VideoProxy: A Media and Protocol Converter for Internet Video

Kiyokuni KAWACHIYA Nagatsugu YAMANOUCHI Takayuki KUSHIDA
IBM Research, Tokyo Research Laboratory,
1623-14, Shimotsuruma, Yamato, Kanagawa 242, Japan
<{kawatiya,yamanouc,kushida}@trl.ibm.co.jp>

Abstract. Various types of information are available on the Internet via the WWW, FTP servers, and so on. But most data are downloaded in "batch mode," which is not suitable for large-volume multimedia data such as video because it does not allow instant playback. This paper proposes a new video-transfer scheme, named "VideoProxy," for very low-latency playback of Internet video data. A VideoProxy server is introduced between existing Internet video servers and clients. Both the video data and the transfer protocol are converted by the VideoProxy server.

Keywords. Internet, WWW proxy, adaptive video transfer, media scaling

1. Introduction

Various types of information are available on the Internet via the WWW, FTP servers, and so on. But most data are downloaded in "batch mode," which is not suitable for large-volume multimedia data such as video. For example, to see an MPEG video clip on the WWW, a user must wait a long time for all the data to be transferred. One approach to solving this "high-latency" problem is to start the playback simultaneously with the transfer of the video data. This approach works very well in a dedicated and specialized environment such as video-on-demand. But in the Internet environment, it involves the following difficulties:

- An end-to-end real-time transfer protocol must be provided.
- The information server (WWW, FTP, etc.), as well as the client, must be modified to support real-time data transfer.
- The bit rate of the video data may be higher than the client's connection bandwidth (e.g. 28.8 Kbps).

To solve these problems, this paper proposes a new scheme for video transfer over the Internet, named "VideoProxy." A VideoProxy server exists between an information (video) server and a client, and converts the video data and transfer protocol to allow low-latency playback by the client.

The rest of this paper is organized as follows. Section 2 discusses existing mechanisms for transferring video data over networks, and problems involved in applying these mechanisms to Internet video. Section 3 proposes the VideoProxy approach as a way of solving the problems, and Section 4 describes the prototype implementation. After a discussion of related work in Section 5, Section 6 offers some conclusions and topics for future work.

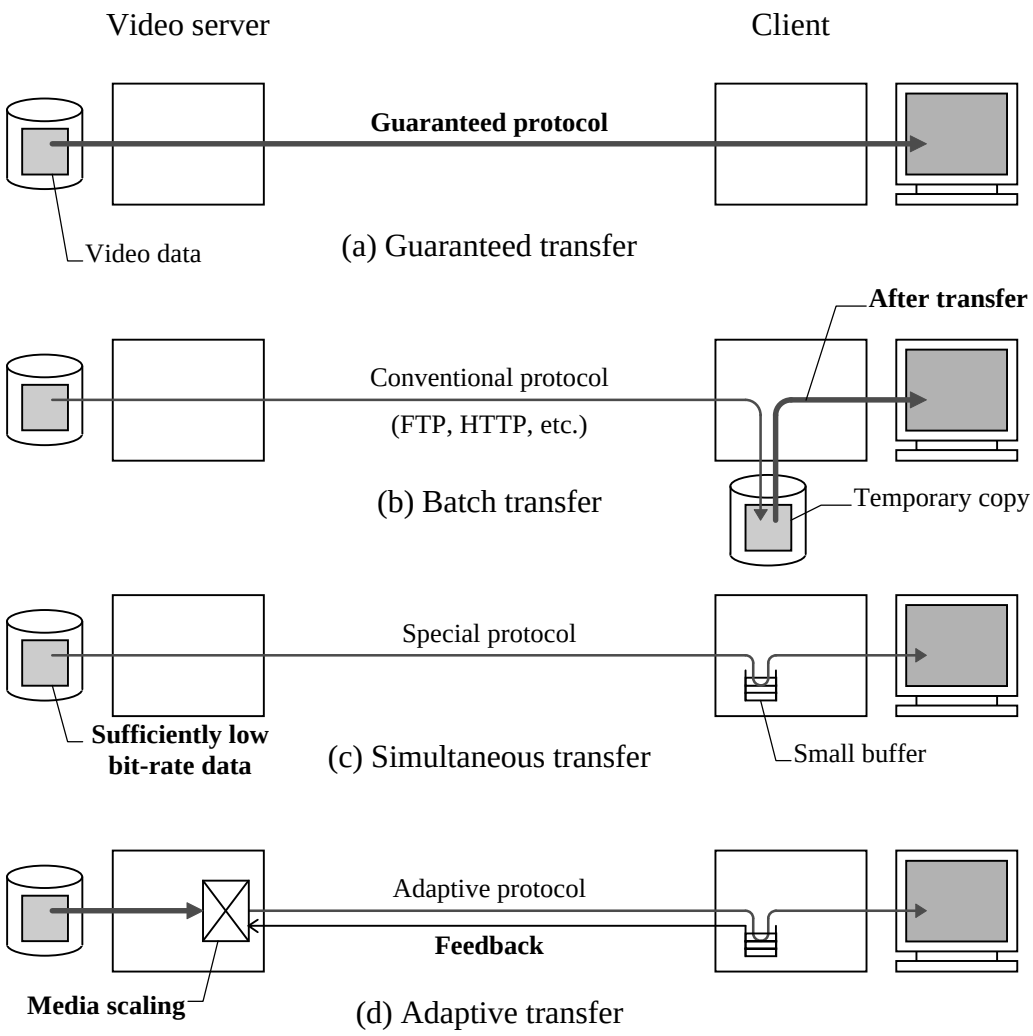


Figure 1: Existing video-transfer mechanisms

2. Video Transfer over Networks

The most notable characteristic of video data, in comparison with conventional data such as text or still image, is the existence of constraints on the timing of the data processing. Every frame of video data must be processed continuously before its deadline. To transfer and play back video data over a network, these timing constraints should be observed. Several frameworks have been considered for transferring video data. Figure 1 shows some existing video-transfer mechanisms.

2.1. Guaranteed Transfer

One trivial (but difficult) way of transferring video data is the "reservation" approach shown in Figure 1(a). This approach uses a *dedicated* network and protocol to ensure that video data are transferred before their deadline. Most video-on-demand systems intended to replace CATV adopt this approach, which seems to be one of the best solutions for transferring video over networks. However, there are several problems in applying it to the Internet environment.

It is difficult to provide an end-to-end real-time data-transfer protocol for the Internet. Several protocols of this type, such as ST-II [1][2] and RSVP [3], have been proposed. However, to guarantee real-time transfer, not only the server and client but also all the intermediate routers must support these protocols. This is not realistic in the current Internet environment. If a firewall gateway is used, it must also be modified to accept these protocols. Moreover, there may be cases in which the network bandwidth is smaller than the bit rate of the video data.

2.2. Batch Transfer

Consequently, "batch-mode" transfer is still widely used in the Internet environment. Figure 1(b) shows the mechanism of batch transfer. First, the whole set of video data is transferred to the client by means of some conventional protocol such as FTP or HTTP. The client then starts playing back the video using the copied video data.

This mechanism can easily realize video playback that meets the timing constraints, because no real-time quality is needed for the network transfer. However, video data are usually much more voluminous than conventional text or still image data, so a user must wait a long time for all the video data to be transferred before the playback can start. In the worst case, the client may not have enough space to store the whole set of video data.

2.3. Simultaneous Transfer

To solve these problems, the playback should start *before* the data transfer is complete. Figure 1(c) shows this "simultaneous-transfer" approach, which can also be used with non-real-time data. But for video data to meet the timing constraints, they should have a sufficiently low bit-rate in relation to the network bandwidth.

Several video systems on the Internet, such as StreamWorks [4] and VDOLive [5], adopt this approach. These systems uses their own low bit-rate encodings and dedicated network protocols. One of our colleagues has also developed a low bit-rate MPEG VOD system for a 128 Kbps ISDN connection [6]. But unlike the guaranteed-transfer approach, it is difficult to completely guarantee that the video data will be transferred before their deadline, even if the bit rate is very low.

2.4. Adaptive Transfer

The fourth approach, shown in Figure 1(d), is a modified version of simultaneous transfer. It uses a dedicated "adaptive" protocol for transfer over the Internet. In this protocol, the network data flow is *monitored* at the client. A small buffer whose size is based on this monitoring is provided at the client to absorb the jitter of the transfer. The monitoring is also *fed back* to the server, and the server controls the data rate of the outgoing video. To control the video data rate, media-scaling (dynamic QOS-control) technology is used — for example, the frame rate is changed by skipping some frames, or the quality of each frame is changed [7][8][9].

Vosaic, developed by the University of Illinois [10], is one example of this approach. We have also designed a video-transfer mechanism based on adaptive control of the MPEG data rate [11].

This approach can work well on the Internet without any modification of existing routers, but still involves the same problems as the first guaranteed-transfer approach. The server must be modified to support the adaptive protocol and media scaling, and the firewall gateway may need to be modified to accept the protocol.

Table 1: Comparison of existing video-transfer mechanisms

	Applicable to the Internet?	Capable of accessing existing servers?	Low-latency playback?	High-quality playback?
(a) Assured transfer	Difficult at this point	Difficult	**Yes**	**Yes**
(b) Batch transfer	**Widely used**	**Yes**	(Very) high latency	**Yes**
(c) Simultaneous transfer	**Becoming major**	Difficult	**Yes**	Depends on the bandwidth
(d) Adaptive transfer	**Yes**	Difficult	**Yes**	Scaled quality

2.5. Requirements for Internet Video Transfer

As shown in previous sections, all existing video-transfer frameworks involve some problems. To summarize, transfer of video data over networks involves the following requirements:

1. Applicability to the Internet
   The transfer mechanism should be applicable to the Internet, where end-to-end reservation is difficult, transfer jitter exists, and the connection bandwidth may be smaller than the video data rate.
2. Capability of accessing existing servers
   The transfer mechanism should be able to access video data in existing servers by using conventional protocols, such as FTP and HTTP. It should also work beyond firewall gateways, through which some restricted protocols can pass.
3. Low latency time for playback
   The waiting time before playback should be short.
4. High-quality video playback
   The video data should be played back smoothly with their timing constraints observed. The quality of the played back video should also be as high as possible.

Table 1 shows a comparison of the existing video-transfer mechanisms based on the four requirements. In this table, phrases in bold letters indicate that the mechanism fully meets the requirement.

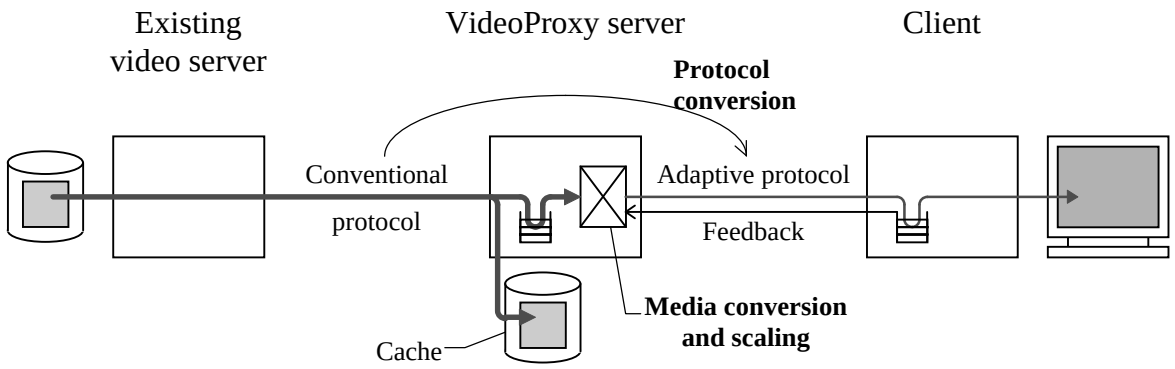
3. VideoProxy — A Media and Protocol Converter

As the summary in Table 1 shows, no existing mechanism satisfies all the four requirements for Internet video transfer. Therefore, we have designed a new video-transfer scheme, named “VideoProxy,” which combines the adaptive-transfer mechanism and the WWW proxy technology.

By using the VideoProxy framework, a user can access video data in existing Internet servers with very low latency.

3.1. Basic Idea of VideoProxy

Figure 2 shows the basic structure of VideoProxy. A “VideoProxy server” is introduced between video servers and clients. It is desirable that the VideoProxy server should be installed at some gateway machine where two networks of different types are connected. A typical location an Internet provider's connection point, where, for example, 28.8 Kbps PPP



(e) VideoProxy transfer

Figure 2: VideoProxy

Table 2: Characteristics of the VideoProxy mechanism

	Applicable to the Internet?	Capable of accessing existing servers?	Low-latency playback?	High-quality playback?
(e) VideoProxy transfer	**Yes**	**Yes**	**Yes**	Scaled quality

line is connected to a high-speed Internet backbone. Another effective location is a firewall gateway, where the server can access both external and internal networks.

The main roles of the VideoProxy server are as follows:

- Protocol conversion — from conventional non real-time protocols to an adaptive real-time protocol
- Media-data conversion — media scaling to lower bit-rate data to match the client's link speed
- Data caching — for further reduction of the latency of future access

From the client, the VideoProxy server can be accessed in the same way as ordinary WWW proxy servers. The VideoProxy server retrieves video data from the specified video server by using some conventional protocol such as FTP or HTTP, and *simultaneously* transfers the video data to the client by using a dedicated adaptive video-transfer protocol. By using a corresponding video-playback program (implemented as an external viewer of a WWW client), the client can display the video with low latency.

In some cases, the bit rate of the original video data is higher than the bandwidth between the VideoProxy server and the client. To achieve low-latency playback in such cases, the VideoProxy server dynamically converts the original video data into a lower bit-rate video data by using media-scaling techniques.

In addition to these protocol and media conversion processes, the VideoProxy server also supports video-data caching in the same way as existing WWW proxy servers. This caching can reduce the traffic between the VideoProxy server and video servers, and enables quicker video playback in the client.

Table 2 summarizes the characteristics of the VideoProxy mechanism on the basis of criteria used in Table 1. VideoProxy satisfies the first three requirements very well. As regards the video quality, it attempts to provide the highest possible quality by using the adaptive video-transfer mechanism. If there is enough connection bandwidth, the highest quality is achieved. The greatest difference between VideoProxy and the adaptive-transfer mechanism is that the former can access existing servers by converting protocols and media data dynamically.

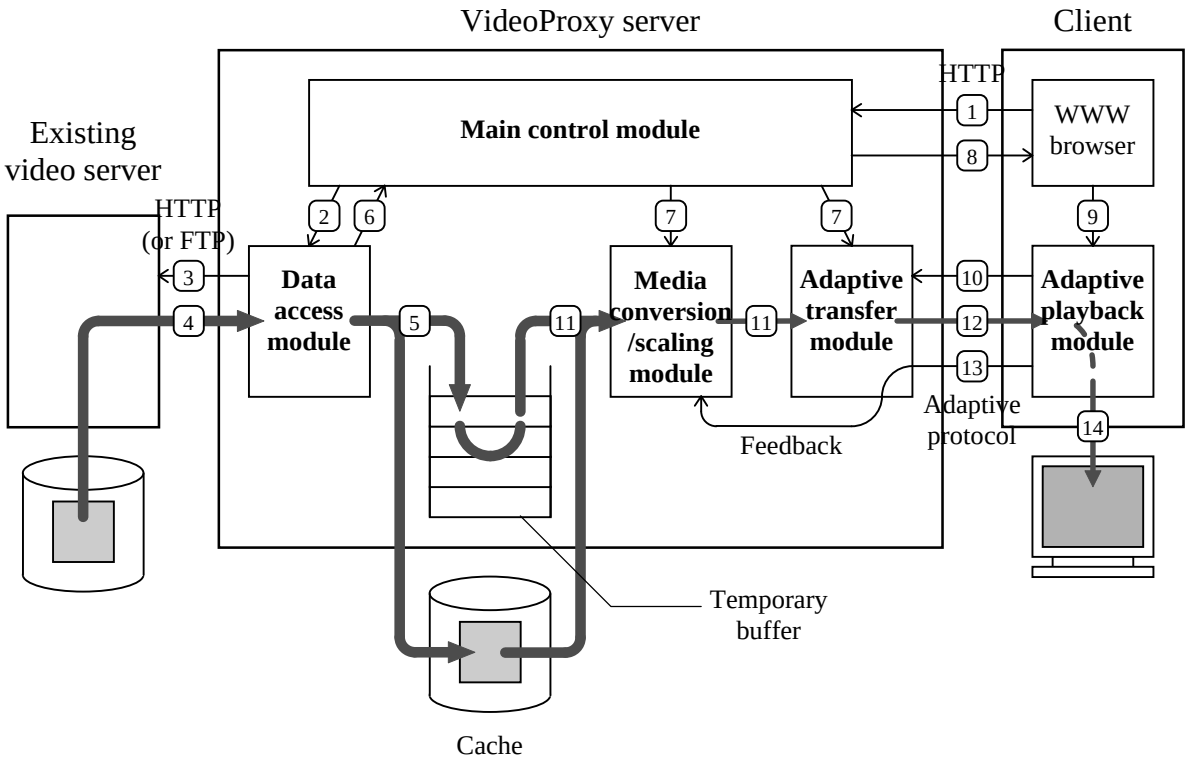


Figure 3: Detailed structure of VideoProxy

3.2. Detailed Design of VideoProxy

Figure 3 shows the detailed structure of the VideoProxy server and client. The VideoProxy mechanism consists of the following five modules: main control module, data access module, media conversion/scaling module, adaptive transfer module, and adaptive playback module in the client.

The remainder of this section describes the role of each module in detail.

3.2.1. Main Control Module

The main control module accepts requests from clients, and controls other modules in the VideoProxy server. It works in the same way as the WWW proxy server: First, it gets a URL from the client, and invokes the data access module to retrieve the data specified by the URL. If the specified data are not video data, they will be passed to the client with no modification, as in the case of normal WWW proxy. The following description focuses on the video data case.

If the specified data are video data that can be handled by the media conversion/scaling module, the main control module returns a special reply[1] to the client to invoke an adaptive playback module. It then invokes the media conversion/scaling module and the adaptive transfer module in order to send the scaled video data adaptively.

The main control module also manages the data caching. If the specified video data are in the cache, they are used instead of invoking the data access module.

[1]A special "Content-type" that indicates the adaptive-transfer mechanism is returned; for example, "`Content-type: video/x-vproxy`."

3.2.2. Data Access Module

The data access module retrieves video data by using some conventional protocol (specified in the URL, such as "`http://...`"), and puts the data into the temporary buffer. While receiving the data, the module also monitors the incoming data flow. On the basis of this monitoring and information on the total data size, the main control module decides when it can start video transfer to the client without causing buffer starvation.

The retrieved data are also stored in the cache for future access.

3.2.3. Media Conversion/Scaling Module

After enough data have been received in the temporary buffer, the main control module invokes the media conversion/scaling module, which is the core of the VideoProxy server. This module converts the original video data in the buffer to match the client's connection bandwidth, which is fed back from the adaptive transfer module.

Several methods can be considered for this conversion. One way is to preserve the media format and change only the bit rate. An example of this approach is to skip some frames of the MPEG data, as in our adaptive transfer system [11]. Another is to change the media format itself to a lower bit-rate format such as H.263 [12] or MPEG-4. Which way should be used depends on the client's connection bandwidth and the processing power of the VideoProxy server machine.

3.2.4. Adaptive Transfer Module

The converted video data are transferred to the client by the adaptive transfer module. A dedicated adaptive transfer protocol, such as one of the protocols used by the adaptive transfer mechanisms shown in Section 2.4, can be used for this transfer.

In the adaptive transfer protocol, the data flow is monitored and the available bandwidth is calculated by the module continuously. This value is fed back to the media conversion/scaling module to enable it to change the bit rate of the converted video data.

3.2.5. Adaptive Playback Module in the Client

The adaptive playback module is invoked in the client as a WWW external viewer that corresponds to the converted "Content-type." The module connects to the adaptive transfer module in the VideoProxy server, and receives the converted video data. Simultaneously, it plays back the video data continuously in accordance with their timing constraints.

The module also monitors the incoming data flow, and feeds it back to the adaptive transfer module in the VideoProxy server. It is also possible to use existing external-viewer (or plug-in) programs for the module by using the corresponding "Content-type."

3.2.6. Example of Video-Data Flow in VideoProxy

The numbers and arrows in Figure 3 shows a typical working scenario of the VideoProxy modules and video-data flow.

1. First, the WWW browser in a client sends a request for video data to the VideoProxy server by using HTTP.
   For example, "`GET http://vserver.xxx.com/video1.mpg`."
2. The main control module receives the request and invokes the data access module.
3. The data access module accesses the video server ("`vserver.xxx.com`" in this case), using a conventional protocol (HTTP in this case).

4. Transfer of the video data ("`video1.mpg`") to the data access module begins.
5. The module puts the data into the temporary buffer (and cache).
6. The data access module monitors the incoming data flow and reports it to the main control module.
7. When the main control module judges that enough data have been received in the buffer, it invokes the media conversion/scaling module and the adaptive transfer module.
8. Simultaneously, the main control module replies to the client by HTTP. A special "Content-type," such as "`Content-type: video/x-vproxy`," and information for connecting to the adaptive transfer module are passed.
9. An adaptive playback module is invoked as an external viewer by the reply.
10. The module connects to the adaptive transfer module.
11. The media conversion/scaling module converts the video data into lower-bit-rate data.
12. The converted data are transferred to the client by the adaptive transfer protocol.
13. The data flow is monitored in the adaptive transfer protocol, and the available bandwidth is fed back to the media conversion/scaling module. The module dynamically adjusts the bit rate of the converted video data to match the bandwidth.
14. The scaled video data is played back by the adaptive playback module.

4. Prototype Implementation of VideoProxy

On the basis of the detailed design described in Section 3.2, we are implementing a prototype VideoProxy server. The first prototype is being developed by combining existing programs.

For network monitoring and media scaling, we have already developed an MPEG-based dynamic QOS-control system over a TCP/IP LAN without resource reservation [11]. This "DQOS" system consists of two components: flow management and media scaling. In the flow-management subsystem, the sent time and received time of each packet are recorded, and available bandwidth is calculated dynamically. The media-scaling subsystem dynamically scales down the MPEG data to meet this bandwidth restriction by skipping B and P pictures accordingly. The scaled MPEG data can be played back on a normal MPEG player.

This DQOS technology can be used for the media-scaling module and adaptive transfer module of the VideoProxy server. The monitoring technology can also be applied to the data access module in order to monitor the incoming flow. For the main control module with caching support, and for the data access module, an existing WWW proxy server program can be used, and we are currently modifying the W3C httpd code [13].

5. Related Work

There have been several studies of proxy technologies, media-scaling technologies, and adaptive-transfer technologies, all of which are related to VideoProxy.

Researchers at Hitachi, Ltd. have proposed a "WWW Quick View System" to reduce the waiting time for browsing large still images [14]. This system also uses the proxy approach, and scales down inlined still images in the proxy server. It also introduces a "Virtual URL" to indicate the scaling information to the server.

The "DeleGate" system developed at Japan's Electrotechnical Laboratory (ETL) is a WWW proxy server that provides several data-conversion services [15]. A typical application is conversion of the character encoding of Japanese text data.

A group at the Japan Advanced Institute of Science and Technology (JAIST) has proposed a “Service Proxy” for mobile computing [16]. The network protocol is converted from the usual TCP/IP into a dedicated wireless protocol for mobile computing. The Service Proxy is also used for video-data scaling of JAIST's own scalable format.

For dynamic QOS control of video transfer, a group at Columbia University has proposed a rate-shaping mechanism for coded video data such as MPEG [17]. In addition, the IBM European Networking Center has proposed a media-scaling architecture with its transport system, HeiTS [8].

A group at Boston University has proposed scalable video delivery using dynamic renegotiation of connections [18][19]. There has also been research on video players that can control the QOS dynamically [20][21].

Several systems have been developed for transferring video over the Internet [4][5][10], as described in Section 2. Most of them use their own video formats and transfer protocols for low-latency playback.

In relation to these studies, we believe that our primary contribution consists in the combination of existing adaptive transfer and media scaling technologies with the Internet-WWW framework. We think some of these technologies, especially some Internet video transfer protocol and media scaling technologies, can be incorporated into the VideoProxy framework.

6. Conclusions and Future Work

Compared with conventional data such as text or still images, video data is very voluminous. Therefore, the usual “batch-mode” data transfer is unsatisfactory for transferring video over networks, because playback cannot start for a very long time. Another characteristic of video data is that there are timing constraints on the processing. Therefore, to play back the video data while they are being transferred, it is necessary to use some special protocol, which may not be applicable to the Internet environment.

To solve these problems, this paper proposed the “VideoProxy” scheme, which combines the adaptive video-transfer mechanism and the WWW proxy technology. A VideoProxy server is introduced between existing video servers and clients, and converts video data dynamically to match the client's connection bandwidth. By using the VideoProxy mechanism, a user can access video data in existing Internet servers with very low latency. Currently, we are developing a prototype VideoProxy based on our DQOS video system.

After confirming the basic efficiency of VideoProxy with the prototype, we plan to develop a more practical VideoProxy server. We are targeting a 28.8 Kbps connection bandwidth for the client, and we plan to conform to some existing mechanism for low bit-rate video transfer over the Internet.

In addition to these prototype implementations, we are currently extending the VideoProxy framework to a more generic “MediaProxy” concept that can manage multiple data types. If this MediaProxy framework is used with appropriate media conversion modules, it will be possible to introduce many new services in the Internet environment — for example, conversion of text data into speech data, or summarization of lengthy video data into a video digest. To facilitate the implementation of such a MediaProxy scenario, we are also designing a new framework named “Proxy Plug-in,” which will make it easy to install a new media conversion module in the MediaProxy server.

Acknowledgments

We would like to thank to Mr. Shuichi Shimizu for his support in modifying the WWW proxy code, and Dr. Kazuya Tago for his encouragement in this work.

References

[1] C. Topolcic et al.: “Experimental Internet Stream Protocol, Version 2 (ST-II),” *Internet RFC-1190* (1990).
[2] L. Delgrossi et al.: “Internet Stream Protocol Version 2 (ST2) Protocol Specification — Version ST2+,” *Internet RFC-1819* (1995).
[3] L. Zhang et al.: “RSVP: A New Resource ReSerVation Protocol,” *IEEE Network, Sep. '93*, pp. 8–18 (1993).
[4] Xing Technology Corp.: “StreamWorks Internet Audio and Video,” URL: <http://www.xingtech.com/streams/>.
[5] VDOnet Corp.: “VDOLive Home Page,” URL: <http://www.vdolive.com/>.
[6] Y. Nakayama: “A VOD System using ISDN,” *Proc. 53rd Annual Convention IPS Japan*, in Japanese (1996).
[7] H. Tokuda et al.: “Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network,” *Proc. ACM SIGCOMM '92*, pp. 88–98 (1992).
[8] L. Delgrossi et al.: “Media Scaling for Audiovisual Communication with the Heidelberg Transport System,” *Proc. ACM Multimedia '93*, pp. 99–104 (1993).
[9] K. Kawachiya and H. Tokuda: “Dynamic QOS Control Based on the QOS-Ticket Model,” *Proc. 3rd IEEE Intl. Conf. on Multimedia Computing and Systems* (1996).
[10] Vosaic Project: “Vosaic: Continuous Media on the Web,” URL: <http://choices.cs.uiuc.edu/Vosaic/Vosaic.html>.
[11] N. Yamanouchi et al.: “An Experimental Dynamic QoS Control System over Internet,” *Proc. INTERWORKING '96* (1996).
[12] ITU-T: “Video Coding for Low Bitrate Communication,” *ITU-T Recommendation H.263* (1995).
[13] World Wide Web Consortium: “World Wide Web Software,” URL: <http://www.w3.org/pub/WWW/Status.html>.
[14] T. Shimada et al.: “User oriented data scaling control for WWW systems,” *Proc. 52nd Annual Convention IPS Japan*, 5F-2, pp. 3-247–3-248, in Japanese (1996).
[15] Y. Sato: “What is the DeleGate?,” URL: <http://www.etl.go.jp:8080/etl/People/ysato@etl.go.jp/DeleGate/>.
[16] A. Hokimoto and T. Nakajima: “Handling Continuous Media in Mobile Computing Environment,” *Proc. 6th Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 175–182 (1996).
[17] A. Eleftheriadis and D. Anastassiou: “Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video,” *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 95–106 (1995).
[18] A. Krishnamurthy and T. D. C. Little: “Connection-Oriented Service Renegotiation for Scalable Video Delivery,” *Proc. 1st Intl. Conf. on Multimedia Computing and Systems*, pp. 502–507 (1994).
[19] H. J. Chen et al.: “A Scalable Video-on-Demand Service for the Provision of VCR-Like Functions,” *Proc. 2nd Intl. Conf. on Multimedia Computing and Systems*, pp. 502–507 (1995).
[20] T. Nakajima and H. Tezuka: “A Continuous Media Application supporting Dynamic QOS Control on Real-Time Mach,” *Proc. ACM Multimedia '94*, pp. 289–297 (1994).
[21] K. Fall et al.: “Workstation Video Playback Performance with Competitive Process Load,” *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 179–182 (1995).