PAPER *Special Issue on Content Delivery Networks*

Proxy Caching Mechanisms with Quality Adjustment for Video Streaming Services

Masahiro SASABE[†], ***Student Member***, **Yoshiaki TANIGUCHI**[††], ***Nonmember***, **Naoki WAKAMIYA**[††], **Masayuki MURATA**[††], ***and*** **Hideo MIYAHARA**[††], ***Members***

SUMMARY The proxy mechanism widely used in WWW systems offers low-delay data delivery by means of "proxy server". By applying proxy mechanisms to video streaming system, we expect that high-quality and low-delay video distribution can be accomplished without introducing extra load on the system. In addition, it is effective to adapt the quality of cached video data appropriately in the proxy if user requests are diverse due to heterogeneity in terms of the available bandwidth, end-system performance, and user's preferences on the perceived video quality. In this paper, we propose proxy caching mechanisms to accomplish high-quality and low-delay video streaming services. In our proposed system, a video stream is divided into blocks for efficient use of cache buffer. A proxy cache server is assumed to be able to adjust the quality of cached or retrieved video blocks to requests through video filters. We evaluate our proposed mechanisms in terms of the required buffer size, the play-out delay and the video quality through simulation experiments. Furthermore, to verify the practicality of our mechanisms, we implement our proposed mechanisms on a real system and conducted experiments. Through evaluations from several performance aspects, it is shown that our proposed mechanisms can provide users with a low-latency and high-quality video streaming service in a heterogeneous environment.

key words: *video streaming service, proxy caching, quality adjustment, MPEG-2*

1. Introduction

With the growth of computing power and the proliferation of the Internet, video streaming services have widely diffused. Then, a considerable amount of video traffic injected by the services causes serious congestion and, as a result, network cannot provide users with the high-quality and interactive services.

The proxy mechanism widely used in WWW systems offers low-delay delivery of data by means of "proxy server". A proxy server caches multimedia data which have passed through it in its local buffer, called "cache buffer", then it provides the cached data to users on demand. By applying proxy mechanisms to a video streaming system, high-quality and low-delay video distribution can be accomplished without introducing extra load on the system [1]–[6]. In addition, it is effective to adapt the quality of cached video data in the proxy if user requests are different and diverse due to heterogeneity in terms of the available bandwidth, end-system performance, and user's preferences on the perceived video quality [1]. Taking into account the heterogeneity among clients is indispensable when we want to provide users with a distributed multimedia service of a satisfactory level of quality.

Manuscript received September 5, 2002.
Manuscript revised January 10, 2003.

[†]The author is with the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University.

[††]The authors are with Department of Information Networking, Graduate School of Information Science and Technology, Osaka University.

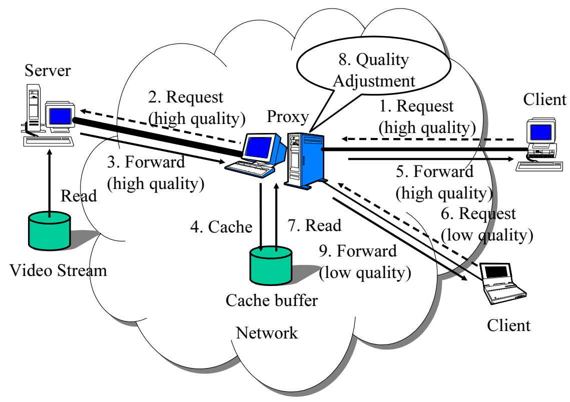

Fig. 1 Video streaming system

Mocha, proposed in [1], employs a layered video coding algorithm to tackle the client-to-client heterogeneity. A proxy retrieves, deposits, and provides video data on a block-by-block and layer-by-layer basis. The system can provide a client with a video stream whose level of quality fits for the client's environment by choosing an appropriate set of layered blocks. However, the number of layers is limited due to a coding algorithm employed and, as a result, the layered-coding based system lacks the scalability and adaptability to rate and quality variations. Even if a satisfactory number of layers can be prepared at a video server, it introduces an extra overhead in video coding and decoding, cache management, and network bandwidth.

In this paper, we propose proxy caching mechanisms to accomplish high-quality and low-delay video streaming services in a heterogeneous environment. In our proposed system illustrated in Fig. 1, a video stream is divided into blocks for efficient use of a cache buffer as in [2]–[5], which propose partial-caching-based proxy mechanisms under a homogeneous environment. A proxy cache server is assumed to be able to adjust the quality of cached or retrieved video blocks to re-

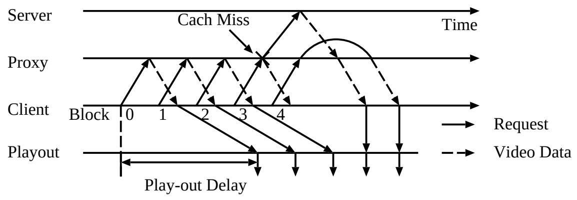

Fig. 2 Basic behavior of our mechanisms

quests through video filters or transcoders. By employing video filters or transcoders, the system can adapt a video stream to fit to dynamically and flexibly changing demands. Video filters and transcoders can directly generate a video stream of a lower level of quality from a stream of a higher level of quality. With the development of image processing technologies, such manipulations can be performed in a real-time fashion [7]. We develop effective algorithms for determining the quality of a block to retrieve from a video server, replacing cached blocks with a newly retrieved block, and prefetching blocks in prior to requests. These algorithms are important when we want to suppress the possibility of cache misses, and decrease the block transfer delay introduced by retrieving missing blocks from a distant video server.

Through simulation experiments, it is shown that our system with the above algorithms can provide users with high-quality and low-delay video streaming services. Furthermore, we implement proposed mechanisms on a real system to verify their practicality and usefulness. We conduct several experiments and evaluate the traffic condition, the video quality variation, and the overhead of quality adjustment. Then, we confirm that our implemented system can continuously provide users with a video stream in accordance with the network condition.

The rest of the paper is organized as follows. In section 2, we propose mechanisms for a proxy cache server with a video-quality adjustment mechanism. Next in section 3, we evaluate our proposed mechanisms through several simulation experiments. In section 4, we describe implementation of proposed mechanisms on a real system, then in section 5, we conduct several experiments and evaluate our proposed mechanisms. Finally, we conclude the paper and describe some future research works in section 6.

2. Proxy Caching Mechanisms with Video-Quality Adjustment

2.1 Overview of Mechanisms

Figure 2 illustrates the basic behavior of our mechanisms. Considering the re-usability of cached data, a video stream is divided into N blocks [1]–[3].

A client periodically requests a designated proxy to send a block. Each request expresses the desired level of quality of the block, which is determined based on the client-system performance, user's preferences on the perceived video quality, and the available bandwidth specified by an underlying protocol, e.g., TFRC (TCP Friendly Rate Control) [8] or the link capacity. We assume that the desired level of quality changes block by block due to changes in user's preferences and the available bandwidth. However, proposed mechanisms can be applicable to the case where user's preferences and the available bandwidth are constant and stable.

The proxy maintains a table, called "cache table", for each of video streams and possesses informations on cached blocks. Each entry includes the block number, the size and quality of the cached block and the quality of blocks under transmission. The size and quality become zero, if the block is not cached or not being transmitted.

On receiving a request, a proxy compares it to a corresponding entry in the table. If the quality of a cached block can satisfy the request, i.e., cache hit, the proxy reads out the cached block, adjusts the level of the quality to the request, and transmits it to the client. Video-quality adjustment is performed by QoS filtering techniques such as frame dropping, low-pass, and re-quantization filters [9]. In some cases, a block being received can satisfy the request. To avoid introducing extra delay in retrieving a block, the proxy waits for the completion of the reception and provides it to the client. Otherwise, when a cache misses the request, the proxy retrieves a block of the appropriately determined quality from the server on a session established to the server for the client. Then, the newly obtained block is stored in the cache. If there is not enough room, one or more cached blocks are replaced with the new block. Cached blocks are useful to reduce the response time, but further reduction can be expected if the proxy retrieves and prepares blocks in advance using the residual bandwidth.

In the following subsections, we propose several algorithms for the block retrieval, prefetching, and replacement mechanisms.

2.2 Block Retrieval Mechanism

When a cache cannot supply a client with a block of the requested quality, a proxy should retrieve the block. The quality of a block that a proxy can retrieve from a server in time is determined in accordance with the available bandwidth between the server and the proxy. Thus, when the path between the server and the proxy is congested, the proxy cannot satisfy the client's demand even if it retrieves the block from the server. We introduce a parameter α which is given as;

$$\alpha = \frac{\max(Q_{sp}(i,j), Q_{cache}(j))}{Q_{pc}(i,j)}. \tag{1}$$

α is the ratio of the quality that the proxy can provide to the request. j $(1 \leq j \leq N)$ is the block number that client i requires. $Q_{sp}(i,j)$ stands for the quality of block j that can be transfered from the server to the proxy within a block time. The block-time is given by dividing the number of frames in a block by the frame rate. For example, in our evaluations, the block corresponds to 30 frames and it is played out at 30 frames per second. Thus, one block-time is equal to one second. By multiplying the available bandwidth by the block-time, the proxy obtains the size feasible for block j. Then, assuming that the quality can be determined from the size [10], $Q_{sp}(i,j)$ is derived. The quality affordable on the path between the proxy and the client is expressed as $Q_{pc}(i,j)$ and is regarded as the client i's request on block j. The quality of a cached block j, $Q_{cache}(j)$, is obtained from a corresponding entry of the cache table. In this subsection, since we consider a case of the cache miss, $Q_{cache}(j) < Q_{pc}(i,j)$ holds.

When $\alpha \geq 1$, that is, the proxy can provide the client with the block of the desired quality, we have three alternatives of determining the quality of block j to retrieve for client i, $Q_{req}(i,j)$.

method1: A possible greedy way is to request the server to send block j of as high quality as possible. This strategy seems reasonable because cached blocks can satisfy the most of the future requests and probability of cache misses becomes small. Then, the request $Q_{req}(i,j)$ becomes

$$Q_{req}(i,j) = Q_{sp}(i,j). \tag{2}$$

method2: When the available bandwidth between the server and the proxy is extremely larger than that between the proxy and the client, method1 cannot accomplish the effective use of bandwidth and cache buffer. Thus, we propose an alternative which determines the quality $Q_{req}(i,j)$ based on prediction of demands on block j, as follows;

$$Q_{req}(i,j) = \min(\max_{k \in S, 0 \leq l \leq j} Q_{pc}(k,l), Q_{sp}(i,j)), \tag{3}$$

where S is a set of clients which are going to require block j in the future. The client i is also in S. Since the affordable level of quality is strictly limited to the capacity of a bottleneck link, we can expect that the maximum quality requested by a client does not change much during a session. The method2 avoids a future cache miss on block j by preparing the block of the highest quality among levels of quality requested by clients on preceding blocks.

method3: To accomplish a further efficient use of the cache, it is possible to request block j of the same quality that the client requests, as follows;

$$Q_{req}(i,j) = Q_{pc}(i,j). \tag{4}$$

With this strategy, the number of cached blocks increases and the probability of cache misses is expected to be suppressed as far as future requests can be satisfied with them.

In some cases, both cached and retrieved blocks cannot meet the demand $(\alpha < 1)$. One way is to request a server to send a block of the desired quality, but it may cause undesirable delay. The other is for a client to be tolerant of the quality degradation and accept a block whose quality is lower than the request. We introduce another parameter β to tackle this problem. β is defined as the ratio of the acceptable quality to the demand, and it expresses the client's insistence on the video quality. Clients with β close to one want to receive blocks in accordance with the request at the risk of undesirable transfer delay. On the other hand, those who value timeliness and interactivity of applications will choose β close to zero.

First, we consider the case that the quality of a cached block can satisfy the client, but is still lower than the request $(\beta \leq \frac{Q_{cache}(j)}{Q_{pc}(i,j)} \leq \alpha < 1)$. In such a case, in order to effectively reuse the cached block, the proxy only sends the cached block to the client regardless of the quality of the block that the server can provide, $Q_{sp}(i,j)$. When the quality of the cached block is not high enough $(\frac{Q_{cache}(j)}{Q_{pc}(i,j)} < \beta \leq \frac{Q_{sp}(i,j)}{Q_{pc}(i,j)} = \alpha < 1)$, the proxy requests the server to send block j whose quality is equal to $Q_{sp}(i,j)$ as;

$$Q_{req}(i,j) = Q_{sp}(i,j). \tag{5}$$

Finally, if the proxy cannot provide the client with a block of the satisfactory quality $(\alpha < \beta)$, it requests the server to send the block of the minimum quality which is expected not to cause a cache miss, that is,

$$Q_{req}(i,j) = \beta \cdot Q_{pc}(i,j). \tag{6}$$

The proxy requests the server to send block j of the quality $Q_{req}(i,j)$. The corresponding entry for client i $Q_{rec}(i,j)$ for the quality of block j being received in the cache table is set to $Q_{req}(i,j)$. When the block reception is finished, $Q_{cache}(j)$ is set to $Q_{rec}(i,j)$.

2.3 Block Prefetching Mechanism

To reduce the possibility of cache misses and avoid the delay in obtaining missing blocks from a server, a proxy prefetches blocks that clients are going to require in the future. After checking the cache table for block j being requested by client i, a proxy compares the minimum requirement $\beta \cdot Q_{pc}(i,j)$ to the quality of cached blocks $Q_{cache}(k)$ and that of receiving blocks $Q_{rec}(i,k)$ (for $\forall i,\ j+1 \leq k \leq j+P \leq N$). Here, P is the size of a sliding window called a prefetching window, which determines the range of examination for prefetching. If there exists any block whose quality is lower than the minimum, a block retrieval mechanism is triggered. The mechanism is the same as one explained in subsection 2.2 except that the available bandwidth

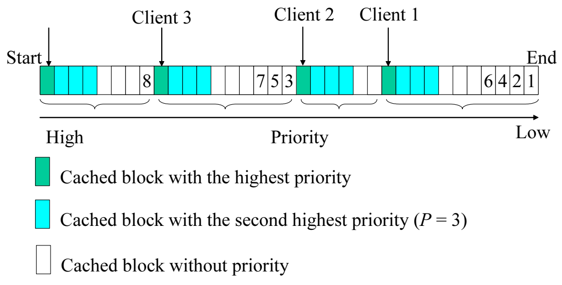
Client 3
Client 2
Client 1
Start
End
8
7 5 3
6 4 2 1
High
Priority
Low
Cached block with the highest priority
Cached block with the second highest priority (P = 3)
Cached block without priority

Fig. 3 Priorities of cached blocks

to prefetching is the remainder of bandwidth between the server and the proxy. Prefetch requests have a lower priority than requests for retrieving cache-missed blocks at the server in order not to disturb urgent block-retrieval.

2.4 Cache Replacement Mechanism

Since a cache has a limited capacity, the replacement of cached blocks should be considered to accomplish the effective use of a storage. When the quality of a newly retrieved block is lower than that of a cached block, the new block is not to be cached. Otherwise, a proxy first removes a cached block of the lower quality if exists. Then, it tries to deposit the new block in the cache. If there is not sufficient room to store the new block, some cached blocks must be removed. We propose a replacement algorithm with consideration of size, quality, and re-usability of blocks.

First, the proxy assigns priority to cached blocks. Blocks requested by clients at the moment of the replacement have the highest priority and are never removed from the cache. The block resides at the beginning of the stream is also assigned the highest priority to provide potential clients with a low-latency service. The second important blocks are those in the prefetching windows following the most important blocks. The other blocks are with no priority.

Then, blocks candidate for replacement are chosen one by one until the sufficient capacity becomes available. In Fig. 3, we show an example of victim selection. A cached block, which locates at the end of longest succession of un-prioritized blocks, is regarded as the least important and becomes the first victim as indicated as “1” in the figure. Among successions of the same length, one closer to the end of the stream has a lower priority.

The proxy first tries the quality adjustment to decrease the size of the victim if it is larger than the new block. Since it is meaningless to hold a block whose quality is smaller than $Q_{req}(i,j)$ determined by the method chosen in the block retrieval mechanism, no further adjustment is performed and the victim is removed from the cache. The proxy repeatedly chooses the next victim and applies the same techniques until the capacity for the new block becomes sufficient. When all un-prioritized blocks are removed but it is still insufficient, the proxy gives up storing the new block.

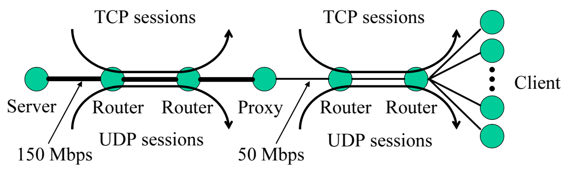

Fig. 4 Simulation system model

3. Simulation Experiments

In this section, we conduct simulation experiments to evaluate performance of the proposed caching mechanisms in terms of the required buffer size, the play-out delay, and the video quality.

Figure 4 illustrates our simulation system model. A video stream of two hours long is coded using the MPEG-2 video coding algorithm. It is segmented into GoPs (Group of Pictures) and a GoP corresponds to a one-second block. The video-quality adjustment is performed by a re-quantization filter which regulates the quantizer scale, that is, the degree of the quantization. The size of entire video stream ranges from 8.6 Gbits to 194.5 Gbits according to the applied quantizer scale. Ten clients are connected to the proxy on the same path and watch the same video stream from the beginning to the end without interactions such as rewinding, pausing, and fast-forwarding. The inter-arrival time between two successive client participations follows the exponential distribution whose average is 1,800 seconds. The propagation delay between the server and the proxy is 200 msec and that between the proxy and the client is 50 msec. The simulation runs for 29,000 seconds in simulation time unit. It is assumed that all sessions employ TFRC as an underlying rate-control protocol. TFRC is a rate control algorithm for non-TCP based applications to react against network congestions and regulate data sending rate in a TCP-friendly fashion. A sender estimates the TCP-friendly rate based on feedback information on RTT and packet loss ratio observed at a receiver. By adjusting the sending rate to the estimated TCP-friendly rate, network bandwidth is expected to be fairly shared among TCP and non-TCP sessions. For further details of a mechanism, please refer to the implementation described in subsection 4.2. We conduct simulation experiments on TFRC with ns-2 [11] and obtained results are used as the available bandwidth of sessions. An enlarged view of the available bandwidth between the server and the proxy is shown in Fig. 5 and that between the proxy and the client is shown in Fig. 6.

Although requests are sent to the proxy at the regular interval of a block-time, inter arrival times of blocks at the client fluctuate due to cache-hit, cache-

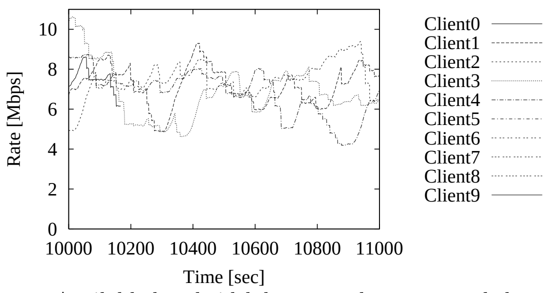

Fig. 5 Available bandwidth between the server and the proxy

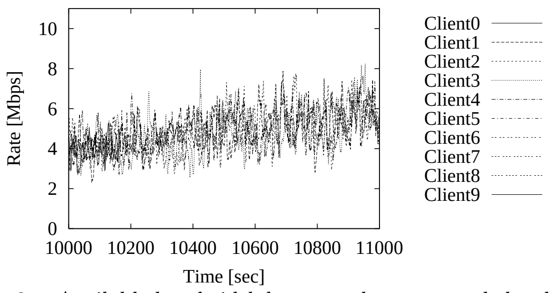

Fig. 6 Available bandwidth between the proxy and the client

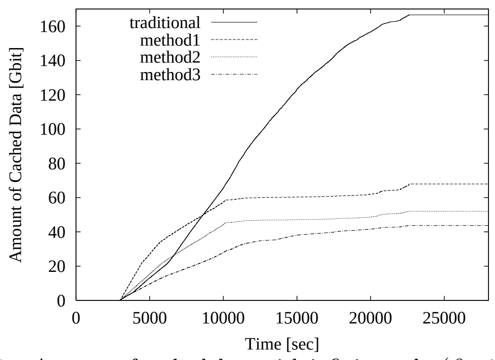

Fig. 7 Amount of cached data with infinite cache ($\beta=1$, $P=0$)

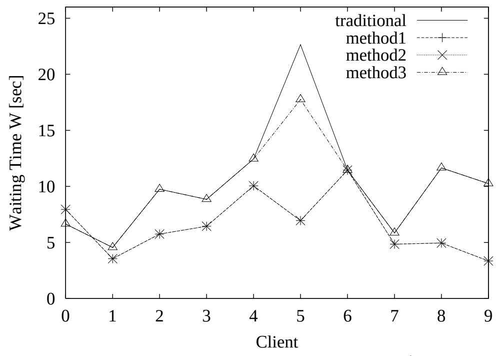

Fig. 8 Play-out delay with infinite cache ($\beta=1$, $P=0$)

miss, and available bandwidth. In any types of streaming services, it is necessary for a client to defer the play-out and buffer some amount of video preparing for expected or unexpected delay jitter (See Fig. 2). We define the time that client i waits and buffers video blocks at the beginning of the session in order to ensure regularity and smoothness of video play-out as the play-out delay $W(i)$. $W(i)$ is derived as;

$$W(i) = \max_{1 \le j \le N} (T(i,j) - I(i,j)), \tag{7}$$

where j stands for the block number, and N is the number of blocks in the stream. The arrival time of block j at client i is denoted as $T(i,j)$. $I(i,j)$ corresponds to the ideal arrival time of block j and those conditions hold that $I(i,j) - I(i,j-1) =$ one block-time and $I(i,1) = T(i,1)$.

Next, we define the degree of user's satisfaction with video quality as;

$$S(i) = \frac{1}{N} \sum_{j=1}^{N} \frac{Q_{act}(i,j)}{Q_{pc}(i,j)}, \tag{8}$$

where $Q_{act}(i,j)$ is the quality of block j provided to client i whose request on the block is $Q_{pc}(i,j)$.

In Figs. 7 through 9, we summarize simulation results on the amount of cached blocks and the play-out delay. The proxy is assumed to have an infinite cache buffer. Prefetching window size is set to zero, i.e., no prefetching, in Figs. 7 and 8, and 30 in Fig. 9. Client's insistence on the quality β is set to 1. Those results labeled "traditional" correspond to the case that the proxy does not have capability of neither quality adjustment or prefetching. In such a case, a cache hits a request only when it has a video block of the same quality as the request. Otherwise, the proxy retrieves the block of the requested quality from the server. The traditional proxy tries to store every blocks it retrieves even if it already has a block of the same number and the higher quality. Figure 7 shows that the required buffer size of proposed methods is down to one forth of the traditional method while providing clients with video blocks of the requested quality. In addition, even if clients insist on the quality, the play-out delay is suppressed by introducing the quality adjustment and the prefetching mechanism as shown in Figs. 8 and 9. In the case of method3, the prefetching mechanism is not so effective in comparison with the others because the proxy retrieves and caches blocks of the minimum quality.

Next, we show simulation results for the case where the proxy is equipped with the cache of 20 Gbits, which is smaller than the half of that required (see Fig. 7). Since we cannot expect an efficient use of cached blocks with obstinate clients, we assume that they are tolerant of quality degradation, that is, $\beta = 0.6$.

The lower β leads to the higher cache-hit probability. Consequently, regardless of methods, the play-out delay becomes small enough while the amount of cached blocks is limited to 20 Gbits as shown in Fig. 10. Furthermore, since method3 requests the server to send blocks of the lowest quality among three methods, the block transfer time in method3 becomes small and the

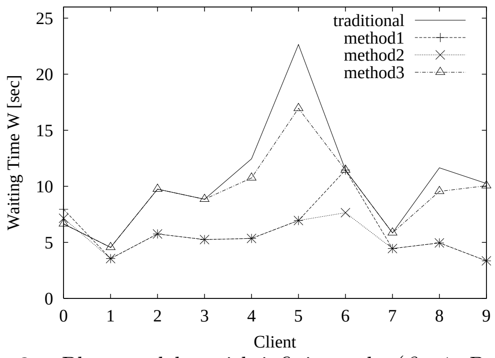

Fig. 9 Play-out delay with infinite cache ($\beta = 1$, $P = 30$)

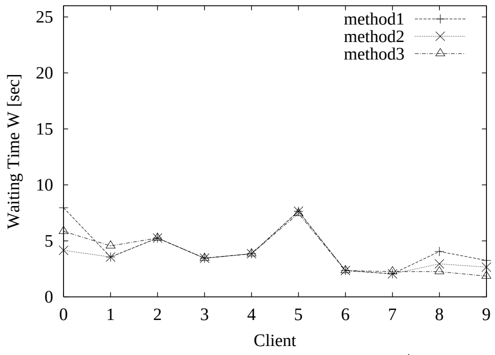

Fig. 10 Play-out delay with 20 Gbits cache ($\beta = 0.6$, $P = 30$)

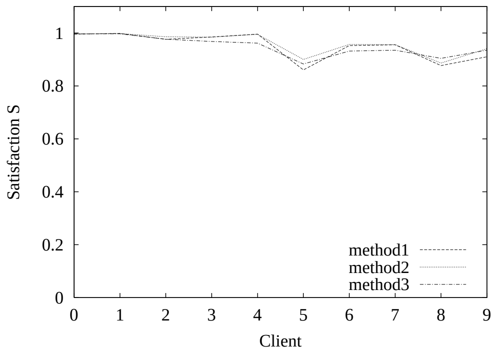

Fig. 11 Degree of satisfaction on delivered video with 20 Gbits cache ($\beta = 0.6$, $P = 30$)

number of cached blocks increases. As a result, performance improvement of method3 becomes high. Due to a limited cache buffer, the degree of satisfaction $S(i)$ slightly decreases but is still higher than 0.6 as shown in Fig. 11.

4. Implementation of Proposed Mechanisms

In this section, we describe our implementation of proposed mechanisms on a real system. Figure 12 illustrates modules constituting our video streaming system. The implemented system consists of a video server, a proxy cache server, and several clients. We employ well-known and widely-used protocols for inter-system communications. For example, the video streaming is controlled through RTSP (Real Time

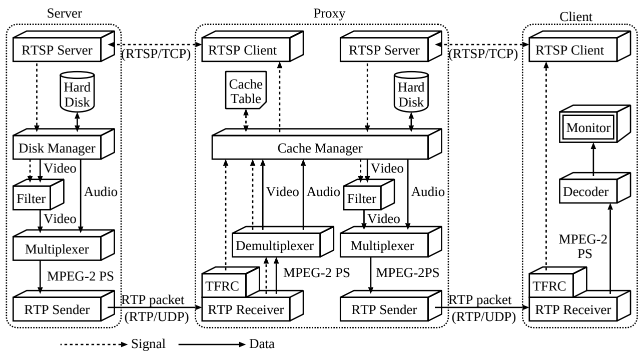

Fig. 12 Modules constituting system

Streaming Protocol) / TCP sessions. Video blocks are transferred over RTP (Realtime Transport Protocol) / UDP sessions as being segmented into 1 K bytes-long RTP packets. The video stream is coded using the MPEG-2 video coding algorithm in the PS (Program Stream) format. The available bandwidth, that is taken into account in three mechanisms explained in section 2, is determined by TFRC. The video-quality adjustment is performed by a low-pass filter [9]. In the following subsections, details of our implementation are given.

4.1 Demultiplexing MPEG-2 PS Blocks

MPEG-2 PS is one of formats for multiplexing video and audio streams. As the quality adjustment is applied only to video data, a block received through a proxy's *RTP Receiver* is divided into a pair of video and audio blocks by *Demultiplexer*. The divided blocks are stored in a cache separately. In our implemented system, each block corresponds to a GoP (Group of Pictures) of MPEG-2, which consists of a series of frames.

The block to request and its quality are specified in the header of an RTSP PLAY message using the Range field and Bandwidth field, respectively. In a case of a cache hit, a proxy reads out both video and audio blocks from its cache, and it applies the quality adjustment only to the video block. Then, those blocks are multiplexed and transmitted to the requesting client.

4.2 Rate Control with TFRC

TFRC is a protocol that enables a non-TCP session to behave in a TCP-friendly fashion. TFRC sender estimates the throughput of a TCP session sharing the same path using the equation (9).

$$X = \frac{s}{R\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \quad (9)$$

X is the transmit rate in bytes/second. s is the packet size in bytes. R is the round trip time in seconds. p is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted. t_{RTO} is the TCP retransmission timeout value in seconds. b is the number of packets acknowledged by a single TCP acknowledgement. Those informations are obtained by exchanging RTCP (Real-Time Control Protocol) messages between a sender and a receiver. In our implemented system, we can use RTSP as a feedback mechanism where a client calculates the TCP-friendly rate and informs a proxy of the rate using the Bandwidth field of a RTSP PLAY message.

4.3 Video-Quality Adjustment

We employ the low-pass filter as a quality adjustment mechanism. We compared several video filters such as the low-pass, re-quantization, and frame dropping [7]. Through experiments, it was shown that the low-pass filter is the most suitable as an MPEG-2 video filter for its flexibility in rate adaptation, faster processing, and video quality. The low-pass filter adjusts the video quality to the desired level by discarding some portion of less influential information in video blocks.

4.4 Cache Manager

A proxy maintains information about cached blocks as *Cache Table*. On receiving a request, *Cache Manager* examines the table. When a cache miss occurs, it determines the quality of a block to retrieve from a video server in accordance with the available bandwidth and requests using methods explained in section 2. Then, it requests the server to send the block via an RTSP session established between them.

The server reads out a pair of a video block of the highest quality and a corresponding audio block through *Disk Manager*, adjusts the quality of the video block to the request using *Filter*, rebuilds a PS block by *Multiplexer*, and finally sends the block to the proxy via an RTP session in a TCP-friendly fashion.

Cache Manager obtains a pair of blocks through *RTP Receiver* and *Demultiplexer*. The block is sent to the client in a similar way to the block transfer from the video server to the proxy. At the same time, a pair of blocks is stored in *Hard Disk*. *TFRC* calculates TCP-friendly rate while receiving RTP packets from the server.

Cache Manager is also responsible for prefetching blocks and replacing cached blocks with new blocks.

5. Experimental Evaluation

In this section, we conduct experiments to evaluate the rate variation, the video quality variation, and the overhead of quality adjustment.

Figure 13 illustrates a configuration of our experimental system. Two clients are connected to the proxy and watch the same video stream of 10 minutes from the beginning to the end without interactions such as

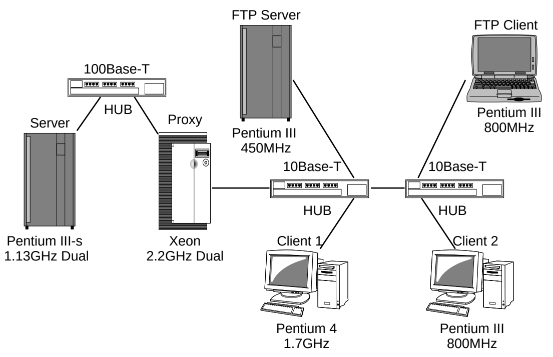

Fig. 13 Configuration of experimental system

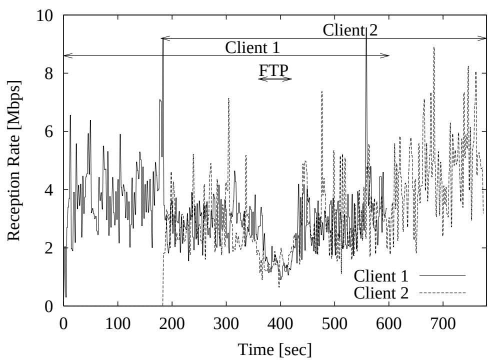

Fig. 14 Reception rate variation

rewinding, pausing, and fast-forwarding. The video server has the whole video blocks of the highest quality of 8 Mbps. The proxy also has the whole video blocks, but the quality is 3 Mbps and a cache buffer capacity is limited to 450 MBytes. The proxy determines the quality of a block to retrieve from a video server using method3 which does not need to adjust the quality of a newly retrieved block at the proxy. The prefetching window size P is set to 5. There exists a TCP session for the file-transfer as a disturbance that competes with video sessions for the bandwidth. The client's insistence on the video quality, β, is set to 1. Clients defer playing the video until video data of 4 MBytes are stored in play-out buffer.

For purpose of comparison, we also conducted experimental evaluation of the traditional method where users persistently request blocks of the quality of 6 Mbps. The proxy has the whole video blocks of the quality of 6 Mbps in the cache buffer. Thus, in the case of the traditional method, there is no cache miss.

Figure 14 illustrates variations in reception rates observed at the clients' *RTP Receiver*. At time 180, the client2 begins a video session. From 360 to 420, the TCP session transfers a file. As shown in the figure, the video rate is regulated as the network condition changes, to avoid unexpected transfer delay and quality degradation that would be caused by congestions.

Figure 15 illustrates variations in the perceived video quality in terms of the coding artifact measured by VP2000A of KDD Media Will Corporation. A

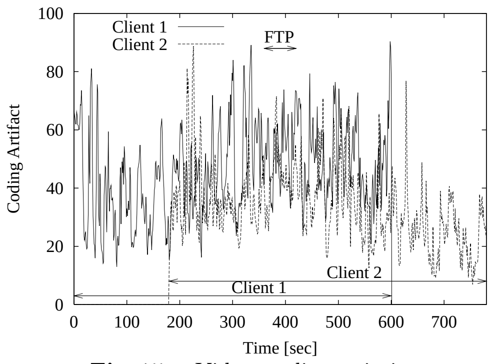

Fig. 15 Video quality variation

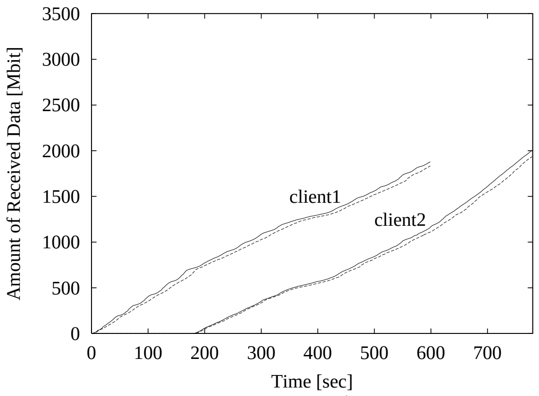

Fig. 16 Amount of received data (proposed mechanisms)

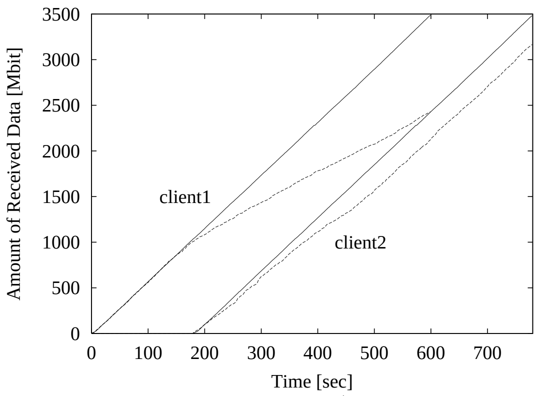

Fig. 17 Amount of received data (traditional mechanism)

higher coding artifact value means that quality degradation is higher. These figures show that the perceived video quality changes in accordance with the network condition.

Through experiments, the maximum processing time of adjusting the quality of a video block was less than 0.5 second. Since the proxy is required to process each block faster than one second, i.e., the interval between two consecutive requests in our experiments, more than 0.5 second can be devoted to the other tasks including retrieving a block from the video server. Figures 16 and 17 illustrate trajectories of the cumulative amount of received data for the proposed system and the traditional system, respectively. Solid lines stand for the ideal and expected amount of data to be received, which gradually increases by the size of a frame at the time when the frame is to be played-out. Dashed lines stand for the actual and observed amount of data received at clients. Gaps between the ideal line and the actual line in our system are apparently smaller than those in the traditional system. In the case of our system, since we carefully regulate the quality of a video block in order not to violate the deadline, a major reason of the gap is packet loss. On the other hand, the gaps are caused by the delay and packet loss in the case of the traditional system, because clients persistently request blocks of 6 Mbps quality regardless of the available bandwidth. As a result, we observed only several freezes during 10-minutes video play-out in the case of our system and we could not have continuous and in-time video play-out in the case of the traditional system. Thus, we can conclude that our mechanisms can accomplish a low-latency and high-quality video streaming service under a heterogeneous and dynamically changing environment.

6. Conclusions and Future Work

In this paper, we proposed several caching mechanisms for the video streaming system with the proxy server capable of video-quality adjustment. Simulation results show that our system is effective enough in suppressing the play-out delay and reducing the required cache size while providing users with a video stream of the desired quality. Especially for the limited buffer, it is shown effective for clients to be tolerant in order to accomplish the low-delay and efficient video streaming service. Furthermore, we implemented and evaluated our proposed mechanisms on a real system. Through the experiments, it is shown that our implemented system can continuously provide users with a high-quality and low-delay video service in accordance with the network condition.

As future research works, we consider improvement of our proposed mechanisms. For example, we tackle the case that several video streams are requested by clients in a larger network environment. We also propose mechanisms for interactive controls and algorithms for further stable video quality.

Acknowledgements

This work was partly supported by Special Coordination Funds for promoting Science and Technology of the Ministry of Education, Culture, Sports, Science and Technology of Japan, and Telecommunication Advancement Organization of Japan.

References

[1] R. Rejaie and J. Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for internet streaming," in *Proceed-*

ings of NOSSDAV, (New York), June 2001.

[2] M. Hofmann, T. S. E. Ng, K. Guo, S. Paul, and H. Zhang, "Caching techniques for streaming multimedia over the internet," *Technical Report BL011345-990409-04TM*, April 1999.

[3] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in *Proceedings of the 10th International WWW Conference*, pp. 36–44, 2001.

[4] J. Shudong, B. Azer, and I. Arun, "Accelerating internet streaming media delivery using network-aware partial caching," *Technical Report BUCS-TR-2001-023*, October 2001.

[5] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," in *Proceedings of IEEE INFOCOM 2002*, (New York), June 2002.

[6] M. Reisslein, F. Hartanto, and K. W. Ross, "Interactive video streaming with proxy servers," *Information Sciences: An International Journal*, December 2001.

[7] T. Yamada, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation and evaluation of video-quality adjustment for heterogeneous video multicast," in *Proceedings of APCC 2002*, (Bandung), pp. 454–457, Sept. 2002.

[8] M. Handley, J. Padhye, S. Floyd, and J. Widmer, "TCP, friendly rate control (TFRC),: Protocol specification," *Internet draft draft-ietf-tsvwg-tfrc-04.txt*, April 2002.

[9] N. Yeadon, F. Garcí,a, D. Hutchinson, and D. Shepherd, "Filters: Qos support mechanisms for multipeer communications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1245–1262, September 1996.

[10] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara, "QoS, mapping between user's preference and bandwidth control for video transport," in *Proceedings of IFIP, IWQoS '97*, pp. 291–302, May 1997.

[11] UCB/LBNL/VINT, "Network simulator - ns (version 2)." available at `http://www-mash.cs.berkeley.edu/ns/`,.

Masahiro Sasabe Masahiro Sasabe received the B.E. degree in Informatics and Mathematical Science from Osaka University, Osaka, Japan, in 2001. He is currently the master student at the Graduate School of Engineering Science, Osaka University, Japan. His research research interests include QoS architecture for real-time and interactive video distribution system. He is a student member of IEICE.

Yoshiaki Taniguchi Yoshiaki Taniguchi received the B.E. degree in Informatics and Mathematical Science from Osaka University, Osaka, Japan, in 2002. He is currently the master student at the Graduate School of Information Science and Technology, Osaka University, Japan. His research work is in the area of QoS architecture for real-time and interactive video distribution system.

Naoki Wakamiya Naoki Wakamiya received M.E. and Ph.D. from Osaka University in 1994 and 1996, respectively. He was a Research Associate of Graduate School of Engineering Science, Osaka University from April 1996 to March 1997, a Research Associate of Educational Center for Information Processing, Osaka University from April 1997 to March 1999, an Assistant Professor of Graduate School of Engineering Science from April 1999 to March 2002. Since April 2002, He has been an Associate Professor of Graduate School of Information Science and Technology, Osaka University. His research interests include QoS architecture for distributed multimedia communications. He is a member of IEICE, IPSJ, ACM, and IEEE.

Masayuki Murata Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April, 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From 14 September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1992 to 1999, he was an Associate Professor in the Graduate School of Engineering Science, Osaka University, and from April 1999, he has been a Professor of Osaka University. He moved to Advanced Networked Environment Division, Cybermedia Center, Osaka University in April 2000. He has more than two hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modeling and evaluation. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.

Hideo Miyahara He received the M.E. and D.E. degrees from Osaka University, Osaka, Japan in 1969 and 1973, respectively. From 1973 to 1980, he was an Assistant Professor in the Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan. From 1980 to 1986, he was an Associate Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University, Osaka, Japan. From 1986 to 1989, he was a Professor of the Computation Center, Osaka University. Since 1989, he has been a Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1995 to 1998, he was a director of Computation Center of Osaka University. From 1998 to 2000, he was a dean of the Faculty of Engineering Science, Osaka University. From 2002, he is a Dean of Graduate School of Information Science and Technology, Osaka University. From 1983 to 1984, he was a Visiting Scientist at IBM Thomas J. Watson Research Center. His research interests include performance evaluation of computer communication networks, broadband ISDN, and multimedia systems. Prof. Miyahara is a Fellow of IPSJ and an IEEE Fellow.