On Proxy-Caching Mechanisms for Cooperative Video Streaming in Heterogeneous Environments

Naoki Wakamiya, Masayuki Murata, and Hideo Miyahara

Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
{wakamiya, murata, miyahara}@ist.osaka-u.ac.jp

Abstract. In this paper, we investigate mechanisms in which proxy servers cooperate to provide users with low-latency and high-quality video-streaming services. The proxy is capable of adapting incoming or cached video blocks at the user's request by means of transcoders and filters. On receiving a request from a user, the proxy checks its own cache. If an appropriate block is not available, the proxy retrieves a block of a higher quality from the video server or a nearby proxy. The retrieved block is cached, its quality is adjusted to that which was requested as necessary, and is then sent to the user. Each proxy communicates with the others and takes the transfer delay and video quality into account in finding the appropriate block for retrieval. We propose several caching mechanisms for the video-streaming system and evaluate their performance in terms of the required buffer size, the play-back delay, and the video quality.

1 Introduction

To provide fascinating video content without irritating, disappointing, or discouraging the viewer, mechanisms that accomplish low delays and high quality for the streaming service are required. Proxy caching [1], which was originally proposed for WWW (World Wide Web) systems and is now widely deployed, provides savings on bandwidth, load balancing, reduced network latency, and better content availability. The technique thus appears to be helpful in the context of video steaming.

However, existing caching architectures are not intended for video streaming services; they are thus not directly applicable to video delivery. One problem in the caching of video streams is the size of video objects. For example, two hours of video encoded as MPEG-2 at the lowest coding rate and quality takes up 1.35 Gbytes. The buffer in the proxy-cache server is of limited size, it may, for example, only be capable of caching ten or so streams. Segmentation of the video stream, into `blocks`, `segments` or `frames`, provided the basis for the proposals made in earlier research work on proxy-cache mechanisms for video-steaming services [2,3,4,5]. Retrieving, caching, and sending video streams on a per-segment basis makes effective use of the cache buffer possible and obtains a high probability of cache hits.

A problem, however, that arises with video streams which are segmented in some way is the strict requirements on the timing of block delivery. Since conventional web

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-540-45812-8_28

K.C. Almeroth and M. Hasan (Eds.): MMNS 2002, LNCS 2496, pp. 127–139, 2002.

browsing does not impose any requirements in terms of temporal QoS, except for the so-called eight-second rule, the proxy is allowed ample time to find and retrieve requested objects. On the other hand, once a user starts to watch a video stream, the service provider must guarantee the continuous delivery of the blocks of video so that there are no interruptions or freezing during play-back.

A further issue is the heterogeneity among clients. The available bandwidth and the capacity of the access links differ from client to client. Client systems are also diverse in terms of their capabilities. Since receiving and decoding a video stream places large loads on the network and on the client systems, the affordable video rate, i.e., the quality, also differs from client to client. Users may have preferences in terms of perceived video quality. Two teams have introduced a proxy-cache server which is capable of video-quality adjustment as a solution to the heterogeneity issue [3,6]. The proxy adjusts the quality of retrieved or stored blocks to requests and provides clients with blocks of a satisfactory quality.

In this paper we propose proxy-caching mechanisms for streaming-video services. Several papers [2,3,4,5,6] describe valid ways of achieving a low-delay and high-quality video-streaming service. However, none provides a solution to all of the problems outlined above and explained in the next section. Our proposal is based on our own previous research [6], and takes into account the segmentation and prefetching of the data, and the client-to-client heterogeneity. More importantly, we propose a new mechanism for cooperation among video proxy-cache servers. In this paper, on the basis of research in that area, we describe a generally applicable idea for the provision of more effective and higher-quality video-streaming services in the *heterogeneous* environment. To the best of our knowledge, insufficient attention has been given to this point. We tackles the problem of how the requested block should be provided to the client, and what the quality of the block must be. The mechanisms proposed in the paper are somewhat general, but provide a good starting point from which we are able to see how several basic techniques may be combined to accomplish an effective video-streaming service in a heterogeneous environment.

The organization of the rest of this paper is as follows. In section 2, we present our assumptions on the video-streaming system and briefly summarize the related issues that have been considered in this work. The basic mechanism of proxy caching with video quality adjustment is introduced in section 3. In section 4, we give the results of some preliminary evaluations. Finally, we summarize the paper in section 5.

2 System Model and Related Issues

In this section, we start by illustrating our video-streaming system. We then summarize the issues in relation to a cooperative video-streaming service for an environment with heterogeneous clients.

2.1 A Video-Streaming System with Proxies

The video-streaming system considered in this paper consists of an originating video server, several heterogeneous client systems, and proxy-cache servers which are capable

of video-quality adjustment. Several video streams are prepared and placed in the local storage of the video server, which communicates with the proxies. Each proxy further communicates with a number of designated clients and neighboring proxies. Communications among the entities of this system for the video streaming are according to the RTSP/TCP protocol stack. The video streams are segmented into blocks. Delivery of the blocks from the originating server to the proxies and to the clients takes place in RTP/UDP sessions.

A client obtains information on a video-streaming service through the SDP. If the user decides to receive the service, the client sets up an RTSP session with the designated proxy-cache server, which is typically the closest such server to the client, and informs the proxy server about the client's requirements in terms of the video stream. The information includes the desired level of video quality, which is determined by both the user's preferences and the limitations of the client system, including its processing capability and any multimedia equipment that is available, the size of the dedicated buffer for prefetching, and, if possible, the available bandwidth.

On receiving the `PLAY` request from the client, the proxy searches its local cache for a corresponding block of video. If it does hold such a block and the proxy is able to satisfy the client request in terms of video quality (this constitutes a "cache hit"), the proxy applies any video-quality adjustment that is needed. Note that the cache is only hit if the block is both held and is of the same quality as has been requested or of higher quality. The block is then sent to the client in an RTP/UDP session. If, on the other hand, a "cache miss" occurs, the proxy retrieves the block from the most appropriate server, which is selected from among the the originating server and the other proxy-cache servers. The proxy then deposits the block in its local cache, adjusts the block's quality as required, and then sends the block to the client.

At the same time, the proxy predicts the blocks which will be required in the near future, and retrieves them in anticipation of client requests. This mechanism is called prefetching [3,6] and provides a way of avoiding breaks in the play-back of the video stream. Prefetching was originally developed for WWW systems in general, but is particularly effective when used with video objects.

The client is equipped with a prefetch buffer in which the delivered blocks are initially stored. Play-back by the client is delayed until sufficient blocks are thought to have been stored. The client then begins to read the blocks from the buffer for play-back. At the same time, the client deposits new blocks, as they arrive, in the buffer.

2.2 Related Issues and Relevant Previous Works

Several research issues are involved in the building of the video-streaming system which we have introduced in the previous subsection. In this subsection, we introduce those issues and some relevant work, and then explain the solutions we have selected.

Segmentation of video streams : When we considered the structure of a coded video stream and its frame-by-frame mode of play-back, we decided to use a block of several video frames in segmentation [6]. Such frame-based segmentation is preferable when we are employing RTSP as the communications protocol for video streaming. For the proxy to retrieve specific blocks from the other servers, a `PLAY` request must specify

the range of the video stream in terms of an SMPTE relative timestamp, a normal play timestamp, or an ISO 8601 timestamp. Employing frame-based segmentation allows us to easily derive appropriate range parameters from the order of a block in the sequence, since each block corresponds to a certain number of frames and we know the period of one frame, e.g., 1/24 second.

Adjusting video quality : Of the mechanisms for the adjustment of video quality, which are sometimes called transcoders or filters, we have tended to concentrate on the quantizer-scale-based approach [6]. The originating server generates a video stream of the finest possible quality by using the smallest quantizer scale, i.e., $Q = 1$. The originating server requantizes the blocks if the proxy's request was for block of lower quality and then sends them to the proxy. The proxy also adjusts cached and retrieved blocks when a lower quality has been requested by another proxy or the client. Note that we do not intend to limit the adjustment of video quality to the quantizer-scale-based technique, since this sometimes leads to sudden jumps in the level of quality due to diverse quantizer scale. Yeadon et al. [7] report on a range of useful and scalable ways of controlling the quality of encoded video streams. The layered coding algorithm inherently enables the quality adjustment.

Locating the appropriate server : To locate a server which is appropriate to handle a request for missing blocks, the originating server and the proxy-cache servers need to communicate with each other and to maintain up-to-date information. The information includes the blocks that are available and their quality, the round trip time and the available bandwidth between the proxy-cache server and the other servers. Extending the messages of the well-known inter-cache communications protocols, including ICP, CARP, and WCCP [1] to include information on the quality of the cached blocks and other performance metrics makes them applicable to video-streaming services.

The proxy estimates the block size, propagation delay, and throughput and thus obtains as accurate a prediction of the transfer time as is possible. The block size can be derived by using a traffic model for coded-video streams [8] or the information provided by the server through out-of-band communications. The latest research activity in network measurement, monitoring, and telecommunications engineering has provided us with several tools and methods for obtaining information on network conditions. Any of the methods may be applied to the proxy servers, as long as this is easy to apply and the selected method provides accurate estimates. For example, latency may be estimated by a means of *ping*, *echoping* and so on. The throughput of a session between two servers may be measured by using *pathchar*, *pchar*, or and any other suitable methods. Employing the TCP-friendly rate-control protocol [9,10] as the underlying rate-control mechanism simplifies the estimation of latency and throughput.

Cache management : Since the proxy server's cache buffer is of finite size, a less important cached block must sometimes be replaced by a newly retrieved block. On the contrary to the LRU and LFU algorithms which implicitly track the pattern of client requests [11], the algorithms proposed in earlier work [3,6] explicitly track the client requests and determine the blocks to be kept and to be thrown away on the retrieval of a new block. Those blocks which are located at the beginning of the video, those which are currently being watched by clients, and those which are considered likely to be requested in the near future are regarded as important and are not replaced. Of the

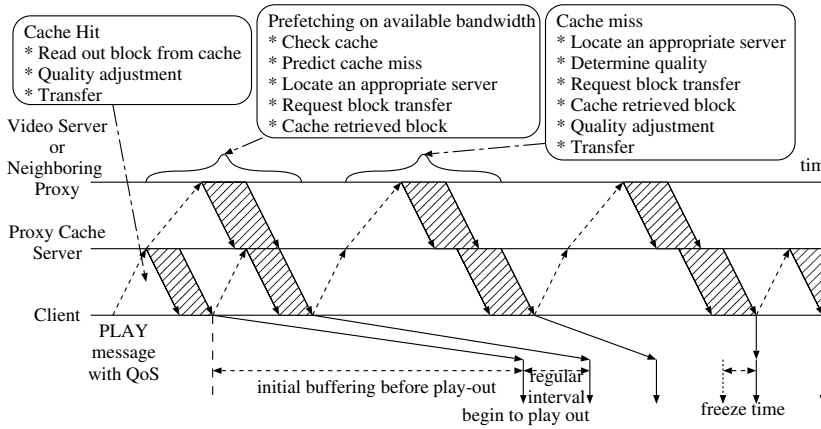

Fig. 1. Behavior of the proposed system

other blocks, those closer to the end of the stream are labeled as being lower priority and become the first victims when replacement is required.

Prefetching : Introducing prefetching mechanisms to the proxies and clients of the video-streaming system conceals the effects of variations in network conditions on delays in the transfer of data and increases the time that is available for finding and retrieving requested blocks with the desired quality. The proxy prefetching mechanism we present in this paper predicts future cache misses and prefetches blocks of the appropriate quality on the basis of requests from the client request and of the number of blocks in the client's prefetch buffer.

3 A Proxy Cache with Video-Quality Adjustment

Our system consists of an originating server, cooperative proxies, and heterogeneous clients. To simplify the discussion, we have concentrated on the case where clients concentrate on a single video stream. Our mechanism is, however, applicable to cases where the video-streaming system is providing multiple video streams to individual users. The basic behavior of the system is illustrated in Fig. 1. Although video streaming according to the RTSP does not necessarily require block-by-block requests, we have assumed that the client explicitly informs the proxy server of a requested level of quality per block. Of course, we are also able to employ implicit quality notification, where the server continues to apply the most recently requested level of quality to subsequent blocks and does not wait for new requests.

3.1 The Cache Table and Remote Table

The proxy server has two tables. Information on the locally cached blocks is maintained in one, referred to as the cache table, while the other, the remote table, is for information

on the caching of blocks at other servers. One such pair of tables is organized for each video stream.

The cache tables maintain information on the cached and non-cached blocks of video streams. A block consists of N frames. Each entry in the cache table corresponds to one block of the stream and consists of a block number i, the quality of the block $q(i)$ in the cache, and a list of markers $M(i)$ for the cached block i. Larger values of $q(i)$ indicates higher levels of video quality while a zero means that the block i is not cached in the local buffer. After a block has been received, the corresponding $q(i)$ is replaced by the quality of the newly retrieved block. The list of markers $M(i)$ is a set of identifiers of proxy servers.

The remote table in a given proxy holds information on the other servers. Each entry in the remote table corresponds to the name of a server (e.g., server k, $k = 0$ indicates the originating server). It includes the estimated latency d_k^S and the estimated throughput r_k^S for the connection, along with the list of blocks available on that server and their respective quality $O_k(i)$. The table is built and maintained by exchanging `QUERY` and `REPLY` messages.

When a proxy receives a first request from a new client, it sends a `QUERY` message to a multicast address which is shared by all of the servers. The proxy sets a TTL for the `QUERY` message and thus limits the scope of neighboring servers that will respond to the query as in SOCCER [2]. The `QUERY` message informs the other servers of the video stream which the proxy is requesting and a list of the blocks it expects to require in the near future. On receiving the query, the proxy server k searches its local cache for the blocks which are listed in the `QUERY` message, and sets marks $M(i)$ for the blocks in its cache table. After a block has been marked, it leaves the scope of block replacement. Those blocks not being declared in the message are un-marked for the proxy. The server k then returns a `REPLY` message which carries a timestamp and the list of $O_k(i) = q(i)$ values for those blocks it holds which were in the list included with the query. The blocks indicated in the inquiries about quality are those which are currently requested by clients and subsequent I blocks. The window of inquiry I is defined as a way of restricting the range of blocks to include in a single inquiry and preventing all blocks from being locked. I blocks from the beginning of the stream are also listed in the `QUERY` message to prepare for new clients that will request the stream in the future [5].

The interval at which a proxy sends `QUERY` messages must be neither too short nor too long. The proxy issues a `QUERY` message when the end of the prefetching window, which is later explained, of any client finds an entry $O_k(i)$ which is zero. In addition, we also introduce a timer to force refreshing of the remote tables. Timing reaches expiry every $(I - P - 1)N/F$ where P is the size of the prefetching window, N is the number of frames in a block and F the frame rate, respectively.

3.2 Block Retrieval Algorithm

The block-retrieval algorithm determines the levels of quality of the blocks that will satisfy client requests and the way that the proxy-cache server provides the block to the client. The client sends request messages to its designated proxy server on a per-block basis as shown in Fig. 1. The client j's request for the block i is denoted by $q_j(i)$.

On receiving a request $q_j(i)$, the proxy determines how the requested block should be provided to the client on the basis of the cache table, the remote table, the available bandwidth r_j^C, the latency d_j^C to the client j, the number of video blocks stored in the client's prefetch buffer p_j, and the client's persistence or tolerance on the video quality degradation β_j. The bandwidth r_j^C which is available for sending the block i to the client j and the one-way delay d_j^C for this transfer are determined in the same way as the estimated throughput r_k^S and the estimated latency d_k^S for the server k. The number of video blocks in the client j's prefetch buffer is denoted by p_j and is included in the request from the client to the proxy. At the beginning of a session with a proxy by a client, the proxy is notified of the client's persistence on the video quality, i.e., the parameter β_j. When the client chooses $\beta_j = 1$, the proxy is obliged to provide the client with blocks of a level of quality that is in accordance with the request. On the other hand, a value of β_j close to zero means that the client is tolerant of lower quality but is placing tight restrictions on the punctual or in-time arrival of blocks. Finally, we assume that the proxy knows the size of a block i with a level of quality q, $s(i, q)$.

The quality, for a block i, that the proxy is able to offer to a client j is derived as

$$q_j^P(i) = \min(q(i), q_j^{Pmax}(i)), \tag{1}$$

where $q_j^{Pmax}(i)$ indicates the maximum quality with which the block i is deliverable to the client j without making the client's prefetch buffer underflow. Reading of video data from the prefetch buffer is block by block and a reception of a block must be completed before the first frame of that block is required. The sending of a next required block to the client j must be finished before all blocks which are stored in the client's prefetch buffer have been consumed. Thus $q_j^{Pmax}(i)$ is derived by using the following equation;

$$q_j^{Pmax}(i) = \max(q|2d_j^C + \frac{s(i,q)}{r_j^C} \leq \frac{p_j N}{F} - \Delta). \tag{2}$$

Here, Δ has been introduced to avoid risky control and to absorb unexpected delay jitter and errors in estimation of delay and throughput. At the beginning of the session, the client employs a large value of p_j, independently of the actual number of blocks in the prefetch buffer, and intentionally sets the quality $q_j(i)$ to as low as possible in order to store as many blocks as possible to avoid unexpected delays. This is maintained until sufficient blocks have been prefetched or the client begins play-back. On receiving a request $q_j(i)$, the proxy starts by examining its own cache table. If $q_j^P(i) \geq \beta_j q_j(i)$ (i.e., the cache is hit), the proxy is able to satisfy the client by providing a cached block. The block is immediately sent to the client with a level of quality $\hat{q}_j(i) = \min(q_j(i), q_j^P(i))$.

If the cache is missed and the given proxy decides to retrieve the block from the other server, the given proxy sends a request message to the proxy chosen by algorithms that will be described later. The proxy keeps track of requests it has sent out in lists of the form $Q_k = \{q_n^{Q_k}\}$. The nth entry in such a list, $q_n^{Q_k}$, corresponds to the quality setting of the nth request sent to a server k for the retrieval of a block b_n. An entry is added to the list Q_k whenever the given proxy sends a request to the server k and is removed from the list on completion of the retrieval of the block. The lists are referred to when the cache is missed on the client's request. The proxy compares $q_j(i)$ with the quality

of the blocks that the proxy has already been requested of the other servers. To save on bandwidth, the proxy provides the client with the block in the list if the mth request in the list Q_k is for the block i, i.e., $b_m = i$, the quality satisfies the client request, i.e., $q_m^{Q_k} \geq \beta_j q_j(i)$, and the given proxy is expected to finish retrieving the block in time, i.e., $2d_j^C + \frac{\sum_{n=1}^{m} s(b_n, q_n^{Q_k})}{r_k^S} \leq \frac{p_j N}{F} - \Delta$. The quality of the block i provided to the client j is given as $\hat{q}_j(i) = \min(q_j(i), q_m^{Q_k})$.

Otherwise, the proxy retrieves the block of desired quality from an appropriate server. The quality of block j that a server k can offer is derived by using the following equation.

$$q_{k,j}^S(i) = \min(O_k(i), q_{k,j}^{Smax}(i)) \tag{3}$$

The maximum quality $q_{k,j}^{Smax}(i)$ of a block i such that there is no starvation of the client j's prefetch buffer is derived by using

$$q_{k,j}^{Smax}(i) = \max(q | 2d_j^C + \max(\frac{\sum_n s(b_n, q_n^{Q_k})}{r_k^S}, 2d_k^S) + \max(\frac{s(i,\hat{q})}{r_j^C}, \frac{s(i,q)}{r_k^S}) \leq \frac{p_j N}{F} - \Delta) \tag{4}$$

If there is no request in the list Q_k, $\sum_n s(b_n, q_n^{Q_k})$ becomes zero. In the same way as with $\hat{q}_j(i)$, the quality of the block to be provided to the client by the proxy, $\hat{q}$, is determined once the quality of the block that is provided to the proxy by the server, q, has been specified. The proxy sends the request for the block to that server among all of the neighboring servers, i.e., candidates to supply the block, which holds the block at the highest quality. To avoid retrieving a block of unnecessarily high quality, the quality requested for the block, $q_{k,j}(i)$, is limited to $\max_{l_m \leq i} q_m(l_m)$ where l_m corresponds to the block identifier requested by the client m that joined the service later than the client i. That is, the proxy retrieves the block of the minimum quality such that the future requests from the clients that succeed the given client on the block will hit its cache. If multiple servers are capable of supplying the block at the same quality, the proxy chooses the fastest of the servers.

If no server is capable of delivering the block in time and at a satisfactory level of quality, the proxy determines the server k which is capable of most quickly providing the given proxy with the block i at the quality $\beta_j q_j(i)$. In this case, the buffer starves and the continuity of video play-back is broken. The client j must wait for the reception of the block i over a period, which is referred to as the freeze time, given by

$$f_j(i) = \max(0, 2d_j^C + \max(\frac{\sum_n s(b_n, q_n^{Q_k})}{r_k^S}, 2d_k^S) + \frac{s(i, \beta_j q_j(i))}{\min(r_j^C, r_k^C)} - \frac{p_j N}{F} + \Delta) \tag{5}$$

3.3 Block Prefetching Algorithm

To achieve even more effective control, the proxy prefetches blocks in accordance with client requests. On receiving a request for a block i, the proxy consults its cache table and checks the lists Q^k from block $i+1$ to $i+P$ to find a potential future cache miss. The parameter P determines the range of the prefetching window. When the proxy finds that a block $j > i$ has been cached or requested in a form with a lower level of quality than $\beta_j \cdot q_j(i)$, it attempts to prefetch the block in a finer quality form. If two or more such unsatisfactory blocks are detected, only that block which closest to the ith

block is chosen and prefetched. A request for prefetching that has arrived at the server k overwrites the preceding prefetch request and will be served only when no request for typical block retrieval of cache-missed blocks is waiting.

The quality of the block m that the server k is able to offer to the client j is derived by using the following equation;

$$q_{k,j}^{R}(m) = \min(O_k(m), q_{k,j}^{Rmax}(m)) \tag{6}$$

where $q_{k,j}^{Rmax}(m)$ stands for the maximum quality which is deliverable, in time, to the client in time and is given as

$$q_{k,j}^{Rmax}(m) = \max(q|d_p \leq \frac{(p_j + m - i)N}{F} - \Delta) \tag{7}$$

Due to limited space, we omitt the detailed explanation of prefetching mechanisms. The expected transfer delay d_p is appropriately derived taking into account the condition, i.e., how block i is provided to the client j. For each of candidate servers, the offerable quality of the block m is calculated. In a similar way to the retrieval of a block i, the most preferable server is chosen on the basis of the quality of available blocks and the block-transfer delay.

3.4 Cache Replacement Algorithm

The proxy makes a list of blocks to be discarded on the basis of its cache table. Those blocks which are located in windows of inquiry and marked by the other proxy servers are placed beyond the the scope of replacement. Reference to the blocks about which the inquiry was made is expected in the near future, i.e., in IN/F seconds. The marked blocks should not be dropped or degraded in order to maintain consistency of the remote tables that the other proxies hold. In addition, retention of the first I blocks is also considered important as a way of hiding the initial delay [5]. The rest of the blocks are all candidates for replacement. Of these, the block which is closest to the end of the stream is the first to be a 'victim', i.e., to be assigned a reduced level of quality or discarded. Let n be the identifier of this block.

The proxy first tries lowering the quality of the victim as a way of shrinking the block size and making a room for the new block. The space which may result from this adjustment is limited to $\max_{l_m<n} \beta_m q_m(l_m)$, that is, to the highest value for minimum tolerable quality in the client requests for blocks that precede the victim. If the adjustment is not sufficiently effective, the victim is discarded from the cache buffer. If this is still insufficient, the proxy moves on the next victim.

When all of the candidate victims have been dropped and there is still not enough room, the proxy identifies the least important of the protected blocks. The last P blocks of the successive marked blocks are only for use in prefetching and may thus be considered less important since the prefetching is given a lower priority by the servers, as was explained in subsection 3.3. The proxy starts the process of trying degradation of quality and dropping blocks at the block closest to the end of a video stream. Finally, if all of the attempts described above fail, the proxy gives up on storing the new block.

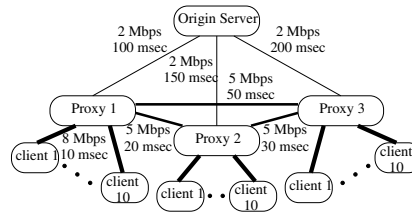

Fig. 2. The model used in simulation

4 Evaluation

In this section, we describe some of the results we obtained in our preliminary experimental evaluation of the system described above. The network we employed is depicted in Fig. 2. The originating server is behind a wide-area network or the Internet, and the proxy servers are located in an ISP network. There are ten clients in the local network of each proxy server. The one-way delays and the available bandwidth for each session are indicated beside the respective connections. Although the available bandwidth and one-way delay fluctuate greatly under realistic conditions, we have employed static values so that we are able to clearly observe the behavior obtained by applying the control mechanisms in various way. Clients start their subscriptions one after another in accordance with the given numbers and the inter-arrival time of the first `PLAY` messages issued by the clients follows an exponential distribution with an average of 30 minutes.

The video stream is 90-minutes long and is played back at 30 fps. The stream is divided up into one-second blocks, that is, $L = 5400$ and $N = 30$. The one-second block is too short and is a somewhat unrealistic assumption, but our proposed mechanism still provides satisfactory performance improvement under this difficult condition, as is described later. Ten levels of quality are available and the respective block sizes are given by $\forall i\ s(i,q) = \frac{qN}{F} \times 10^6$ bits. The window sizes are $P = 10$ and $I = 20$ blocks. These windows mean that a proxy predicts a cache miss ten seconds in advance of the block being required and that `QUERY` and `REPLY` messages are exchanged every ten seconds. The time prepared for errors Δ is set to 2 seconds and the proxy tries to leave 60 frames in the client's prefetch buffer. The period of initial waiting time is set to 4 seconds, that is, the client waits for 4 seconds before beginning to play a requested stream back. During the initial wait, the requested quality $q_j(i)$ is set to its lowest level, and the client then randomly determines the quality requirement on a block-by-block basis. In the experiments, a probability of 10% was assigned to each event of the client increasing or decreasing $q_j(i)$ by one, while retaining the same value for $q_j(i)$ was assigned a probability of 80%.

In the following description, we focus on the performance which was observed at proxy 2 and describe the results for the four different schemes which we applied for purposes of comparison. One is referred to as "Independent w/o Prefetch"; in this approach proxies always retrieve the missing or unsatisfactory blocks from the originating video server and proxy prefetching is not employed. "Independent w/ Prefetch" corresponds

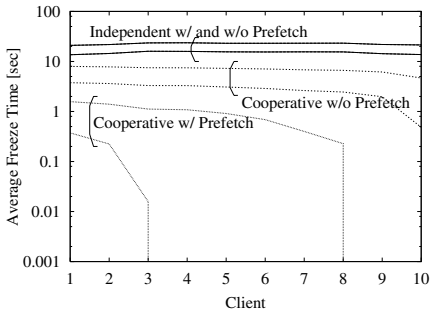

Fig. 3. Average freeze time

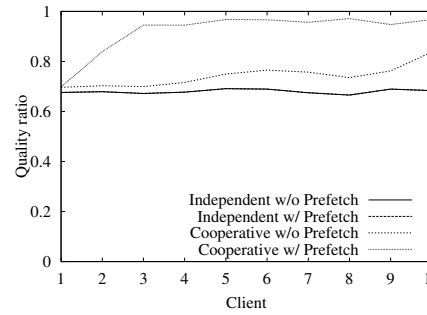

Fig. 4. Average quality ratio

to the case where independent proxies are coupled via the prefetching mechanism and retrieve blocks from the originating server in accordance with client requests. "Cooperative w/o Prefetch" and "Cooperative w/ Prefetch" indicate the cases where the proxies cooperate in providing the stream of video data. The prefetching mechanism is absent from the former approach and present in the later. The "Cooperative w/ Prefetch" corresponds to our proposed mechanism. In all four schemes, proxies are capable of video-quality adjustment to accomplish the effective use of cached data.

4.1 Infinite Buffer Case

We start by considering the case where the proxy servers are equipped with infinitely large cache buffers. A cached block is thus never discarded, but is replaced by a block of the same sequence number and higher quality. Figure 3 summarizes results for cases where the clients persist in guaranteeing the quality requirements, i.e., $\beta = 1.0$, and where the clients are made tolerant of quality degradation, i.e., $\beta = 0.6$. For each pair of lines for a scheme, the upper line corresponds to the case of $\beta = 1.0$ and the lower does to the case of $\beta = 0.6$. Results for the independent schemes are almost the same and it is impossible to tell difference. The comparison is in terms of the average freeze time $f_j(i)$ (see Eq. (5)).

Figure 3 shows that, the average freeze time of the independent schemes is much greater than that of the cooperative scheme regardless of the prefetching mechanism. This is because the proxies in the independent schemes are forced to retrieve blocks from the distant originating server even though other proxy servers may be holding the blocks in their local caches and thus most of the bandwidth between the originating server and the proxy is taken up by the retrieval of blocks. The performance of the cooperative scheme is improved when the proxies prefetch blocks for which a cache miss would otherwise be expected. However, the maximum of the average freeze times for our proposed mechanism is quite long, at 1.57 seconds. This value is for client 1, which is the first client of proxy 2. Since there are no corresponding blocks in the cache of proxy 2's cache, the client is forced to wait for about 1.57 seconds every time it move on the next block.

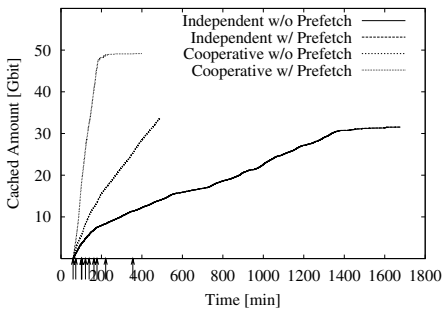

Fig. 5. Amount of cached data

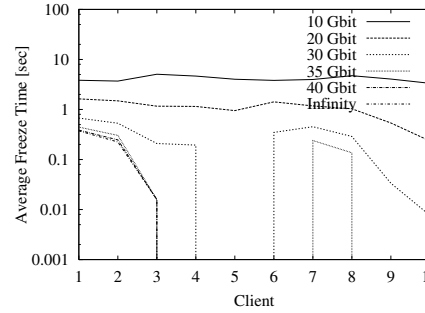

Fig. 6. Variation in average freeze time

The freeze time is improved by the lower value of β, regardless of the scheme which is applied. With our scheme in particular, the maximum freeze time fell to 0.37 seconds for client 1. A continuous video-streaming service was provided to the client 4 to client 10 since blocks of sufficient level of quality are stored in the cache buffers. In addition, there is less loss of quality and the clients are able to enjoy a higher quality of video streaming. Figure 4 depicts the average quality ratio which was defined as the ratio of the quality of the delivered block to the requested level of quality in the case of $\beta = 0.6$. For example, the ratios of quality are 0.70 for the first client and 0.97 for the last, even though a ratio of 0.60 is approved. Even if $\beta = 0.2$ is chosen, the first user suffers from a ratio of 0.40, while the others are able to enjoy video play-back with a quality ratio that is more than 0.85. On this basis, we conclude that intentionally employing a lower β leads to a comfortable streaming service with no unnecessary loss of video quality.

4.2 Limited Buffer Case

Figure 5 depicts transitions in the amount of data cached by proxy 2. The arrows on the x-axis indicate instances where the first `PLAY` requests have been issued by clients. As shown in the figure, the size of the buffer required for our scheme is about 49 Gbits. Figure 6 summarizes the results of a comparative evaluation of various cache capacities for use with our scheme. Comparison between Figs. 3 and 5 shows that as long as the buffers in the servers for our scheme are larger than 20 Gbits, the average freeze times for our scheme are obviously shorter than those for the other schemes. For clients 1 to 3, increasing the buffer capacity from 35 Gbits provides a small reduction in discontinuities in video play-back. The other clients, however, are freed from annoying breaks when each proxy is equipped with a cache buffer that has a capacity of more than 40 Gbits. With 35-Gbit buffers, clients 7 and 8 suffer from freezing that lasts for a few hundreds milliseconds. This is because cached blocks are being degraded to make a room for freshly retrieved blocks but these two clients require blocks of higher quality than do the preceding clients. Thus, we are able to conclude that the preferable buffer capacity in this case is as much as 40 Gbits. Note that further performance improvements would be expected if the users choose lower value for β.

5 Conclusion

In this paper, we have proposed a new scheme for video streaming in which proxies that are capable of video-quality adjustment retrieve, prefetch, cache, and deliver video blocks in a cooperative manner. Simulation demonstrated that our scheme is effective in reducing the freeze times and in providing high-quality blocks to clients. Our scheme is applicable to any video-streaming service that employs a coding algorithm which allows segmentation and adjustment of quality. However, areas for further research remain. There is room for improvement, for example, in terms of the protection of blocks against replacement, which leads to a problem of scalability. We have to further investigate the practicality of our scheme, since several assumptions were made for the control mechanisms. In addition, the effect of such control parameters as I, P, β, δ, and the initial waiting time on performance was not fully evaluated in work reported in this paper. We will be able to derive an algorithm to determine these effects in an appropriate way. Furthermore, the system model employed in this evaluation is somewhat artificial. For this reason, we want to apply our scheme to a more generic network model.

References

1. G. Barish and K. Obraczka, "World Wide Web caching: Trends and techniques," *IEEE Communications Magazine*, pp. 178–185, May 2000. 127, 130
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. 127, 128, 132
3. R. Rejaie, H. Yu, M. Handley, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the Internet," in *Proceedings of IEEE INFOCOM 2000*, March 2000. 127, 128, 128, 129, 130
4. Z. Miao and A. Ortega, "Proxy caching for efficient video services over the Internet," in *Proceedings of Packet Video Workshop'99*, April 1999. 127, 128
5. S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of IEEE INFOCOM'99*, vol. 3, pp. 1310–1319, March 1999. 127, 128, 132, 135
6. M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy caching mechanisms with video quality adjustment," in *Proceedings of SPIE International Symposium on The Convergence of Information Technologies and Communications*, vol. 4519, pp. 276–284, August 2001. 128, 128, 128, 129, 129, 130, 130
7. N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, "Filters: QoS support mechanisms for multipeer communications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1245–1262, September 1996. 130
8. A. M. Dawood and M. Ghanbari, "Content-based MPEG video traffic modeling," *IEEE Transactions on Multimedia*, vol. 1, pp. 77–87, March 1999. 130
9. S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications: the extended version," *International Computer Science Institute technical report TR-00-003*, March 2000. 130
10. M. Miyabayashi, N. Wakamiya, M. Murata, and H. Miyahara, "MPEG-TFRCP : Video transfer with TCP-friendly rate control protocol," in *Proceedings of IEEE International Conference on Communications 2001*, vol. 1, pp. 137–141, June 2001. 130
11. M. Reisslein, F. Hartanto, and K. W. Ross, "Interactive video streaming with proxy servers," in *Proceedings of First International Workshop on Intelligent Multimedia Computing and Networking*, vol. II, pp. 588–591, February 2000. 130