Adaptive and Robust P2P Media Streaming

MASAHIRO SASABE[†] NAOKI WAKAMIYA[§]
MASAYUKI MURATA[§]
Cybermedia Center
Osaka University [†]
1-32 Machikaneyama, Toyonaka, Osaka 560-0043
JAPAN

Graduate School of Information Science and Technology
Osaka University [§]
1-5 Yamadaoka, Suita, Osaka 565-0871
JAPAN
m-sasabe@cmc.osaka-u.ac.jp http://www.ane.cmc.osaka-u.ac.jp/˜m-sasabe/

Abstract: - With the growth of computing power and the proliferation of broadband Internet access, the use of media streaming has become widely diffused. In this paper, based on our previous work, we propose a Peer-to-Peer (P2P) media streaming system that can provide a large number of users with continuous media streaming services while dynamically adapting to peer departures and changes in network conditions. For this purpose, we propose a new media retrieval method that dynamically switches provider peers. Through several simulation experiments, we show that the proposed media retrieval method improves the completeness of media playout compared with previous methods. Furthermore, we also show that our search method is more robust to peer departures than flooding methods.

Key-Words: - P2P, media streaming, streaming caching, continuity, adaptability, robustness

1 Introduction

With the growth of computing power and the proliferation of such broadband Internet access as Asymmetric Digital Subscriber Line (ADSL) and Fiber To The Home (FTTH), the use of media streaming has become widely diffused. A user receives a media stream from an original media server through the Internet and plays it on his/her client system as it progressively arrives.

Because client-server architecture lacks scalability and adaptability, there have been several research works on media streaming with P2P networks [1, 2, 3, 4]. Most of these were designed for live broadcasts by constructing an application-level multicast tree in which an original media server is the root and peers are intermediate nodes and leaves [1, 2, 3]. A media stream emitted from a server is distributed over a tree. Each intermediate peer receives media data from its parent peer, makes copies of data, and forwards them to its child peers. Thus, they are effective when user demands are simultaneous and concentrated on a specific media stream. However, when demands arise intermittently and peers request a variety of media streams, as in on-demand media streaming services, an efficient distribution tree cannot be constructed. On the other hand, such works as PROMISE [4] are similar to our proposed system where a peer finds a provider peer for the entire or a part of a desired media stream, retrieves it from the provider, and deposits it in a local buffer to supply to other peers. However, those works failed to address variations in the popularity among multiple media streams. For example, LRU, the most common caching strategy, proved to sweep away unpopular media streams from a P2P network.

In [5], we proposed a novel system for on-demand media streaming in pure P2P networks. In our system, a media stream is divided into blocks that efficiently use network bandwidth and cache buffers. By retrieving blocks from appropriate provider peers in time, a peer can watch a desired media stream. If there is no room to store the newly retrieved block, a peer performs replacement on cached blocks while considering the balance between the supply and demand of media streams. We also proposed methods for scalable block search, in-time block retrieval, and efficient cache replacement. Our proposed system provides users with continuous media streaming on stable and static P2P networks.

However, in actual situations, media streaming fails since peers participating in a service occasionally leave or depart from a P2P network due to user interactions or system failures. In addition, network conditions including available bandwidth and Round Trip Time (RTT) dynamically change. In this paper, we propose a new block retrieval method that dynamically switches provider peers based on the estimation of available bandwidth and RTT. In addition, we extend our block search method to prepare for peer departures. Through several simulation experiments, we evaluate our proposed methods for completeness of media playout under unstable system conditions.

The rest of the paper is organized as follows. In section 2, we give an overview of our streaming system on pure P2P networks, propose a new adaptive retrieval method, and extend our block search method. In section 3, we evaluate our proposed methods through several simulation experiments. Finally, we conclude the paper and describe future works in section 4.

2 Adaptive and Robust P2P Media Streaming

2.1 Overview of Proposed System

Peers participating in our system first join a logical P2P network for media streaming. For efficient use of network bandwidth and cache buffers, a media stream is divided into blocks. In our system, a peer retrieves a media stream and plays it in a block-by-block manner. To avoid flooding a network with query messages, our method employs a per-group search scheme that considers the temporal order of references in a media stream. A peer sends out a query message for every N consecutive blocks called a round. Figure 1 illustrates an example of $N = 4$. P_A, P_B, P_C, and P_D indicate peers within the range of the propagation of query messages. The numbers in parentheses next to peers stand for the identifiers of a peer's blocks. At time $T_s(1)$, a query message for blocks 1 to 4 is sent out from P to the closest

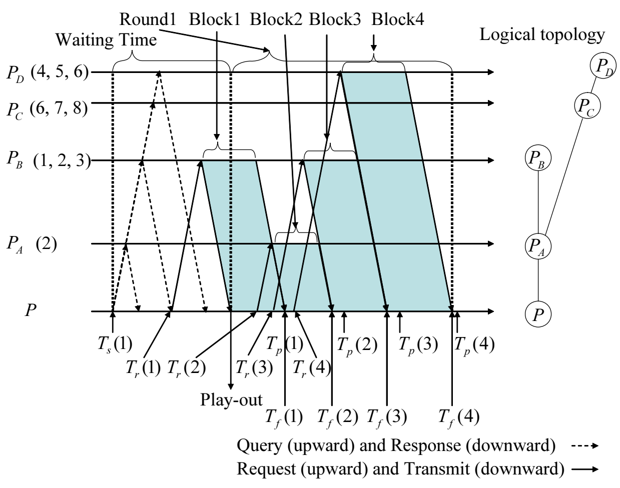

Fig. 1: Example of per-group search and retrieval.

peer P_A. The query is relayed among peers, that is, *flooding*. Since P_A, P_B, and P_D have one or more blocks out of the four requested blocks, they return response messages.

From search results obtained by the query, P determines a provider peer for each block in the round from which a block can be retrieved in time. For this purpose, a peer keeps estimating the available bandwidth and RTT to other peers while exchanging query and response messages. For example, the inline network measurement technique [6] can be used for such estimations without introducing extra load on a network. Then, a peer retrieves blocks from provider peers as illustrated in Fig.1.

If there is no room to deposit a newly retrieved block in its cache buffer, P determines a media stream in which cached blocks are replaced with new blocks, considering the balance between the supply and demand for media streams in the network. The probability that media i is chosen as a replacement victim is given as:

$$P_r(i) \quad = \quad \frac{s^2(i)}{s^2(i) + \theta^2(i) + l^2(i)}, \qquad (1)$$

where $s(i)$ indicates the ratio of supply $S(i)$ to demand $D(i)$ for media stream i after the replacement and is derived as $\max\left(\frac{S(i)-1}{D(i)}, 0\right)$. $S(i)$ and $D(i)$ are conjectured from query and response messages sent by the peer, received, and relayed. $\theta(i)$ is the response threshold of media stream i. $l(i)$ is the ratio of the number of locally cached blocks to the number of blocks in media stream i.

To accomplish continuous media playout, P sends a query for the next round at a time $2RTT_{worst}$ earlier than the starting time of the next round. RTT_{worst} is the RTT to the most distant peer among the peers that returned response messages in the current round. The range of the next search is based on the search results of the current round. If all blocks in the next round are expected to be cached in peers that P knows, it selectively sends query messages to those peers (selective search). If not all, but some blocks are assumed to exist, it adopts a strategy that floods query messages with a limited TTL value (limited flooding).

In [5], our system could provide users with continuous media playout under stable system conditions, when all peers stayed in the network and either the available bandwidth or the RTT did not change. However, in actual situations, system conditions dynamically change. In-time block retrieval fails if the available bandwidth decreases due to congestion. A peer cannot find a provider peer since search results become unreliable due to peer failures or departures. In the following subsections, we propose a block retrieval method that adapts to system changes and extend our block search method to be robust to peer departures.

2.2 Block Retrieval Method

First we introduce our in-time block retrieval method [5] and then newly propose a block retrieval method adaptive to system changes.

2.2.1 In-time Block Retrieval Method

The peer sends a request message for the first block of a media stream immediately after receiving a response message from a peer that has the block, because it cannot predict whether any better peer exists at that time. In addition, it is essential for low delay and effective media streaming service to begin media presentation as quickly as possible. Thus, in our method, the peer immediately plays out the first block when its reception starts. Of course, we can also defer the playout to buffer a certain number of blocks in preparation for unexpected delays.

The deadlines for the retrieval of succeeding blocks $j \geq 2$ are determined as follows:

$$T_p(j) = T_p(1) + (j-1)B_t, \qquad (2)$$

where $T_p(1)$ corresponds to the time that the peer finishes playing out the first block.

Although block retrieval should follow a playout order, the order of request messages does not. We don't wait for the completion of the reception of the preceding block before issuing a request for the next block because this introduces an extra delay of at least one round trip. Cumulative delay affects the timeliness and completeness of media playout. Instead, the peer sends a request message for block j at $T_r(j)$ so that it can start receiving block j immediately after finishing the retrieval of blocks $j-1$, as shown in Fig. 1. As a result, our block retrieval method can maintain the completeness of media playout.

The peer estimates the available bandwidth and the transfer delay from the provider peer by using existing measurement tools. For example, by using the inline network measurement technique [6], those estimates can be obtained by exchanging query and response messages without introducing any measurement traffic. Furthermore, the estimates are updated by the reception of media data. Our estimation-based control is effective for dynamically changing P2P networks since the frequency of peer departures/joins is relatively small in media streaming services. In addition, the influence of incorrect estimations is limited to a block. Every time the peer receives a response message, it derives the estimated completion time of the retrieval of block j, that is $T_f(j)$, from the block size and the estimated bandwidth and delay for each block to which it has not yet sent a request message. Then it determines an appropriate peer in accordance with deadline $T_p(j)$ and calculates time $T_r(j)$ at which it sends a request.

In the provider determination algorithm, the peer calculates set S_j, a set of peers having block j. Next, based on the estimation of available bandwidth and transfer delay, it derives set $S_j^{'}$, a set of peers from which it can retrieve block j by deadline $T_p(j)$, from S_j. If $S_j^{'} = \phi$, then the peer waits for the arrival of the next response message. However, it abandons retrieval and playing block j when deadline $T_p(j)$ passes without finding an appropriate peer. To achieve continuous media playout, block retrieval time must be shortened. The Select Fastest (SF) method selects a peer whose estimated completion time is the smallest among those in S'_j. By retrieving block j as quickly as possible, the remainder of $T_p(j) - T_f(j)$ can be used to retrieve the succeeding blocks from distant peers. On the other hand, an unexpected cache miss introduces extra delay in the client system. The Select Reliable (SR) method selects the peer with the lowest possibility of block disappearance among those in S'_j. As a result, this suppresses block disappearance before a request for block j arrives at the provider peer. In simulation experiments for this paper, we employed the SF method.

A peer emits a request message for block j to provider peer $P(j)$ at $T_r(j)$. On receiving the request, $P(j)$ initiates block transmission. If it replaced block j with another block since it returned a response message, it informs the peer of a cache miss. When a cache miss occurs, the peer determines another provider peer based on the above algorithm. However, if it has already requested a block after j, it gives up retrieving block j to maintain media playout.

After receiving block j, the peer replaces $T_f(j)$ with the actual completion time. In the algorithm, the estimated completion time of the retrieval of block j depends on block $j-1$. Therefore, if the actual completion time $T_f(j)$ of the retrieval of block j changes because of changes in network conditions or estimation errors, the peer reapplies the algorithm and determines provider peers for succeeding blocks.

2.2.2 Adaptive Block Retrieval Method

In actual situations, system conditions dynamically change. In-time block retrieval fails if the available bandwidth decreases due to congestion. A peer cannot find a provider peer since search results becomes unreliable due to peer failures or departures. To tackle this problem, we propose an adaptive block retrieval method that appropriately switches provider peers based on the estimated available bandwidth and RTT. As long as the following condition holds, a peer can retrieve block j in time.

$$\frac{V(t) + r(j,t)}{\frac{B_s}{B_t}} \geq \frac{r(j,t)}{\Delta_{bw} A(i,t)}, \tag{3}$$

where $A(i,t)$ is the available bandwidth from peer i at time t, $V(t)$ is the remaining buffer size at t, and $r(j,t)$ is the remaining size of block j being retrieved at t. B_s and B_t are block size and time, respectively. Δ_{bw} considers the degree bandwidth estimation accuracy. When peer i leaves a P2P network, the available bandwidth is considered zero.

When Eq.(3) cannot be satisfied, the peer tries to find an alternative provider peer i' from which it can retrieve the remainder of block j in time. Peer i' must satisfy the following condition:

$$\frac{r(j,t)}{A(i,t)} \geq \min_{i' \in S_j - i} \left(\frac{r(j,t)}{A(i',t)} + 2R(i',t) \right), \tag{4}$$

where S_j is a set of provider peers of block j obtained from the search results and $R(i',t)$ is the RTT from peer i' at time t. If the peer can find i', then it retrieves the remainder of block j from i'. Otherwise, it abandons retrieving block j.

2.3 Robust Block Search Method

In [5], a peer conducts a selective search in the next round when it finds one provider peer for each block. It then directly sends a query message to each provider peer. Thus, if any provider peers leaves the P2P network, a peer loses an opportunity to find and retrieve the corresponding block. In the robust block search method, we introduce parameter x, which defines the number of provider peers to be found in the current round to move to the selective search in the next round. For example, by setting x to two, a peer moves to the selective search when it finds two provider peers and can prepare an alternative provider peer.

3 Simulation Experiments

We conducted simulation experiments to evaluate the proposed methods in terms of the completeness of media playout. We define completeness as the ratio of the number of retrieved blocks in time to the number of blocks in a media stream.

3.1 Simulation Model

We used a P2P logical network with 100 peers randomly generated by the Waxman algorithm with parameters $\alpha = 0.15$ and $\beta = 0.3$.

Forty media streams 60 minutes long were available. The media coding rate was set to CBR 500 kbps. A media stream was divided into blocks of 10 sec. Media streams were numbered from 1 to 40 (most to least popular), where the various levels of popularity followed a Zipf-like distribution with $\alpha = 1.0$. Therefore, media stream 1 was forty times more popular than media stream 40.

At first, none of the 100 peers watched any media stream. Then, peers randomly began to request a media stream one by one. The interarrival time between two successive media requests for the first media stream among clients followed an exponential distribution that averaged 20 minutes. Each peer watched a media stream without such interactions as rewinding, pausing, or fast-forwarding. Each peer sent a query message for a succession of six blocks, i.e., $N = 6$. Blocks obtained were deposited into a cache buffer of three media streams. When a peer finished watching a media stream, it became idle during the waiting time, which also followed an exponential distribution that averaged 20 minutes.

At the beginning of each simulation experiment, each peer stored three entire media streams in its cache buffer. The initial population of each media stream in the network also followed the same Zipf-like distribution as the media popularity. To prevent the initial condition of the cache buffer from influencing the system's performance, we only used the results after the initially cached blocks were completely replaced with newly retrieved blocks. We show the average values of twenty sets of simulations in the following figures.

3.2 Changes in Network Conditions

We randomly changed the RTT from 10 to 660 ms and the available bandwidth from 450 and 550 kbps between two arbitrary peers at one-second intervals. A peer estimated the available bandwidth and RTT at one-second intervals with a degree of accuracy of 0.975. We set Δ_{bw} to 0.95, which was less than the estimation accuracy. Since in this scenario peers did not leave, we set x to 1.

Figure 2 depicts the completeness of the previous and the proposed methods. For comparison purposes, we show the results in a stable environment where the available bandwidth constantly exceeds the media coding rate. As shown in this figure, the completeness of the previous method is less than 0.5, even for the most popular media stream under unstable network conditions. This

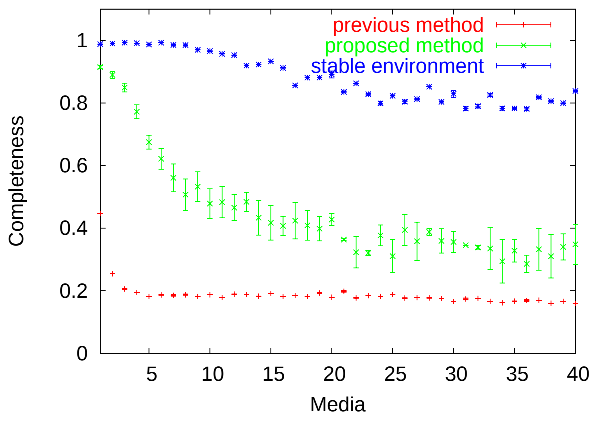

Fig. 2: Completeness (40 streams)

is because peers keep retrieving blocks even when the available bandwidth decreases and they cannot finish the retrieval in time. In contrast, the proposed method improves completeness by appropriately changing provider peers. However, it is still lower than in a stable environment. For a peer effectively to switch provider peers, it needs at least one alternative peer i' that satisfies Eq.(5).

$$\left\{ i' \in S_j - i \,\middle|\, \frac{V(t) + r(j,t)}{\frac{B_s}{B_t}} \geq \frac{r(j,t)}{A(i',t)} + 2R(i',t) \right\} \quad (5)$$

The number of alternative peers depends on the number of media streams against the capacity of the entire P2P network. Figure 3 illustrates the results of a case with ten media streams. As shown in the figure, the completeness of the proposed method is more than 0.9 independent of media popularity. Although not shown in the figures, we conducted additional experiments with various numbers of media streams and found that completeness of more than 0.9 could be attained for all media popularity when the number of media streams was less than 12 against network capacity of $100 \times 3 = 300$ media streams.

3.3 Departures of Provider Peers

To evaluate the robustness to peer departures, we next considered the following scenario. First, one designated peer was randomly chosen. Then, we removed peers while the designated peer was retrieving the first media stream. Peers to be removed were randomly chosen among peers that

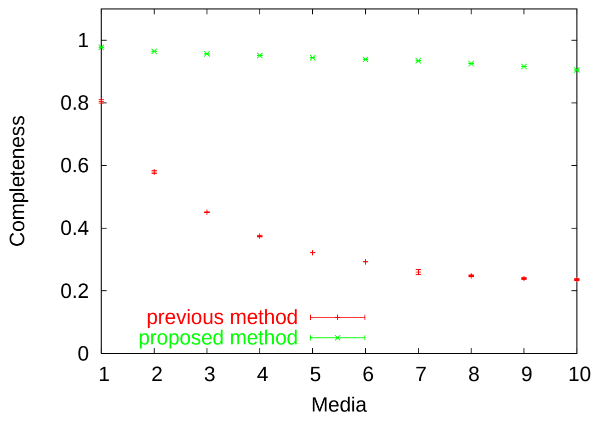

Fig. 3: Completeness (10 streams)

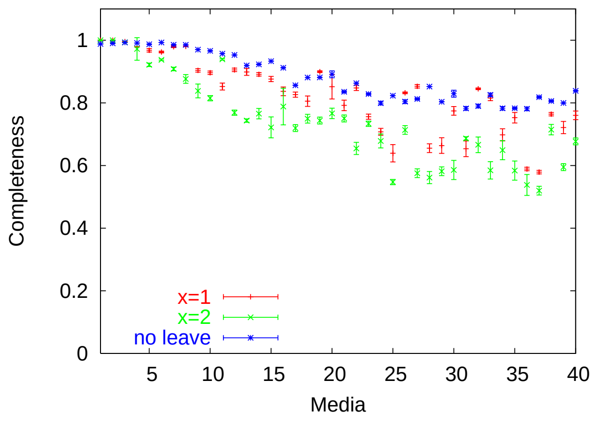

Fig. 4: Completeness with peer departures

deposited blocks of the first media stream of the designated peer. The interarrival time between two successive removals followed an exponential distribution that averaged ten minutes. To investigate the impact of peer departures, there was no reconstruction or recovery of the P2P network when peers left and links were broken. In this scenario the available bandwidth and RTT did not change.

Figure 4 shows that the completeness of $x = 1$ is higher than $x = 2$, contrary to our expectations. With $x = 2$, a peer conducts limited flooding more often than cases with $x = 1$. In limited flooding, query messages are diffused over a P2P network and relayed by peers. Thus, the possibility that a peer can find an appropriate provider peer decreases due to breaks in the network caused by the disappearance of peers. On the other hand, in selective search, query messages are directly sent to provider peers without mediations from other peers. As a result, even though only 6 % peers left the network in Fig. 4, the completeness of $x = 1$ becomes superior to $x = 2$. Thus, we can conclude that the selective search is more robust to peer departures than the limited flooding from simulation results.

4 Conclusions

In this paper, we proposed an adaptive block retrieval method for continuous media streaming on unstable P2P networks. Through several simulation experiments, we showed that completeness was successfully improved from previous methods. In addition, we found that selective search was more efficient than flooding methods when peers failed and left.

As a future research work, we plan to implement our proposed methods on an actual system to verify practicality and evaluate applicability.

Acknowledgement

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 17700058, 2005.

References

[1] AllCast. Available at `http://www.allcast.com`.

[2] Padmanabhan V.N., Wang H.J., and Chou P.A. Resilient Peer-to-Peer Streaming. *Microsoft Research Technical Report MSR-TR-2003-11*, Mar. 2003.

[3] Tran D.A., Hua K.A., and Do T.T. A Peer-to-Peer Architecture for Media Streaming. *IEEE Journal on Selected Areas in Communications*, vol. 22(1), Jan. 2004, pp. 121–133.

[4] Hefeeda M., Habib A., Botev B., Xu D., and Bhargava B. PROMISE: Peer-to-Peer Media Streaming Using CollectCast. In *Proceedings of ACM Multimedia 2003*. Berkeley, Nov. 2003.

[5] Sasabe M., Wakamiya N., Murata M., and Miyahara H. Effective Methods for Scalable and Continuous Media Streaming on Peer-to-Peer Networks. *European Transactions on Telecommunications*, vol. 15(6), Nov. 2004, pp. 549–558.

[6] Man C., Hasegawa G., and Murata M. Available Bandwidth Measurement via TCP Connection. In *Proceedings of IFIP/IEEE MMNS 2004 E2EMON Workshop*. San Diego, Oct. 2004.