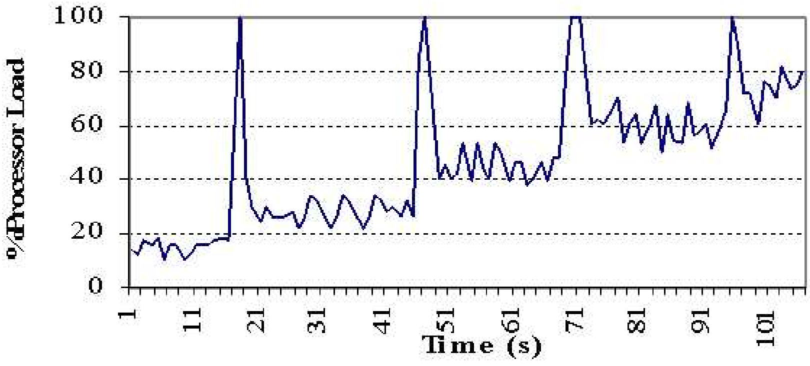

Figure 9. Variations of processor load on the proxy

5 Related work

We compare our work to projects which used adaptation techniques to better manage QoS in distributed multimedia applications.

Numerous projects addressed this issue through network resources reservation [5][17]. These propositions are difficult to integrate in today's heterogeneous networks such as the Internet because they require a globally agreed resource management protocol installed on every intermediary node in the network. In addition, these proposals only address network bandwidth reservation and do not consider adaptation to the capacities of the terminal (handheld PDA or desktop PC).

The second class of projects which considered this issue proposes to adapt at the source the emitted video stream according to the requirements of the clients. A first technique consists in adapting dynamically the output of the source according to the conditions of the receivers, as in [21]. This approach is limited with two respects: it only considers adaptation of the video frame rate and it cannot determine a qualitative level which satisfies all client requirements in a multicast session. A second possibility is to manage several versions of the same video on the server, each of these versions corresponding to different constraints of Qos [19]. The advantage is the simplicity of the approach, but it lacks flexibility because the server can hardly manage all possible versions corresponding to all client requirements and new constraints (for example a new type of terminal) require updates on all the servers. Furthermore, it may lead to network overloading caused by the transmission of multiple replicated streams [7].

Our proposal is to dynamically generate these versions on the proxy. New constraints can be taken into account by dynamically reconfiguring the proxy. Moreover, adaptations can be dynamically produced on proxies close to the clients, without managing per-client end-to-end video streams. More precisely, a single video stream can be sent to a proxy which can implement different client-specific adaptations.

An elegant approach is multi-layered encoding and transmission, which consists in dividing the video into several cumulative layers, each corresponding to an increment of quality [16]. This approach allows to better manage the heterogeneity of the environment since every client can decide on the layers it wishes to receive. However, clients with limited processing capacities will not be able to exploit these encoding formats and they are incompatible with existing multimedia applications. Here, for a multi-layer encoded video, a proxy could be configured to select the layers which are of interest to a client and generate a video stream encoded in the format best suited to the client.

The idea of using intermediate proxies between clients and servers is not new. Several projects have already proposed to adapt video streams on proxy sites [1][20]. These works are essentially based on simulations and were interested in adaptations at the level of the video encoding, which consist either in dropping images or in reducing the quality of the video (at encoding time). Our work aims at adapting the video at any level. In particular, our proxy-based adaptations can change the size of the video or pass it to black and white. Some of the adaptations proposed in this report have been evocated in [22], but we are not aware of any implementation nor performance figures of such proxy-based adaptation framework.

Our framework can also be used to extend the functions of the overall multimedia distributed application. Examples are the integration of commercials in the video or the notification of the arrival of an E-mail.

6 Conclusion

In this article, we presented a framework which allows adapting a distributed multimedia application. This framework relies on intermediate nodes, called proxies, which are traversed by the video streams emitted from server to a client machines. Proxy nodes can be dynamically configured in order to integrate adaptation components. These adaptations can modify the content of the video stream in various ways (e.g. the content of the frames, their size or the encoding format of the video) in order to adapt it according to the conditions of the environment or to extend the functions of the application. We implemented a prototype of this framework in the Microsoft DirectShow environment, validated the framework with the implementation of several adaptation scenarios and presented a performance evaluation. The performance evaluation demonstrates that it is possible to manage adaptation proxies which perform video adaptation on the fly, and that adaptations can significantly improve performance on the client machine when it is limited in terms of processing and display capacities. Our work on adaptability in distributed multimedia applications is going on. We are currently experimenting with more complex adaptation scenarios. A longer term research concerns the dynamic configuration of the proxy machine. We aim at providing a high level language for the description of the proxy configuration.

References

[1] E. Amir, S. McCanne, H. Zhang. *A*n Application Level Video Gateway. Proc. of ACM Multimedia'95, San Francisco, November 1995.

[2] O. Angin, H. Campbell, M Kounavis, R. Liao. *T*he Mobiware toolkit: programmable support for adaptive mobile networking. IEEE Personal Communications Magazine, August 1998.

[3] P. Bahl. Supporting digital video in managed wireless network. IEEE Communications Magazine, June 1998.

[4] Tim Berners-Lee and al. Hypertext Transfer Protocol - HTTP/ 1.1, RFC 2616 ,1997.

[5] G. Bianchi, H. Campbell. Programmable MAC Framework for Utility-based Adaptive Quality of Service Support. IEEE Newspaper on Selected Areas in Communications, February 2000.

[6] Y. Chawathe, S. McCanne, E. Brewer. RMX: Reliable multicast for heterogeneous networks. Proceedings of IEEE InfoCom, 2000.

[7] S. Cheung, M. Ammar, X. Li. On the use of destination set grouping to improve fairness in multicast video distribution. Proceedings IEEE InfoCom'96, March 1996.

[8] DirectX 8.0, Microsoft, Programmer Reference http://www.microsoft.com/directx/

[9] A. Hafid and G. Bochmann. Distributed Multimedia Applications and Quality of service: a review. Electronic Journal on Network and Distributed Processing, n°6, 1998.

[10] C.J Hughes, J. Srinivasan, and S.V. Adve. Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications. Proceedings of the 34th International Symposium on Microarchitecture, December 2001.

[11] IETF. RTP: Transport Protocol for Real-Time Applications. Internet Draft, version 8, revision of RFC-1889, July 2000.

[12] ISO/IEC 11172-2:1993, The MPEG standard, Part 2: Video.

[13] ITU-T Recommendation H.261: Video codec for audiovisual services at p x 64 kbit/s. Geneva, 1990, revised at Helsinki, March 1993.

[14] R. Lee, R. Nathuji. Power and Performance Analysis of PDA Architectures. http://www.cag.lcs.mit.edu/6.893-f2000/project/ , December 2000.

[15] S. McCanne and V. Jacobson. VIC: A flexible framework for packet video. Proc. of ACM Multimedia'95, November 1995.

[16] S. McCanne, V. Jacobson, M. Vetterli. Receiver-Driven Layered Multicast. In SigComm'96, Stanford, CA, August 1996.

[17] M. Margaritidis, G. Polyzos. MobiWeb : adaptive Enabling continuous media applications over wireless links. IEEE International Conference on Third Generation Wireless Communications, San Francisco, June 2000.

[18] MPEG Movie Player for PocketPC, http://www.pockettv.com, 2000.

[19] B. Noble, M. Satyanarayanan, D. Narayanan, J. Tilton, J. Flinn, K, Walker. Agile application-aware adaptation for mobility. Proceedings of the 16th ACM symposium on Operating Systems Principles, France, October 1997.

[20] R. Rejaie, H. Yu, Mark Handley, D. Estrin. Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet. Proceedings of the Conference on Computer Communications (IEEE InfoCom), March 2000.

[21] D. Sisalem, H. Schulzrinne. The loss-delay based adjustment algorithm: A TCPfriendly adaption scheme. Proc. NOSSDAV'98, July 1998.

[22] J. Smith, R. Mohan, C. Li. Transcoding Internet Content for Heterogeneous Client Devices. IEEE Conference on Circuits and Systems, Monterey, June 1998.

[23] M.Thorson. VIC viewer for PocketPC. http://www.oncoursetech.com/video/default.htm, April 2001.

[24] W3C. Working Draft. Composite Capability/Preference Profiles(CC/PP): Structure and Vocabularies, (http://www.w3.org/TR/CCPP-struct-vocab/), March 2001.

[25] X. Wang, H. Schulzerinne. Comparison of Adaptive Internet Multimedia Applications. Institute of Electronics, Information and Communication Engineers (IEICE) Transactions on Communications, Vol. E82-B, June 1999.

[26] W. Yuan, K. Nahrstedt, X. Gu. Coordinating Energy-Aware Adaptation of Multimedia Applications and Hardware Resource. Proceedings of the 9th ACM Multimedia (Multimedia Middleware Workshop), October 2001.

Contents

1 Introduction **3**

2 Motivations **4**
- 2.1 Architeture 4
- 2.2 Adaptation scenarios 4
  - 2.2.1 Performance improvement 5
  - 2.2.2 Treatment of temporary failures 6
  - 2.2.3 Application's function extension 6

3 Framework implementation **7**

4 Performance evaluation **9**
- 4.1 Adaptation needs for clients 10
- 4.2 Adaptation benefits 10
- 4.3 Performance of the proxy 12

5 Related work **14**

6 Conclusion **15**

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803