Video Streaming Systems with Cooperative Caching Mechanisms

Naoki Wakamiya, Masayuki Murata, Hideo Miyahara

Graduate School of Information Science and Technology
Osaka University
Toyonaka, Osaka 560-8531, Japan

ABSTRACT

In this paper, we investigate mechanisms for the video streaming system where proxies cooperate to provide users with low-latency and high-quality services under heterogeneous and mobile environment where hosts have different capabilities and dynamically change their locations. The proxy is capable of adapting incoming or cached video data to user's demand by means of transcoders and filters. With such proxies, heterogeneous QoS requirements on the delivered stream can be fully satisfied by preparing high-quality video data in the local cache buffer and adjust them to the requirements. On receiving a request from a user, the proxy first checks the cache. If no appropriate data is available, it retrieves the video data of the satisfactory quality from the video server or proxies nearby. The proxy communicates with the others and finds an appropriate one for data retrieval by taking into account the transfer delay and the video quality. We propose a cooperative video caching mechanism for the video streaming system and evaluate the performance in terms of the delay introduced and the video quality.

Keywords: video proxy cache, heterogeneous environment, quality adjustment, mobile computing

1. INTRODUCTION

To provide video streams for users without irritating, disappointing, and discouraging them, we should have mechanisms to accomplish a low-delay and high-quality streaming service. The proxy caching originally proposed for WWW (World Wide Web) systems seems also helpful to video streaming systems for bandwidth saving, load balancing, network latency reduction, and high content availability.

There have been several research works in applying proxy caching mechanisms to video streaming services.[1–5] They clarified problems related to efficient and effective video caching. One among them is the size of video object. For example, two hour long MPEG-2 video amounts to 1.35 GBytes when it is coded at the minimum quality of 1.5 Mbps video rate. Since the proxy is equipped with the limited buffer, it can cache only ten and some streams. In addition, retrieving, caching, and replacing video data are performed per whole-stream basis, one cannot expect efficient use of buffer space.

Another is severe requirements on timing constraints. Once a user starts to watch a video stream, the service provider should guarantee continuity of the video playback. Each frame or block of the stream must arrive at a client before a user watches the part. There should be some control mechanisms to guarantee punctual arrivals because network bandwidth is not alway sufficient to transfer video data in time and there are jitters in the transfer delay.

The other is the heterogeneity among clients. The available bandwidth and the performance of end-systems differ among clients. Since receiving and decoding video stream put much loads on both of the networks and end-systems, the affordable video rate and quality also differ from client to client. In addition, users may have preferences in the perceived video quality.

Further author information: (Send correspondence to Naoki Wakamiya)
E-mail: {wakamiya, murata, miyahara}@ist.osaka-u.ac.jp, Telephone: +81-6-6850-6586

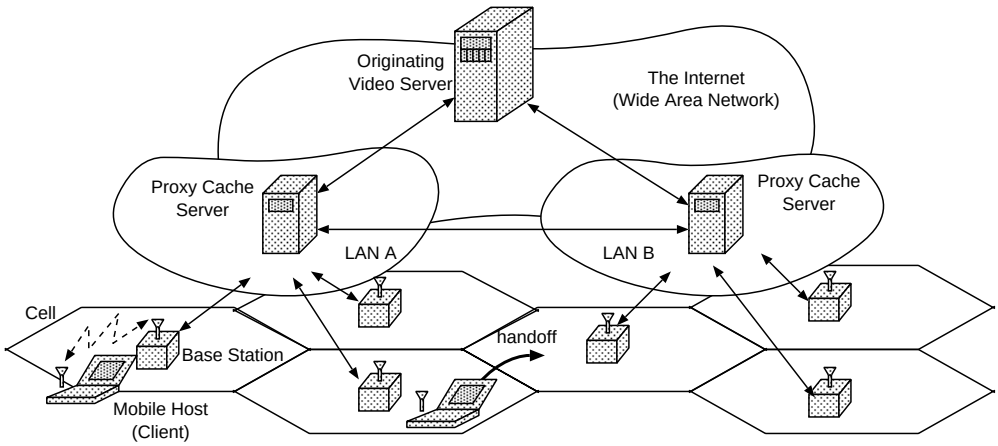

Figure 1. Cooperative Video Streaming System

The client mobility is a new problem that recently arises. With the proliferation of wireless communications technologies, users come to enjoy a streaming service while moving around using laptops equipped with a wireless network card. For such environment, proxies should prepare for the client movement in order to provide a high-quality and continuous video streaming service.

In this paper, we describe a generally applicable idea for the provision of effective and higher-quality video-streaming services in the heterogeneous and mobile environment. 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 benefits of applying our proxy-cooperation mechanism include saving on bandwidth, balancing of the load on the server, a perceived reduction in network latency, and a higher degree of content availability by cooperative caching. The scheme also provides support for heterogeneity from client to client and higher level of video quality. Furthermore, our scheme takes into account client mobility. 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 rest of the paper is organized as follows; In section 2, we present assumptions on the system and our proposed mechanisms are explained. In section 3, we show results of some preliminary experiments and finally we summarize the paper in section 4.

2. COOPERATIVE PROXY CACHING MECHANISMS

The video streaming system we consider in the paper is illustrated in Fig. 1. The system consists of an originating video server, cooperative proxy cache servers, a number of base stations, and heterogeneous mobile clients. Client hosts exist in wireless network environment and move around during a video streaming session. Client hosts are heterogeneous in regard to the available bandwidth, propagation delay from the proxy cache server, end-system performance, and user's preferences on the perceived video quality. A base station is located at the center of a cell and belongs to a LAN by means of a wired link. There exists a proxy cache server in each LAN and takes on resposibility for providing clients in corresponding wireless networks with video streaming services.

A client and a designated proxy cache server communicates over an RTSP session and video data are transferred over an RTP session established among them. A proxy cache server retrieves video data from a distant origin server or neighboring proxies on behalf of clients, deposits the data in its local cache buffer, and adapts the quality of the data to user's demand. In addition, proxy cache servers communicate with each other using inter-cache communications protocols, including ICP, CARP, and WCCP.[6] Servers exchange blocks over a pair of an RTSP and an RTP sessions. We should note here that to reduce loads on networks and servers, there is only one pair of sessions between any pair of servers regardless of the number of clients. Each pair of system

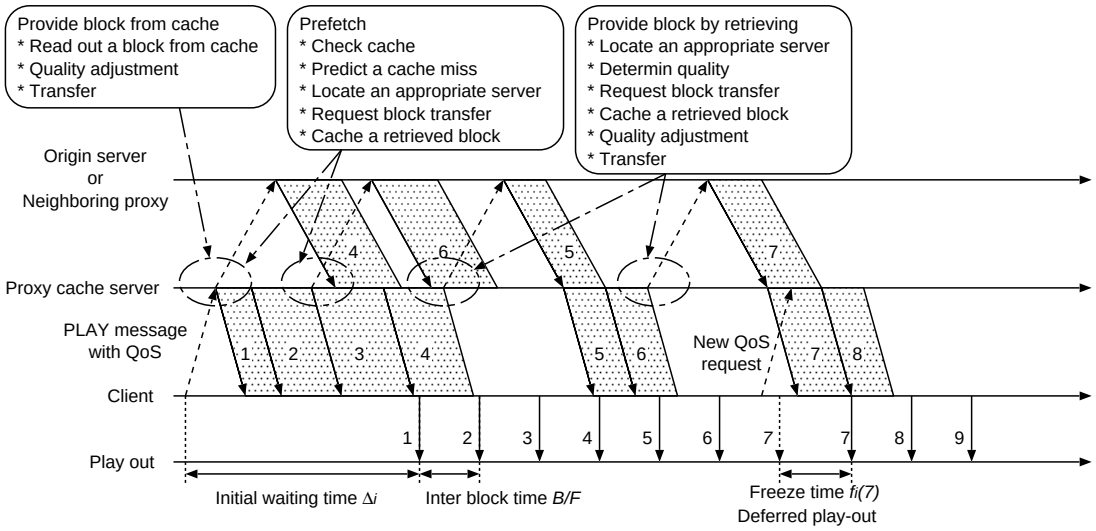

Figure 2. Behavior of the proposed system

entities also communicates with each other using out-of-band control sessions of RTCP. By exchanging RTCP messages, they can keep up-to-date information about the network condition, that is, the available bandwidth and the expected propagation delay.

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.[7] This was originally intended for use with MPEG-2 video streams but is also applicable to streams of data produced by such other coding algorithms as motion JPEG, MPEG-1, and MPEG-4. 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.[8] 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.[9,10]

In the following subsections, we propose several mechanisms for the video streaming system where proxies cooperate to provide users with low-latency and high-quality services under heterogeneous and mobile environment where hosts have different capabilities and dynamically change their locations. 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.

2.1. Data Transfer among Servers and Client

Figure 2 illustrates how servers and client communicate with each other in our video streaming system. The video streaming service is initiated by an RTSP `PLAY` message issued by a client to a designated proxy cache server. The message contains information about preferable (upper-constraint) and tolerable (lower-constraint) levels of video quality which are determined in accordance with the available bandwidth, end-system performance, and user's preferences. The client is allowed to dynamically change QoS requirements during a video session to reflect changes of those constraints.

In our system, a video stream is segmented into *blocks* to efficiently utilize the cache buffer and reduce the required bandwidth for data retrieval.[5,7] 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 frames in segmentation. 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 required part 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.

On receiving the first `PLAY` message, the proxy cache server begins to provide a requested video stream to the client. The proxy adopt the fastest way that can provide the client with a block of higher level of quality. The proxy can (i) read out and send a cached block, (ii) use a block being received, (iii) wait for the preceding request for the same block to be served, or (iv) newly retrieve a block from the other server. If the proxy has a block of the same number and its quality can satisfy the request, the proxy may consider using the cached block is the best way. It applies the video quality adjustment to the block as necessary and transfers the block to the client over an RTP session established between them. If the block being received has the satisfactory quality, the proxy may decide to send the block under transmission to the client. If there is the preceding request for the same block of higher quality, the proxy can wait for the request to be served. In some cases the proxy determines to retrieve the block. It first identifies an appropriate server among candidates, then determines the quality of the block to retrieve, and sends a `PLAY` message to the server. A retrieved block is cached and sent to the client after being applied the video quality adjustment as necessary. When the proxy cache server finishes sending out the block to the requesting client, it moves to the next block and repeat the similar procedures.

While providing clients with video blocks, the proxy cache server predicts and prepares for a future potential cache miss by prefetching a block from the other server. A prefetching request is processed without disturbing normal block retrievals. Details of the prefetching mechanisms will be given in subsection 2.3.

To avoid or absorb unexpected block transmission delays, a client i first defers playing out a received video block for a predetermined waiting time, denoted as Δ_i in the paper, as shown in Fig. 2. Then, it begins to decode and display received blocks one after another, at regular intervals of B/F where B corresponds to the number of frames in a block and F stands for the frame rate. Once a user begins to watch a video, the video streaming system must provide the client with video blocks in a continuous and timely manner. However, in some cases, a block does not arrive in time and the client is forced to pause. Time required for the client i to wait for the arrival of necessary block j is called *freeze time*, $f_i(j)$ in the paper. If the block j arrives in time, freeze time $f_i(j)$ becomes zero. Cache miss is one of reasons that make continuous and in-time delivery of video data difficult. Even if all blocks of a high level of quality is cached, a proxy cache server cannot guarantee the in-time delivery due to unexpected and uncontrollable network congestion.

2.2. Block Provisioning Algorithm

In providing a client with a block, a proxy cache server can send a cached block, apply a receiving block, make use of the preceding request, and retrieve a block. Among them, a proxy chooses the best way in terms of the offerable quality and the block transfer time. For this purpose, servers communicate with each other and maintain the up-to-date information, such as the quality of offerable blocks, round trip time, and available bandwidth. To predict the transfer time as accurate as possible, the proxy estimates the block size, propagation delay and throughput. The block size can be derived using the traffic model of coded video streams[11] or the information provided by the origin server. The latency can be estimated by a means of *ping*, *echoping* and so on. Throughput of a session between two servers can be measured by *pathchar*, *pchar* and any other methods. When we employ the TCP-friendly rate control protocol[12] as an underlying rate control mechanism, estimation of latency and throughput becomes simpler. However, to accurately estimate the network condition, pairs of servers should keep transferring data and periodically exchanging RTCP messages. Information related to the network condition among the proxy and clients are also required.

Assume that the proxy k is trying to provide the client i with the block j at time t. The quality of the block j must be higher than $q_i(j)$ and as high as $Q_i(j)$, which are determined by the QoS requirements specified by the latest `PLAY` message. A deadline $T_i(j)$ that the client i should finish receiving the block j is determined as

$$T_i(j) = T_i + \Delta_i + (j-1)\frac{B}{F} - \delta_i + D_i(j-1), \tag{1}$$

where T_i indicates an instant when the client i issues the first `PLAY` message, Δ_i is the initial waiting time, B and F stand for the number of frames in a block and the frame rate, respectively. δ_i is introduced to absorb unexpected delay jitters and estimation errors. Even if we carefully design control mechanisms and a streaming system is successfully built, we cannot avoid freezes. After a freeze, the client decodes and displays the succeeding blocks at regular intervals as shown in Fig. 2. $D_i(j-1)$ is accumulated freeze time and is given as $D_i(j-1) = \sum_{l=1}^{j-1} f_i(l)$.

The offerable quality of the block j to the client i with the block j cached at the proxy k's buffer, $c_{k,i}^{PC}(j)$, is derived as

$$c_{k,i}^{PC}(j) = \min(q_k(j), \bar{c}_{k,i}^{PC}(j)), \tag{2}$$

where

$$\bar{c}_{k,i}^{PC}(j) = \max(q | t + d_{k,i}^{PC}(t) + \frac{a_j(q)}{r_{k,i}^{PC}(t)} \leq T_i(j)). \tag{3}$$

$q_k(j)$ stands for quality of the block j cached at the proxy k's buffer. For this purpose, the proxy has a *cache table* which maintains information about cached and non-cached blocks of a video stream. Each entry of the table consists of the block number j and the quality of the cached block $q_k(j)$. Larger values of $q_k(j)$ indicates higher quality and zero means a non-cached block. $a_j(q)$ is a function that gives the size of block j of quality q. If $j = 0$ or $q = 0$, $a_j(q)$ gives zero. Thus, $c_{k,i}^{PC}(j)$ becomes zero if a corresponding block is not cached at t. $d_{k,i}^{PC}(t)$ and $r_{k,i}^{PC}(t)$ are estimates of available bandwidth and one-way propagation delay between the proxy k and the client i at t. Thus, $d_{k,i}^{PC}(t) + \frac{a_j(q)}{r_{k,i}^{PC}(t)}$ provides estimated time required for the client i to receive a whole of the block j of quality q via an RTP session whose available bandwidth is $r_{k,i}^{PC}(t)$ and propagation delay is $d_{k,i}^{PC}(t)$.

In this paper, we only consider the case where the quality is expressed by a single scalar, i.e., the reciprocal of the quantizer scale. Several parameters may be combined together to provide a more flexible and convenient service. For example, the three-member tuple (R, F, Q) corresponds to those scalabilities in the MPEG-2 coding algorithm which directly affect the perceived video quality. They are the spatial scalability R pixels, the temporal scalability F frames per sec, and the SNR scalability Q (quantizer scale).

If the block j is being received, the proxy can use it. The quality offerable is derived as

$$b_{k,i}^{PC}(j) = \max_{\forall s, s \neq k} (\min(q_{s,k}^{SP}(t), \bar{b}_{k,i}^{PC}(j))), \tag{4}$$

where

$$\bar{b}_{k,i}^{PC}(j) = \max(q | t + d_{k,i}^{PC}(t) + \max(\frac{a_j(q_{s,k}^{SP}(t)) - a_{s,k}^{SP}(t)}{r_{s,k}^{SP}(t)}, \frac{a_j(q)}{r_{k,i}^{PC}(t)}) \leq T_i(j)). \tag{5}$$

In the above equations, s indicates the server from which the proxy is receiving the block j. $q_{s,k}^{SP}(t)$ is the quality of block being received at t. $a_{s,k}^{SP}(t)$ stands for the amount that has been received. $d_{s,k}^{SP}(t)$ and $r_{s,k}^{SP}(t)$ correspond to estimates of available bandwidth and one-way propagation delay between the server s and the proxy k at t.

A proxy keeps track of requests sent out for block retrieval. $\mathcal{W}_{s,k}^{SP}(t) = \{W_1, W_2, \ldots, W_w\}$ is a list of requests sent from the proxy k to the server s before t. The quality of the block W_n, $q_{s,k}^{w}(n)$, is also maintained in a list. A pair of the block number and the quality is added to the lists when the proxy sends a request for block retrieval and is removed from the lists when the proxy begins receiving the block. When the mth request is for the block j, that is, $W_m = j$, the offerable quality $w_{k,i}^{PC}(j)$ becomes

$$w_{k,i}^{PC}(j) = \max_{\forall s, s \neq k} (\min(q_{s,k}^{w}(m), \bar{w}_{k,i}^{PC}(j))), \tag{6}$$

where

$$\bar{w}_{k,i}^{PC}(j) = \max(q | t + 2d_{s,k}^{SP}(t) + d_{k,i}^{PC}(t) + \frac{\sum_{n=1}^{m-1} a_{W_n}(q_{s,k}^{w}(n)) + a_{p_{s,k}(t)}(q_{s,k}^{p}(t))}{r_{s,k}^{SP}(t)} + \max(\frac{a_j(q_{s,k}^{w}(m))}{r_{s,k}^{SP}(t)}, \frac{a_j(q)}{r_{k,i}^{PC}(t)}) \leq T_i(j)). \tag{7}$$

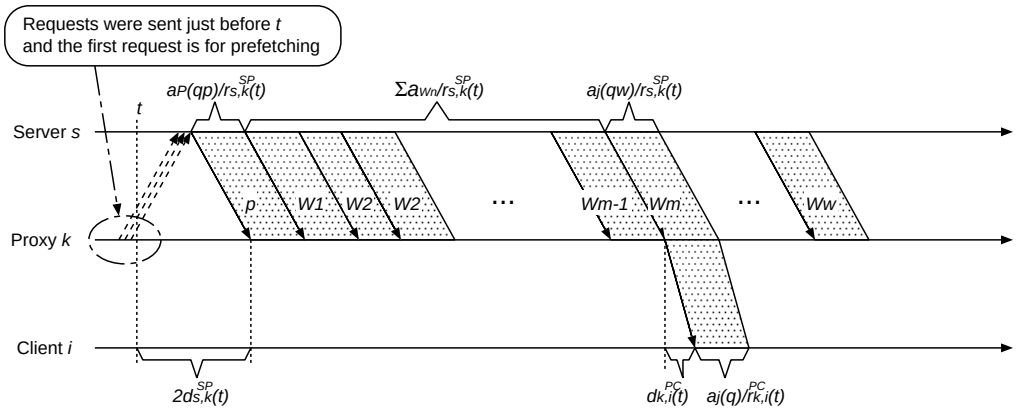

Figure 3. Worst case delay introduced in waiting for the preceding request

$p_{s,k}(t)$ indicates the block number to be prefetched from the server s and $q^p_{s,k}(t)$ is the quality requested for the block $p_{s,k}(t)$. If no prefetching request is waiting, both $p_{s,k}(t)$ and $q^p_{s,k}(t)$ are zero. In our system, only one prefetching request is permitted by a server per proxy and prefetching is preempted by normal block retrievals. Details of a prefetching mechanism will be given in subsection 2.3. Equation 7 considers the worst case when no block is under transmission and the preceding requests from 1 to $m-1$ and the pferetching request will be served in prior to the block W_m, as illustrated in Fig. 3. If the proxy is receiving a block at t, $\bar{w}^{PC}_{k,i}(j)$ is derived using the following equations.

$$\bar{w}^{PC}_{k,i}(j) = \max(q|t+\max(2d^{SP}_{s,k}(t), \frac{a_j(q^{SP}_{s,k}(t)) - a^{SP}_{s,k}(t)}{r^{SP}_{s,k}(t)}) + d^{PC}_{k,i}(t) + \frac{S^{SP}_{s,k}(t)}{r^{SP}_{s,k}(t)} + \max(\frac{a_j(q^w_{s,k}(m))}{r^{SP}_{s,k}(t)}, \frac{a_j(q)}{r^{PC}_{k,i}(t)}) \leq T_i(j)), \tag{8}$$

where

$$S^{SP}_{s,k}(t) = \begin{cases} \sum_{n=1}^{m-1} a_{W_n}(q^w_{s,k}(n)) + a_{p_{s,k}(t)}(q^p_{s,k}(t)), & \text{if } 2d^{SP}_{s,k}(t) > \frac{a_j(q^{SP}_{s,k}(t)) - a^{SP}_{s,k}(t)}{r^{SP}_{s,k}(t)} \\ \sum_{n=1}^{m-1} a_{W_n}(q^w_{s,k}(n)), & \text{otherwise} \end{cases} \tag{9}$$

Finally, the offerable quality $s^{PC}_{k,i}(j)$ by retrieving the block j from the server s is derived by the following equations when there is no block under transmission,

$$s^{PC}_{k,i}(j) = \max_{\forall s, s \neq k} (\min(q^r_s(j), \bar{s}^{PC}_{k,i}(j))), \tag{10}$$

where

$$\bar{s}^{PC}_{k,i}(j) = \max(q|t + 2d^{SP}_{s,k}(t) + d^{PC}_{k,i}(t) + \frac{\sum_{n=1}^{w} a_{W_n}(q^w_{s,k}(n)) + a_{p_{s,k}(t)}(q^p_{s,k}(t))}{r^{SP}_{s,k}(t)} + \frac{a_j(q)}{\min(r^{SP}_{s,k}(t), r^{PC}_{k,i}(t))}) \leq T_i(j)). \tag{11}$$

Here, $q^r_s(j)$ corresponds the quality of the block j cached at the server s. The proxy has information about blocks kept at the originating server and the other proxy servers in *remote table*s. Each table corresponds to a server s from which the proxy can retrieve blocks. Proxies communicates with each other and exchange information about cached blocks. On the contrary, if the proxy k is receiving a block from the server s,

$$\bar{s}^{PC}_{k,i}(j) = \max(q|t + \max(2d^{SP}_{s,k}(t), \frac{a_j(q^{SP}_{s,k}(t)) - a^{SP}_{s,k}(t)}{r^{SP}_{s,k}(t)}) + d^{PC}_{k,i}(t) + \frac{U^{SP}_{s,k}(t)}{r^{SP}_{s,k}(t)} + \frac{a_j(q)}{\min(r^{SP}_{s,k}(t), r^{PC}_{k,i}(t))}) \leq T_i(j)), \tag{12}$$

where

$$U^{SP}_{s,k}(t) = \begin{cases} \sum_{n=1}^{w} a_{W_n}(q^w_{s,k}(n)) + a_{p_{s,k}(t)}(q^p_{s,k}(t)), & \text{if } 2d^{SP}_{s,k}(t) > \frac{a_j(q^{SP}_{s,k}(t)) - a^{SP}_{s,k}(t)}{r^{SP}_{s,k}(t)} \\ \sum_{n=1}^{w} a_{W_n}(q^w_{s,k}(n)), & \text{otherwise} \end{cases} . \tag{13}$$

The fastest and finest way is chosen as far as the offerable quality is higher than the minimum request $q_i(j)$ and is lower than the maximum requeset $Q_i(j)$. If none of $c^{PC}_{k,i}(j)$, $b^{PC}_{k,i}(j)$, $w^{PC}_{k,i}(j)$, and $s^{PC}_{k,i}(j)$ cannot satisfy the request $q_i(j)$, the proxy chooses the fastest way.

2.3. Block Prefetching Algorithm

In order to achieve further effective control, a proxy prefetches blocks in accordance with client requests. While supplying the client i with the block j, the proxy investigates the cache table for blocks $j+1$ to $j+P$ to find a potential cache miss. The parameter P determines the range of the prefetching window. When the proxy finds the block whose quality is lower than $q_i(j)$ and there is no request waiting to be served, it attempts to prefetch the block of finer quality. If there are two or more unsatisfactory blocks, one closest to the block j is chosen.

The prefetch requests are treated at the server in a different way from those for retrieving cache-missed blocks. The origin server and proxy servers maintain pairs of prioritized queues per RTSP session. The queue given a higher priority is for usual block retrieval and requests are handled in a first-come-first-served discipline. The other queue for prefetching has only one room and the request waiting in the queue is overwritten with a new one. The prefetch request in the queue is served only when there is no request in the high-priority queue. The prefetch request is considered obsolete and canceled when the server receives the request for the same block with the higher quality requirement.

The proxy decides to prefetch a block if reception of the block is expected to be completed in time. The expected time $t^p_{s,k}(t)$ when the proxy finishes prefetching is derived as

$$t^p_{s,k}(t) = t + 2d^{SP}_{s,k}(t) + \frac{a_{p_{s,k}(t)}(q^p_{s,k}(t))}{r^{SP}_{s,k}}, \tag{14}$$

where $p_{s,k}(t)$ and $q^p_{s,k}(t)$ stand for the block number and the requested quality, respectively. This equation means that the proxy tries prefetching only when no preceding request is waiting to be served. If the derived time $t^p_{s,k}(t)$ is faster than the time $T^p_i(p_{s,k}(t))$ which is derived as,

$$T^p_i(p_{s,k}(t)) = T_i + \Delta_i + (p_{s,k}(t)-1)\frac{B}{F} - \delta_i + D_i(j-1) - d^{PC}_{k,i}(t), \tag{15}$$

the proxy sends a request to the server s.

Information about a prefetching request is kept at a proxy as a pair of the block number $p_{s,k}(t)$ and the quality $q^p_{s,k}(t)$ and is overwritten by a new prefetching and canceled when a corresponding block reception begins or a normal block retrieval is requested for the same block of higher quality. The quality $q^p_{s,k}(t)$ is determined on the basis of the QoS requirement as $\beta Q_i(j)$ where $0 < \beta \leq 1$. If we set β to a small value, we can expect to prepare enough number of blocks but the quality of blocks provided to clients becomes low. On the other hand, with a large β, there is little chance to successfully prefetch blocks but a high-quality video stream can be provided with prefetched blocks.

2.4. Cache Replacement Algorithm

After the proxy has received the jth block, the block is placed in the proxy's cache buffer if a copy of the block is not already in the cache, or replaces the cached copy of block j in coarser form which is already in the cache. Since a block of higher quality requires more space and the cache buffer is of limited size, the proxy sometimes discards cached blocks and stores the new block instead.

The proxy makes a list of blocks to be discarded on the basis of its cache table. Those blocks which are located in prefetching windows are placed beyond the the scope of replacement. Retention of the first P blocks is also considered important as a way of hiding the initial delay. This was discussed by Sen et al..[13] The rest of the blocks are all candidates for replacement. Among the others, the block closest to the end of the longest run, a succession of non-prioritized blocks, becomes the first 'victim'. Let m be its identifier. If the victim is cached, the proxy first tries degrading its quality to shrink the block size and make a room for the new block. The quality adjustment is limited to the maximum among the QoS requests issued by the clients watching blocks

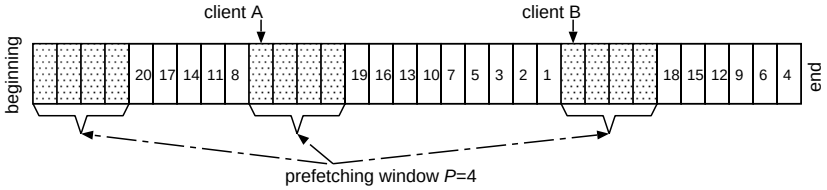

Figure 4. Sample order of block replacement

behind the victim. If the adjustment is not effective enough, the victim is swept out of the cache buffer. If it still is insufficient, the proxy moves to the next victim. When all the victims are dropped but there is not enough room yet, the proxy gives up storing the new block when all attempts failed. Figure 4 illustrates an example of an order of victims. We have a video stream of 32 blocks and the prefetching window $P = 4$ blocks. The proxy is serving client A and B with a block 10 and 23, respectively. Then, those blocks from 1 to 4, from 10 to 14, and from 23 to 26 are not to be replaced. Among the others the first victim is block 22, since a run that the block belongs to is the longest and the block is located at the end of the run. The fourth victim is chosen from the other run which is located closer to the end of the stream as illustrated.

2.5. Moving Client

A client moves around during a video session. A client is assumed to have a capability of predicting its motion.[14] When the client i detects a future handoff, it notifies the designated proxy k of expected time of handoff and the next cell identifier. Proxies know topology of wireless networks, or at least can identify a proxy that is responsible for a cell. If the next cell is under the control of the other proxy s, the proxy k informs the proxy s of client's movement. The message contains the expected time of handoff, the block number j being provided to the client i, the requested quality $q_i(j)$.

The next proxy s begins to prepare for a new client. To avoid degrading services provided to existing clients, only prefetching mechanism starting with the block j is applied to the client i. In a case that the proxy predicts a future cache miss and sends a prefetching requests, it waits for expected time for prefetching and moves to the next block. Otherwise, the proxy moves to the next block after the fixed interval of B/F. The preparation is canceled when (i) the client i enters the proxy s's cell, (ii) the client i predicts another movement to the different server's cell, (iii) the expected time of movement has passed.

3. 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. 5. The originating server is behind a wide-area network or the Internet. It communicates with three proxy servers through 2-Mbps sessions which are established over narrow and long-haul links of propagation delay 200 msec. The proxy servers are located in an ISP network and are connected with each other via 5-Mbps sessions which are established over inter-network broadband lines. Initially, there are ten clients numbered from 1 to 10, from 11 to 20, and from 21 to 30 in the local network of each proxy server. The proxy server establishes an 8-Mbps fixed-bandwidth session with each of its clients. The one-way delays 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. Clients 1, 11, and 21 first issue `PLAY` messages at randomly determined time. The inter-arrival time of the first `PLAY` messages issued by the clients of each proxy 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 $B = 30$. Ten levels of quality are available and the respective block sizes are given by $\forall j \; a_j(q) = \frac{qB}{F} \times 10^6$ bits. Thus, a whole video stream amounts to 5.4 Gbits to 54 Gbits. Each proxy is equipped with a limited cache buffer of 15 Gbits. Initially, all cache buffer are empty and only the originating video server has all blocks of the highest quality. The prefetching window size is $P = 10$ blocks. The buffering

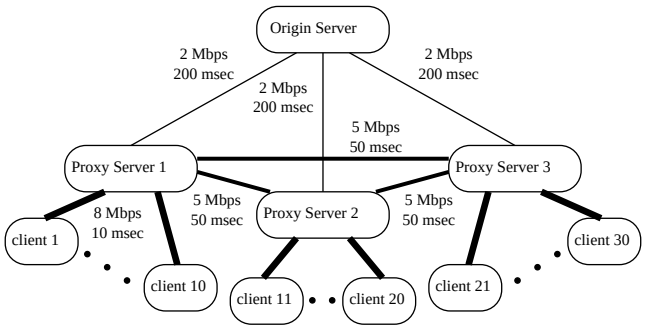

Figure 5. The model used in simulation

times Δ_i and δ_i to absorb delay jitters and prediction errors are set to 4 seconds and 2 seconds, respectively. The tolerable video quality $q_i(j)$ is fixed at 1, since it is the most important to keep playing a video throughout a video session. On the contrary, the preferable quality $Q_i(j)$ varies from 1 to 10. It is initially set to 5, that is, $Q_i(1) = 5$. Then, the client randomly determines the quality requirement on a block-by-block basis. In the experiments, a probability of 5% was assigned to the event of the client increasing $Q_i(j)$ by one. The event of the client decreasing $Q_i(j)$ by one was assigned the same probability, while retaining the same value for $Q_i(j)$ was assigned a probability of 90%. This is introduced as a way of imitating dynamic changes in quality requirements according to system conditions and the user's preference. The ratio of the quality of prefetching block to the request is determined as $\beta = 0.6$. The residence time in a proxy's wireless domain follows an exponential distribution whose average is 10 minutes. The client is assumed to predict a handoff 30 seconds before the movement on average. The next proxy is determined at random.

Figures 6 and 7 depict results of preliminary simulation experiments. For comparison purpose, we conducted simulation of a system where proxies independently operate (referred to as "independent"). We compare the performacen in terms of the average freeze time and the average quality ratio. The average freeze time is derived as $\sum_{j=1}^{L} f_i(j)/L$. The quality ratio is defined as the ratio of quality of provided block to the request $Q_i(j)$. Figure 6 implies that clients of proxy 3 first subscribed to the service and began to watch a video. Thus, clients from 21 to 24 in the independent system were forced to pause and wait for arrivals of blocks. With our system, the average freeze time is improved very much. Clients 2, 11, 12, and 24 become free from waiting for delivery of blocks and the average freeze time of clients 1, 21, 22, and 23 are reduced by 88–56%. In addition, Fig. 7 shows that the quality of blocks provided to clients are improved for all clients. In particular, clients with a larger number, that is, clients subscribed the service later received blocks whose quality ratios are as high as 0.8. This is because that proxies provide those clients with cached blocks and prefetch blocks of higher quality in advance using residual bandwidth. As these figures clearly show, our proposed mechanisms can provide users with continuous and higher-quality video streaming services in heterogeneous and mobile environments.

4. CONCLUSION

In this paper, we propose a new and effective video streaming mechanism where proxies retrieve, prefetch, adjust, cache, and deliver video blocks in a cooperative way in heterogeneous and mobile environments. The proposed mechanism can be applied to any type of video streaming service as long as the block-based transfer and the quality adjustment are possible. However, we put some assumptions on the system and only preliminary experiments have been conducted. We are now considering the implementation to verify the applicability and practicality of our proposal.

ACKNOWLEDGMENTS

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.

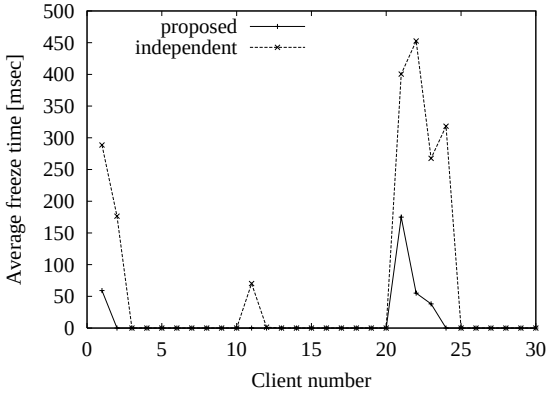

Figure 6. Average freeze time

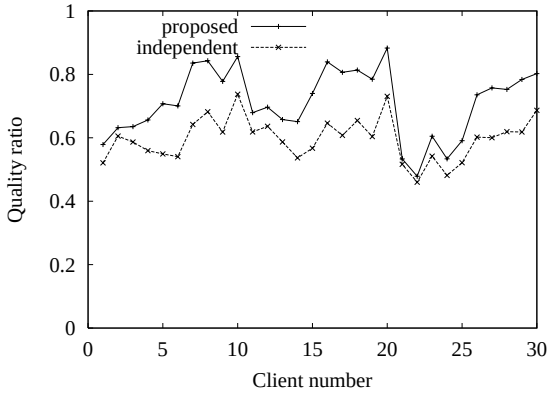

Figure 7. Average quality ratio

REFERENCES

1. M. Andrews and K. Munagala, "Online algorithms for caching multimedia streams," in *Proceedings of European Symposium on Algorithms*, pp. 64–75, September 2000.
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. 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*, **II**, pp. 588–591, February 2000.
4. R. Rejaie and J. Kangasharju, "Mocha: A quality adaptive multimedia proxy cache for internet streaming," in *Proceedings of NOSSDAV'01*, March 2001.
5. K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in *Proceedings of The Tenth International World Wide Web Conference*, pp. 36–44, May 2001.
6. G. Barish and K. Obraczka, "World Wide Web caching: Trends and techniques," *IEEE Communications Magazine* , pp. 178–185, May 2000.
7. 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*, **4519**, pp. 276–284, August 2001.
8. N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, "Filters: QoS support mechanisms for multipeer communications," *IEEE Journal on Selected Areas in Communications* **14**, pp. 1245–1262, September 1996.
9. N. Wakamiya, K. Fukuda, M. Murata, and H. Miyahara, "On multicast transfer of hierarchically coded video with QoS guarantees," in *Proceedings of Internet Workshop'99*, pp. 294–300, February 1999.
10. 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.
11. A. M. Dawood and M. Ghanbari, "Content-based MPEG video traffic modeling," *IEEE Transactions on Multimedia* **1**, pp. 77–87, March 1999.
12. 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.
13. S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of IEEE INFOCOM'99*, **3**, pp. 1310–1319, March 1999.
14. T. Liu, P. Bahl, and I. Chlamtac, "Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks," *IEEE Journal on Selected Areas in Communications* **16**, pp. 922–936, August 1998.