IEICE TRANS. INF. & SYST., VOL.E94–D, NO.12 DECEMBER 2011

PAPER Special Section on Parallel and Distributed Computing and Networking

Adaptive Prefetching Scheme for Peer-to-Peer Video-on-Demand Systems with a Media Server*

Ryusuke UEDERA[†a)], ***Student Member and*** **Satoshi FUJITA**[†], ***Member***

SUMMARY In this paper, we consider Peer-to-Peer Video-on-Demand (P2P VoD) systems based on the BitTorrent file sharing protocol. Since the Rarest First policy adopted in the original BitTorrent protocol frequently fails to collect pieces corresponding to a video file by their playback time, we need to develop a new piece selection rule particularly designed for P2P VoDs. In the proposed scheme, we assume the existence of a media server which can upload any piece upon request, and try to bound the load of such media server with two techniques. The first technique is to estimate pieces which are not held by any peer and prefetch them from the media server. The second technique is to switch the mode of each peer according to the estimated size of the P2P network. The performance of the proposed scheme is evaluated by simulation.

key words: *Peer-to-Peer, video-on-demand, media server, scheduling*

1. Introduction

Video-on-Demand (VoD) is an online service which enables customers to watch their favorite videos at any time, at any place. In this service, each user can acquire a requested video file in such a way that she can start watching while the download is in progress. Most of existing VoD systems such as YouTube and Ustream are implemented under the traditional Client/Server (C/S) model [1], [2], so that they could enjoy several advantages such as an efficient resource management and a secure contents delivery. However, such a centralized approach causes several critical issues such as the access bottleneck and a single point of failure at the central server.

In order to overcome such problems, Peer-to-Peer (P2P) technology has emerged as a perspective way to realize scalable, dependable VoDs. A VoD system based on the P2P technology is generally referred to as a P2P VoD. P2P is a distributed system consisting of a number of autonomous computers called peers, and each peer participating in the system plays the roles of a server and a client at the same time, so that network services will be provided with the aid of many participating peers. Such a cooperative behavior of the participants could reduce the load of the central server, and in many cases, significantly improves the scalability and the dependability of the overall system.

In this paper, we will focus our attention to P2P VoDs of the BitTorrent type. BitTorrent [5] is a P2P file sharing protocol based on: 1) the notions of file chunking, where chunks are called *pieces*, 2) quick propagation of pieces using the Rarest First policy, and 3) an incentive mechanism based on the Tit-for-Tat strategy. BASS [6] is the first attempt to combine the BitTorrent protocol with VoD. In [9], Vlavianos et al. proposed a P2P VoD called BiToS, which equips a modified piece selection rule so as to avoid suspending of a video play while conducting a download (details of those systems will be described in Sect. 3). In general P2P file sharing systems, a peer can leave the system as soon as it finishes the download of an entire file, and it causes a situation in which there remain only few peers to have a copy of the file. In particular, if the number of peers who have downloaded the file is very small, we could not avoid a situation in which a particular piece is not held by any peer. Such a "missing" piece can never be acquired by any peer until another peer who has that piece will join the network. In P2P VoDs, although such a problem of missing pieces could be partially resolved by preparing a media server which stores all video files as in BASS, such a media server easily becomes a bottleneck as increasing the number of missing pieces in the system.

In this paper, we propose a *piece prefetching scheme* to reduce the amount of such missing pieces. Our scheme consists of two basic techniques. The first technique is to estimate a set of missing pieces by referring to the local information around each peer. By limiting the area of references by an appropriate TTL (Time-to-Live), and by limiting the time interval of consecutive references by an appropriate value, we can bound the cost required for each prefetch operation sufficiently low. We can accurately realize such an estimation when the number of peers in the corresponding P2P network is relatively small (as will be described later, we assume that a P2P network is organized for each media file to be downloaded). In our second technique, we try to switch the mode of each peer, i.e., whether or not a prefetching should be executed, by referring to the estimated size of the network.

The performance of the proposed scheme is evaluated by simulation. The result of simulations indicates that: 1) the prefetching scheme certainly reduces the load of the media server when the arrival of peers is bursty, e.g., if the average arrival interval is less than or equal to 1 [s], it re-

Manuscript received December 24, 2010.
Manuscript revised May 25, 2011.

[†]The authors are with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Higashihiroshima-shi, 739–8527 Japan.

*Earlier version of this paper was presented as "Adaptive Prefetching Scheme for Peer-to-Peer Video-on-Demand Systems with a Media Server," by Ryusuke Uedera and Satoshi Fujita, in Proc. the First International Conference on Networking and Computing, Nov. 2010.

a) E-mail: ryusuke@se.hiroshima-u.ac.jp

DOI: 10.1587/transinf.E94.D.2362

duces the load of the media server in a conventional scheme by 39%; 2) the mode switching technique effectively adapts the scheme to the underlying P2P network of various sizes; i.e., it enhances the advantage of prefetching when the network size is small, and it avoids unnecessary overhead in large networks. Result of additional experiments indicates that the accuracy of such an estimation could be improved by increasing the TTLs and the frequency of references.

The remainder of this paper is organized as follows. In Sect. 2, we overview related works. Section 3 gives a detailed description of BitTorrent and previous P2P VoDs. Our proposed scheme is described in Sect. 4. The result of performance evaluation is summarized in Sect. 5. Finally, Sect. 6 concludes the paper with future works.

2. Related Work

Recently, a number of P2P video streaming systems have been proposed in the literature. Those systems can be classified into two types by the configuration of the P2P network, i.e., tree-based systems and mesh-based systems.

In tree-based systems, participant peers organize a tree-structured P2P network called a multicast tree, and a peer located at the root of the tree uploads a copy of a video file in the form of a video stream. Such a stream is delivered to the recipients of the video file through the multicast tree, where each intermediate peer on the delivery paths continuously forwards the stream received from a peer in the upstream to the peers in the downstream. ESM [4] is a tree-based live streaming system which adopts a single multicast tree for each video stream. A critical drawback of tree-based systems such as ESM is that the failure of a peer significantly degrades the routing performance of the overall system. In order to overcome such a problem, SplitStream [3] prepares several multicast trees for each video stream, and uses those trees in a combined manner. P2Cast [7] is another tree-based scheme based on the notion of patching. Patching is a technique which allows each peer to receive video streams having been delivered in the past, by caching those streams in each intermediate peer.

In contrast to tree-based systems, P2P network in mesh-based systems can take any structure, and each link in the network can be used in both directions. Thus, by adopting a graph structure with high connectivity as an underlying P2P network, we could effectively resolve drawbacks of tree-based systems. In mesh-based systems, each file is divided into several chunks, and each peer downloads chunks from adjacent peers, while conducting an upload of other chunks to the adjacent peers. CoolStreaming [10] is a typical mesh-based live streaming system based on that idea, where the mesh structure of the system is organized by executing a gossip protocol.

Recently, BitTorrent file sharing system attracts considerable attention as a key infrastructure to realize an efficient delivery of video streams to their designated recipients. BASS [6] is the first P2P VoD based on that idea. In this system, chunks are called pieces, and can be downloaded from adjacent peers, in addition to the ordinary download from the media server. In [9], Vlavianos et al. improved the piece selection rule in BitTorrent, and designed a P2P VoD called BiToS. A remarkable feature of BiToS is that it does not rely on any media server, while techniques proposed there are also applicable to P2P VoDs with a media server. The piece selection rule in BiToS was further refined by many researchers; e.g., Zhou et al. [11] proposed a mixed strategy consisting of several known rules and Sakashita et al. [8] proposed a rule based on the rarity and the urgency of pieces. The reader should note that those piece selection rules can be used with our proposed scheme in a combined manner, since the objective of our scheme is to reduce the load of a media server rather than the increase of the efficiency of piece exchanges.

3. BitTorrent

In this section, we first describe an overview of the BitTorrent protocol with its two key techniques; i.e., an incentive mechanism and the Rarest First piece selection rule. We then outline BASS and BiToS, which are P2P VoDs based on the BitTorrent protocol.

3.1 Overview

In the BitTorrent protocol, each file is divided into several chunks called pieces, and those pieces are exchanged among peers relevant to that file. A BitTorrent network consists of a tracker and a collection of peers, and the tracker manages the information on all peers in the network, such as peer ID, IP address, and the port number. To start the download of a file, each peer first sends a request to the tracker relevant to the requested file to join the network managed by the tracker. Upon receiving the request, the tracker sends back a list of peers to the requester, so that the requester establishes a connection to several peers relevant to the requested file, where such a request is periodically issued to establish additional connections to the relevant peers. During the download of pieces from adjacent peers, each peer keeps the *piece availability* for each adjacent peer, which denotes whether each piece is held or not by the peer, and uses it to determine the order of pieces to be downloaded (details of the piece ordering scheme are described in the next subsection). In addition, in order to keep such an information as accurate as possible, whenever a piece is newly acquired, each peer informs the fact to all of its adjacent peers.

3.2 Piece Selection Rule

The performance of the BitTorrent protocol is significantly affected by the piece selection rule. A key point to realize an efficient download under the protocol is how to avoid the slowdown due to the *rareness* of the pieces to be downloaded, since a peer to have a rare piece becomes a bottleneck in the overall download process. This motivates a piece selection rule called the Rarest First policy, where the word

"rarest" indicates that the number of adjacent peers holding that piece is the smallest. Such a set of rarest pieces can be easily identified by using the piece availability information locally kept by each peer. Note that this policy is effective not only to avoid the slowdown of a download, but also to protect rare pieces from being extinct due to an unexpected failure or a sudden leave of peers.

3.3 Incentive Mechanism

Another key issue we need to consider is how to motivate each peer to upload its acquired pieces to the other peers. In other words, we should prevent each peer from being selfishly downloading pieces without contributing to the system as an uploader. In order to regulate such a cooperative behavior of the peers, the BitTorrent protocol uses a choke/unchoke mechanism in the following manner (note that we will merely describe a typical scheme based on the Tit-for-Tat strategy, although several alternative schemes are prepared in the BitTorrent protocol): In the protocol, at any point in time, the number of target peers to which acquired pieces can be uploaded is bounded by a constant for each peer, where the status of each connection is controlled by using choke/unchoke operations. The amount of contributions as an uploader is evaluated by the upload rate, i.e., by the amount of uploads per second, and each peer determines the set of target (i.e., unchoked) peers in a non-increasing order of the upload rate of the adjacent peers.

3.4 BASS

BASS is the first P2P VoD based on the BitTorrent protocol. System architecture of BASS is illustrated in Fig. 1. In addition to the basic infrastructure used in the original BitTorrent, it uses a media server which stores all video files to be uploaded. Each peer downloads pieces from the media server in a playback order, while conducting a download from adjacent peers according to the Rarest First policy, where the policy is slightly modified in such a way that the pieces prior to the current playback position will never be selected. During the download from the media server, it skips the download of pieces which have already been acquired from the adjacent peers, or pieces which are expected to be acquired before their playback time.

3.5 BiToS

BiToS is another P2P VoD developed by Vlavianos et al. [9]. Figure 2 illustrates the system architecture of BiToS. In P2P VoDs based on the BitTorrent protocol, each peer should collect pieces close to its playback position as quickly as possible to avoid unnecessary suspending of a video play. In other words, each piece is given a deadline, and the criticalness of such deadline gradually and dynamically changes after starting the play of the video. However, the Rarest First policy adopted in the BitTorrent protocol does not explicitly take into account such a deadline, but merely tries to collect

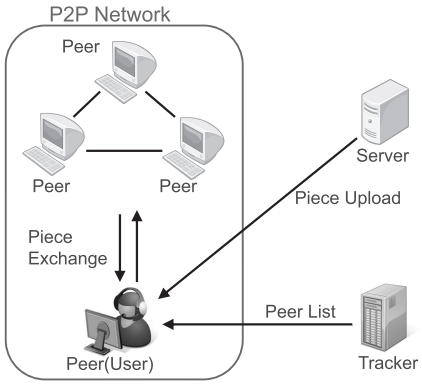


Fig. 1 System architecture of BASS.

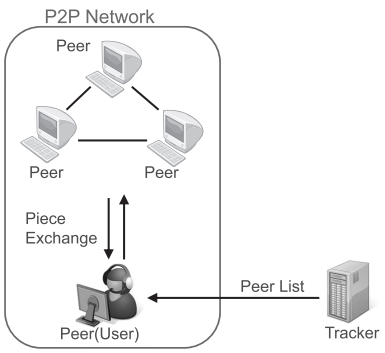


Fig. 2 System architecture of BiToS.

rare pieces independent of the position in the given media file. This means that it is difficult to collect all pieces before deadline under the piece selection rule adopted in the original BitTorrent.

To overcome such a problem, BiToS adopts the following modified piece selection rule: At first, each peer partitions the set of unacquired pieces into two subsets, i.e., a high priority set consisting of a limited number of high priority pieces close to the deadline, and the set of remaining pieces. After selecting one of those two subsets with a fixed probability, each peer selects a piece to be downloaded from the selected subset according to the Rarest First policy, where the probability of selecting the former subset is given by $p \in [0, 1]$. Note that the parameter p controls the balance between the deadline requirement and the diversity of the collected pieces; e.g., for large p's, each peer can have a more chance to acquire pieces before their deadline, and for small p's, each peer can acquire rare pieces which would become a bottleneck in the future.

4. Proposed Scheme

4.1 System Architecture

Similar to BASS, our system architecture consists of a tracker, a media server, and a collection of peers connected by a P2P network. The role of the tracker is the same as the original BitTorrent. The role of the media server is to store

all video files, and to upload those files upon request. The configuration of the network is controlled by the tracker.

After joining the network, each peer starts the download of a requested file by repeatedly exchanging pieces among adjacent peers. Each peer can also download pieces directly from the media server so as to meet the deadline, i.e., when it recognizes that the remaining time before its playback becomes as short as the estimated download time from the media server, each peer directly asks the media server to deliver that piece, where the download time is estimanted by dividing the size of pieces by the download bandwidth.

Note that such a way of estimation merely gives a lower bound on the download time since it would significantly increase if a large number of peers wish to download pieces from the media server at the same time (as will be described later, the objective of our proposed scheme is to bound the number of such requests to the media server as low as possible). In fact, the media server becomes a bottleneck if it receives a number of requests in a short time period, and such a situation frequently occurs if there are only few copies of each piece in the network. In many VoD systems such as YouTube, we cannot avoid such a situation since it provides a large number of "unpopular" videos each of which is shared by a small number of peers. Although such a concentration of the load to the media server could be alleviated by the piece selection rule adopted in BiToS, it is not enough in many cases.

4.2 Missing Piece Request

The first technique used in our proposed scheme is to conduct a "prefetch" of missing pieces from the media server, where a missing piece is a piece which is not held by any peer in the network (note that it is different from the notion of rarest pieces used in the Rarest First policy, since a piece selected under the policy should exist in the network). In the following, we call such a request for a missing piece missing piece request (MPR, for short). The performance of such a prefetching scheme is significantly affected by the timing and the frequency of MPRs issued by each peer, in addition to the selection of pieces to be requested. We are going to design our scheme in such a way that each peer issues an MPR only when there are no pieces which can be downloaded from the adjacent peers. Note that it is different from a simple deadline driven scheme, since in our scheme, each peer can issue an MPR even if the deadline for unacquired pieces is not critical. In addition, we design the scheme such that an MPR is issued only for those pieces included in its high priority set, where the definition of the high priority set is the same as the definition used in BiToS, and that if there are several candidates, it selects a random piece to avoid a collision with MPRs issued by the other peers.

A key point in this approach is how to estimate missing pieces in the network. Although such an information could be acquired by aggregating the piece availability of all peers to the media server, it causes an extra load to the media server. On the other hand, if the acquired information is not accurate, and many of identified pieces could be acquired through the local communication among nearby peers, it would also unnecessarily increase the load of the media server. In the proposed scheme, each peer periodically collects the piece availability of nearby peers by flooding a query with an appropriate TTL. Each query contains the peer ID, IP address, and the port number of the requester, as well as a TTL and a sequence number representing the number of queries having been issued by the requester, which will be used to reduce the number of responses to the same requester. More concretely, each peer returns a response to a query only when the sequence number of the query is greater than the largest sequence number of queries received from the same requester.

In the original BitTorrent protocol, the notion of local piece availability (LPA) is used to identify a set of rarest pieces, where the LPA of a peer is an accumulated value of the piece availability in its adjacent peers. In this paper, in order to estimate a set of missing pieces as accurately as possible, a new notion of piece availability called chain piece availability (CPA) is introduced. The CPA of a peer is a bit array representing the piece availability in a region centered at the peer. It is initialized by using the LPA of the corresponding peer, in such a way that an element in the CPA takes value 1 if and only if the corresponding element in the LPA takes a positive value, and is propagated through the network by conducting an issue of queries in a "chain-reaction" manner. More concretely, each peer repeatedly issues a query requesting for CPAs, and after receiving CPAs from nearby peers (within a given TTL) as the query-response, each peer updates its CPA, such that: 1) to take a bit-wise OR of its own CPA and received CPAs, and 2) to replace the CPA by the outcome of the operation. After calculating the array, each peer recognizes that pieces whose value of the CPA are 0 are missing pieces. Finally, in order to reflect the removal of pieces to the CPA, each peer periodically refreshes the CPA.

4.3 Switching MPR

The second technique developed in the proposed scheme is to switch the mode of peers according to the estimated size of the P2P network. In the following, we call such a mode switch switching MPR (SMPR, for short). Direct download of missing pieces from the media server works effectively when the number of peers is relatively small. However, as will be shown in Sect. 5, the performance of the scheme gradually degrades as increasing the number of peers in the network, while the cost of MPRs increases in proportion to the number of peers. A reason of such phenomena is that an increase of the number of non-adjacent peers makes it difficult to accurately estimate the set of missing pieces in the network, and the number of missing pieces decreases as increasing the number of peers, since for each piece, the possibility of being held by at least one peer should increase.

We overcome such problem by conducting SMPRs.

Let N be a local variable representing the estimated number of peers in the network. In the scheme, each peer switches its mode by the value of N; i.e., if it is smaller than a predetermined threshold, it switches to the Small_Network mode, and issues queries requesting for CPAs and MPRs according to the policy described in the last subsection; otherwise, it switches to the Large_Network mode, and stops to issue queries and MPRs, where it returns the LPA instead of the CPA if it receives a query requesting for a CPA (the impact of the threshold to the performance will be evaluated in Sect. 5). The value of N is updated at each time of requesting a set of peers to the tracker, where in the BitTorrent protocol, each peer should periodically acquire such a list of peers from the tracker. A concrete way of estimating the value of N from the response received from the tracker is given in the next subsection.

4.4 Distributed Estimation of Network Size

Finally, we describe a procedure to estimate the size of the given network which is locally executed by each peer. The procedure is based on a random sampling. Let $\Gamma(i)$ denote the set of adjacent peers of peer i, and N^* denote the number of peers currently participating in the system (note that the value of N^* is not disclosed to each peer). Suppose that the tracker returns a set of peers S as a response to the request from peer i. For simplicity, we fix the cardinalities of $\Gamma(i)$ and S to a constant. Let Y denote a random variable representing the cardinality of set $\Gamma(i) \cap S$. If each element in S is selected randomly, for each element in $\Gamma(i)$, the probability that the element is selected as an element in S is $1/N^*$. Thus, by the linearity of expectation, the expected value of Y is $\frac{|S|\times|\Gamma(i)|}{N^*}$, which means that N^* is calculated as in the following formula:

$$N^* := \frac{|S| \times |\Gamma(i)|}{E[Y]}.$$

In the above formula, $E[Y]$ can be approximated by repeating the calculation of $\Gamma(i) \cap S$ for different S (and $\Gamma(i)$), provided that the size of the network does not change.

5. Evaluation

5.1 Setup

We evaluate the performance of the proposed scheme by simulation with respect to the following metrics: 1) the total amount of pieces U_s [Mb] uploaded by the media server, 2) the average wait time W [s] of peers caused by suspending of a video play, and 3) the total number of queries Q exchanged among all peers. Each result is an average over 20 runs. To exclude possible ambiguity, we call a method issuing MPRs the "MPR method," and a method conducting SMPRs the "SMPR method." The performance of these two methods are compared with a method in which no peer issues an MPR (we call it the "Normal method"). In all methods, we assume that pieces downloaded from its adjacent peers are selected according to the same rule to BiToS.

Parameters used in the simulation are determined as follows. In each run of the simulation, The total number of peers arriving at the system is selected from $\{100, 101, \ldots, 300\}$. All peers are homogeneous, and each peer can maintain at most 30 connections to the other peers. Communication bandwidth of each peer is 1024 [Kbps] in each of upload/download directions, and the communication bandwidth of the media server is 30 [Mbps]. The ratio of the high priority pieces used in BiToS is 8% of the entire pieces as is recommended in [9], and the probability p of selecting the high priority set is 0.8 as in [9]. The interval of requesting a list of peers to the tracker and the interval for SMPRs are both fixed to 15 [s], and the propagation speed of queries used in calculating CPAs is 1 [hop/s].

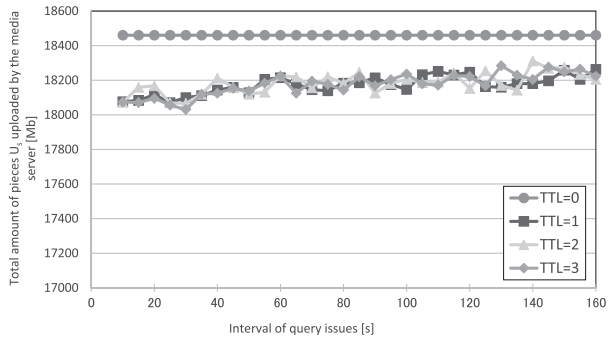
We assume that the system contains exactly one video file of length 600 [s], with the playback bit-rate of 512 [Kbps]. As for the behavior of the peers, we consider the following scenario: All pieces are held by the media server. Each peer arrives at the system according to a Poisson distribution with average arrival interval t [s], where initially, a newly arrived peer has no pieces. Upon arrival, it downloads one random piece from the media server. The piece exchange among peers and the play of the video will start immediately after completing the download of the first piece from the media server, and the playback continues until it reaches the end of the file. Finally, a peer leaves the system as soon as it completes the playback.

5.2 Tuning of Parameters in the MPR Method

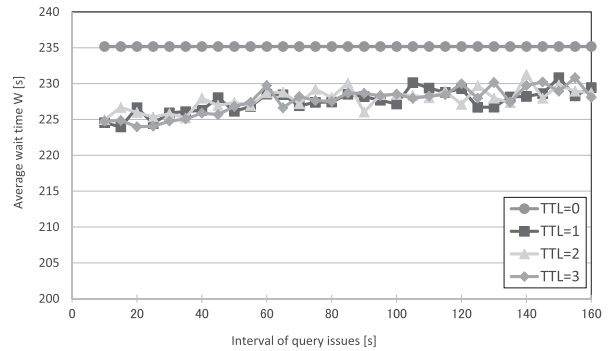
At first, we conduct preliminary experiments to determine the TTL and the interval of query issues used in the proposed scheme. In the simulation, we consider an extreme case such that 300 peers arrive at the system with $t = 0$ (i.e., simultaneously), since the impact of MPRs to the performance of the scheme is maximized in such a bursty situation. In fact, under such a situation, playback positions of peers are close to each other so that missing pieces cause bursty requests to the media server provided that MPRs are not used.

Figure 3 summarizes the results. In this figure, "TTL=0" indicates that each peer estimates missing pieces merely by referring to its own LPA. As shown in Fig. 3 (a), the total amount of pieces U_s uploaded by the media server increases as increasing the time interval, while there is no significant difference among three positive TTLs. A similar phenomenon could be observed for the average wait time W as shown in Fig. 3 (b), which indicates that a heavy load of the media server causes a suspending of the video play. In contrast, as shown in Fig. 3 (c), the total number of queries Q rapidly increases as decreasing the time interval particularly when the TTL is large.

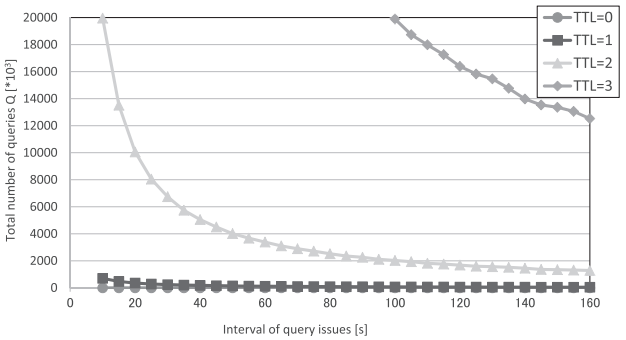
Such a behavior of the scheme is due to the accuracy of the estimation of missing pieces. Recall that a long time interval implies that peers should use an old availability information, although it could provide an accurate information



(a) Total amount of pieces U_s uploaded by the media server.



(b) Average wait time W.



(c) Total number of queries Q.

Fig. 3 Performance of the MPR method (300 peers).

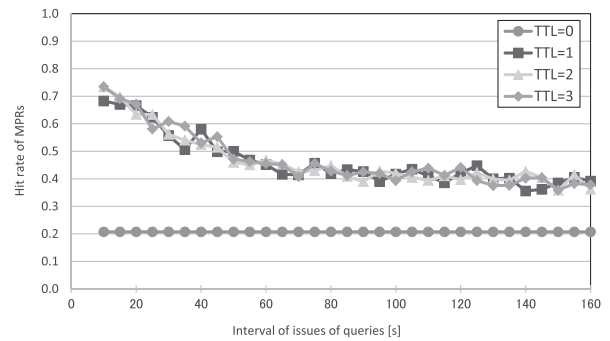
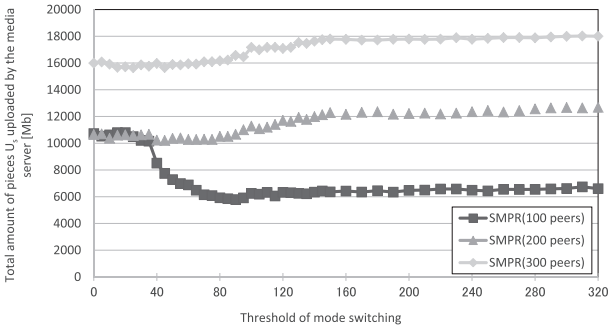


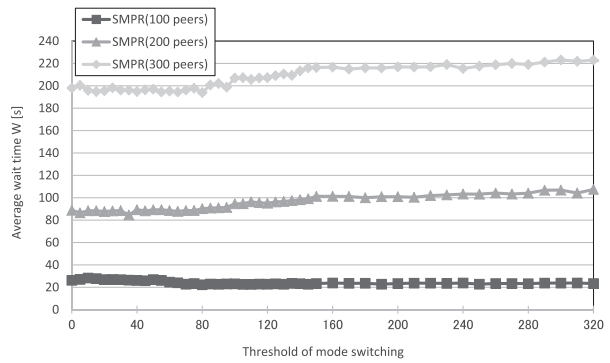
Fig. 4 Hit rate of MPRs for the MPR method (300 peers).

about missing pieces compared with simple LPA. To clarify this point, we evaluated the hit rate of the scheme which is defined as the ratio of the number of (actual) missing pieces to the total number of MPRs issued by all peers. Figure 4 shows the result. We can observe that the hit rate rapidly decreases as increasing the time interval, and that there is no significant difference among positive TTLs; i.e., it is enough to use a small TTL to attain a hit rate around 0.7 provided that the time interval is sufficiently small.

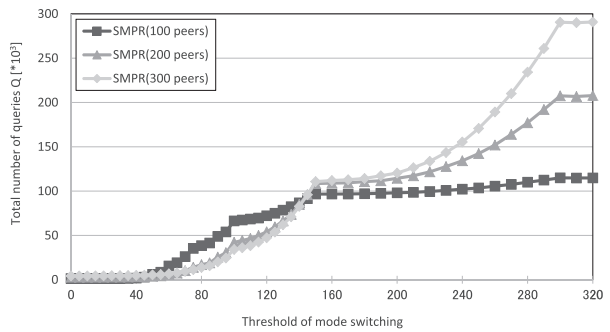
According to the above observations, in the following experiments, we fix the TTL of each query to one, and the time interval to 15 [s].



(a) Total amount of pieces U_s uploaded by the media server.



(b) Average wait time W.



(c) Total number of queries Q.

Fig. 5 Performance of the SMPR method.

5.3 Tuning of Threshold in the SMPR Method

Next, we conduct experiments to determine the threshold used in the SMPR method. We select the total number of peers arriving at the system from 100, 200, or 300, and evaluate the performance by varying the threshold from 0 to 320. All peers are assumed to arrive at the system simultaneously, for the same reason to the previous simulation.

Figure 5 summarizes the results. As shown in Fig. 5 (a),

when the number of peers is 100, the total amount of pieces U_s uploaded by the media server dramatically reduces by setting the threshold to 40 or larger; i.e., MPRs significantly reduce the load of the media server since there are many missing pieces in such small networks. On the other hand, when the number of peers becomes large, U_s increases by setting the threshold to 90 or larger; i.e., MPRs cause an extra server load because of an inaccuracy of the estimation of missing pieces. A similar effect could be observed for the average wait time W, as shown in Fig. 5 (b).

The total number of queries Q increases as increasing the threshold as shown in Fig. 5 (c), which is apparently due to an increase of the number of peers in the Small_Network mode which repeatedly issue a query to their nearby peers. In fact, when the threshold is smaller than 150, a network consisting of 100 peers exchanges more messages than a network consisting of 300 peers, since there are a larger number of peers in the Small_Network mode in such small networks. In contrast, when the threshold becomes large, a small network exchanges less number of messages since it contains smaller number of peers in the Small_Network mode compared with a large network.

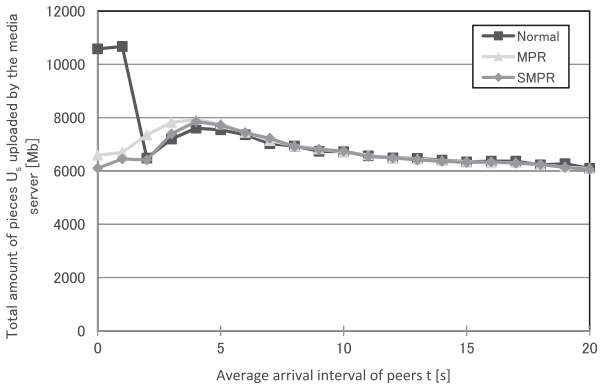
According to the above observations, in the following experiments, we fix the threshold to 70.

5.4 Comparison of Three Methods

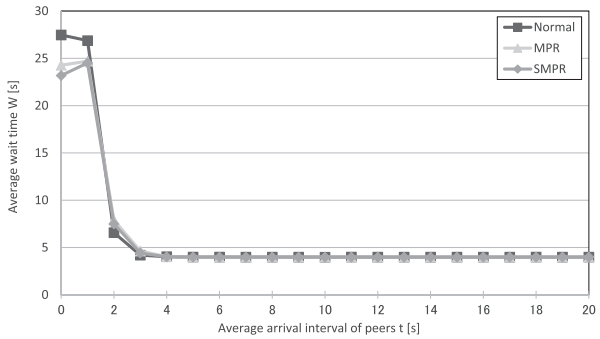
Next, we compare the performance of two proposed methods with the Normal method by varying the average arrival interval t from 0 to 20 [s]. In the simulation, we assume that 100 peers are sequentially arriving at the system with a designated time interval.

Figure 6 summarizes the results. As shown in Fig. 6 (a), when $t \leq 1$, two proposed methods significantly reduce the total amount of pieces U_s uploaded by the media server in the Normal method, e.g., the SMPR method reduces the amount by more than 39% of the Normal method. On the other hand, when $t \geq 2$, all methods could bound U_s because of the variance of the arrival time. A similar phenomenon could be observed for the wait time W as shown in Fig. 6 (b); e.g., when $t \leq 1$, the SMPR method reduces W of the Normal method by 8%. Figure 6 (c) shows the result for the total number of queries Q. Although Q in the SMPR method is bounded small for small t's, it should exchange as large number of queries as the MPR method for large t's, which is because the number of peers participating in the network becomes small for large t's, i.e., most of peers are in the Small_Network mode in such cases.

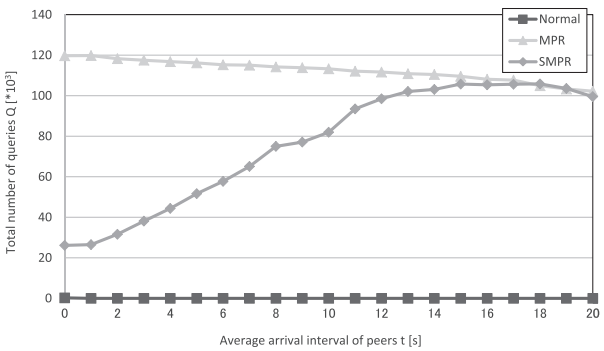
Finally, we examine the relationship between U_s and W of the three methods by varying the communication bandwidth of the media server from 10 to 160 [Mbps] and by plotting U_s with its corresponding W for each setting. In the simulation, we assume that 200 peers are sequentially arriving at the system with $t = 1$. Figure 7 (a) shows the result. Note that a point with larger W corresponds to the result with a narrower bandwidth of the media server, as shown in Fig. 7 (b). In this setting, the arrival of peers is bursty and



(a) Total amount of pieces U_s uploaded by the media server.



(b) Average wait time W.

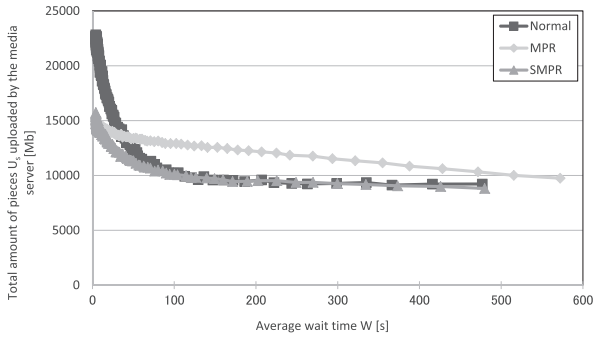


(c) Total number of queries Q.

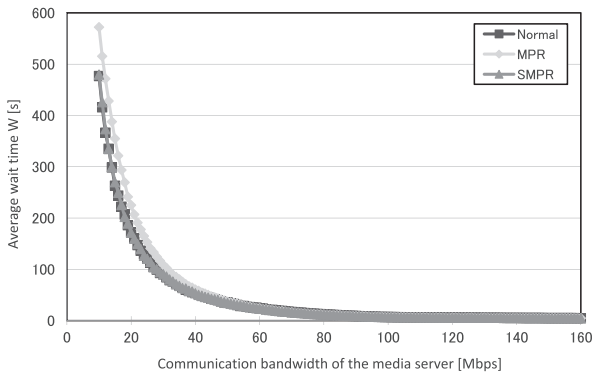
Fig. 6 Comparison of three methods with different average arrival interval t.

the existence of missing pieces is critical. Thus the Normal method should download many pieces from the media server to ensure small W; i.e., U_s is severely affected by W. On the other hand, two proposed methods can prevent the media server from receiving bursty requests by conducting a prefetching and a pre-propagation of missing pieces. Then U_s does not grow large even for small W's and is not affected significantly by W.

We can also observe from Fig. 7 (a) that when W is large, the fluctuation of U_s becomes small. Recall that a point with large W corresponds to a narrow bandwidth of the media server, i.e., it becomes a bottleneck and fully uses its bandwidth at any time. Under such conditions, U_s is proportional to the bandwidth of the media server, and lightly affected by a small difference of the bandwidth, while W



(a) Total amount of pieces U_s uploaded by the media server as average wait time W of peers.



(b) Average wait time W of peers as communication bandwidth of the media server.

Fig. 7 Comparison of three methods with different communication bandwidth of the media server.

is heavily affected by the difference, since the difference largely affects the propagation of pieces. Furthermore, when the bandwidth of the media server is narrow, extra MPRs significantly affect W by consuming the bandwidth, thus the performance of the MPR method degrades in such conditions. In contrast, the SMPR method exhibits a good performance even in such severe situations, since it controls the amount of MPRs by the mode switching.

6. Concluding Remarks

In this paper, we propose a data prefetching scheme for P2P VoDs with a media server. The first idea of the scheme is to directly request missing pieces to the media server by estimating the set of missing pieces as accurately as possible, and the second idea is to switch the mode of each peer in such a way that the request for the missing pieces is issued only when the estimated size of the network is sufficiently small. The result of simulations indicates that the proposed scheme reduces the load of the media server by 39% when the arrival of peers is bursty, compared with a scheme without prefetching.

A future work is to refine the notion of CPA to improve the accuracy of the estimation of the set of missing pieces. In addition, we should evaluate the performance of the scheme under peer churn.

References

[1] YouTube. http://www.youtube.com/
[2] USTREAM. http://www.ustream.tv/
[3] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," Proc. Nineteenth ACM Symposium on Operating Systems Principles, pp.298–313, 2003.
[4] Y.-H. Chu, S.G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," Proc. ACM SIGMETRICS, pp.1–12, 2002.
[5] B. Cohen, "Incentives build robustness in BitTorrent," Proc. 1st Workshop on Economics of Peer-to-Peer Systems, 2003.
[6] C. Dana, D. Li, D. Harrison, and C.-N. Chuah, "BASS: BitTorrent assisted streaming system for video-on-demand," Proc. International Workshop on Multimedia Signal Processing, pp.1–4, 2005.
[7] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-to-peer patching scheme for VoD service," Proc. 12th Int. Conf. World Wide Web, pp.301–309, 2003.
[8] S. Sakashita, T. Yoshihisa, T. Hara, and S. Nishio, "A data reception method to reduce interruption time in P2P streaming environments," Proc. 13th Int. Conf. Netw.-Based Inf. Syst., pp.166–172, 2010.
[9] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," Proc. 25th IEEE Int. Conf. Comput. Commun., pp.1–6, 2006.
[10] X. Zhang, J. Liu, B. Li, and T.-S.P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," Proc. 24th Annual Joint Conference of the IEEE Comput. and Commun. Societies, pp.2102–2111, 2005.
[11] Y. Zhou, D.-M. Chiu, and J.-C. Lui, "A simple model for analyzing P2P streaming protocols," Proc. IEEE Int. Conf. Netw. Protocols, pp.226–235, 2007.

Ryusuke Uedera received the B.E. degree in information engineering, from Hiroshima University in 2010. He is a Master Student at the Graduate School of Engineering, Hiroshima University. His research interests include peer-to-peer networks and distributed systems.

Satoshi Fujita received the B.E. degree in electrical engineering, M.E. degree in systems engineering, and Dr.E. degree in information engineering from Hiroshima University in 1985, 1987, and 1990, respectively. He is a Professor at Faculty of Engineering, Hiroshima University. His research interests include communication algorithms on interconnection networks, parallel algorithms, graph algorithms, and parallel and distributed computer systems. He is a member of the Information Processing Society of Japan, SIAM Japan, IEEE, and SIAM.