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

Ryusuke Uedera and Satoshi Fujita
Department of Information Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, Japan
Email: {ryusuke,fujita}@se.hiroshima-u.ac.jp

Abstract—**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 BitTorrent protocol could not collect all 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 by carefully discriminating pieces to be requested, by taking into account the rareness of the pieces held by its nearby peers, estimated size of the overlay network, and the file size to be downloaded. The performance of the proposed scheme is evaluated by simulation.**

I. Introduction

Video-on-Demand (VoD) is an online service which enables customers to watch their favorite videos at any time, at any place [1]. In this service, video file requested by a user is downloaded from the VoD system, while it is not necessary to wait for the completion of the entire download before starting the play of the video, i.e., she can start watching while download is in progress. Many of existing VoD systems such as YouTube and Ustream are implemented under the traditional client/server (C/S) model [2], [3]. Thus, although they could enjoy several advantages of the C/S model such as an efficient resource management and a secure contents delivery, they have a serious drawback such that the central server easily becomes a bottleneck, and it causes a single point of failure in the worst case.

In order to overcome such problems in conventional VoDs, 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 to 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 peers reduces the load of a server in the C/S model, and in many cases, it significantly increases 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 [4] is a P2P file sharing protocol based on the notions of file chunking, quick piece propagation using the rarest first piece selection rule, and an incentive mechanism based on the tit-for-tat strategy. BASS [9] is the first attempt to combine the BitTorrent with conventional VoDs. In [10], 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 Section III). 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 portion of the file is not held by any peer. Such a "missing" piece can not be acquired by any peer until another peer having 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, the load of the media server becomes heavy 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 the 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. The accuracy of such a local estimation becomes sufficiently high when the number of peers in the corresponding P2P overlay is relatively small (as will be described later, we assume that a P2P overlay 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 P2P overlay and the file size to be downloaded.

The performance of the proposed scheme is evaluated by simulation. The result of simulations indicates that: 1) the prefetching protocol certainly reduces the load of the media server when the size of the P2P overlay is small, e.g., if it consists of less than 150 peers, the load is reduced by 37% of a conventional scheme; 2) the mode switching technique effectively adapts the scheme to the underlying P2P overlay 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 Section II, we overview related works. Section III gives a detailed description of the BitTorrent and its application to the P2P VoD. Our proposed scheme is described in Section IV. The result of performance evaluation is summarized in Section V. Finally, Section VI concludes the paper with future works.

II. 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 overlay, i.e., tree-based systems and mesh-based systems.

In tree-based systems, participant peers organize a tree-structured overlay 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 [5] 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 [6] 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 overlay in mesh-based systems can take any structure, and each link in the overlay can be used in both directions. Thus, by adopting a graph structure with high connectivity as an underlying P2P overlay, we could effectively resolve the drawback of tree-based systems. In mesh-based systems, each file is divided into several chunks, and each peer downloads chunks from adjacent peers, while uploading other chunks to the adjacent peers. CoolStreaming [8] 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 [9] is the first P2P VoD based on that idea. In this system, chunks can be downloaded from adjacent peers, in addition to the ordinary download from the media server. In [10], Vlavianos *et al.* improved the piece selection rule in the 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.

III. 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.

A. 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 a 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. After that, each peer periodically requests a list of peers to the tracker, 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 in determining 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.

B. 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 the peers.

C. 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.

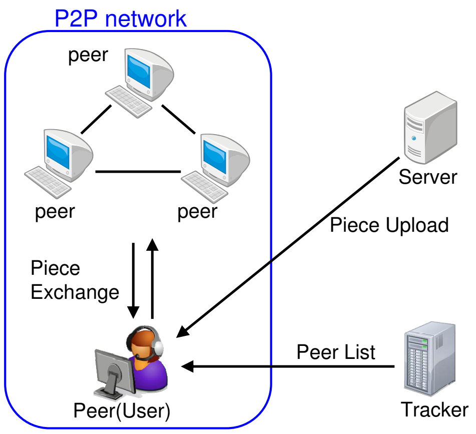

Fig. 1. System architecture of the BASS.

D. BASS

BASS is the first P2P VoD based on the BitTorrent protocol. System architecture of BASS is illustrated in Figure 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 (random) download from adjacent peers, where the rule for the piece selection 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.

E. BiToS

BiToS is anathor P2P VoD developed by Vlavianos *et al.* [10]. Figure 2 illustrates the system architecture of the 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 rare pieces independent of the position in the given media file. Thus, it is difficult to collect all pieces before deadline under the piece selection rule adopted in the original BitTorrent.

To overcome such a problem, the BiToS adopts the

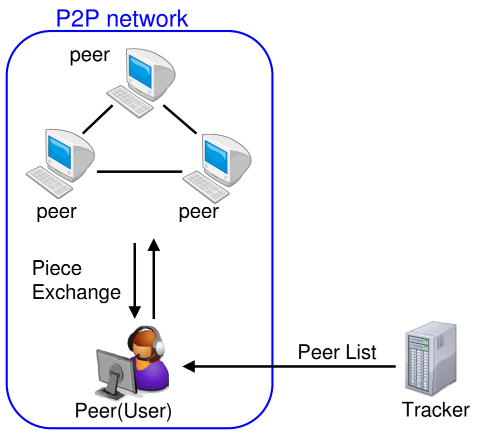

Fig. 2. System architecture of the BiToS.

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 subset by flipping a coin, 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 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.

IV. Proposed Scheme

A. System Architecture

Similar to the BASS, our system architecture consists of a tracker, a media server, and a collection of peers connected by an overlay network. The role of the tracker is the same as the original BitTorrent protocol. The role of the media server is to store all video files, and to upload those files upon request. The configuration of the overlay network is controlled by the tracker. After joining the network, each peer starts to download 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 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.

In such a hybrid approach, the load of the media server becomes a bottleneck if it receives a large number of requests in a short time period. Such a situation frequently occurs if there are only few copies of each piece in the overlay, and in fact, 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 the BiToS, it is not enough in many cases.

B. Missing Piece Request

The first technique used in our proposed scheme is to conduct a "prefetch" of missing pieces from the media server, where **missing piece** is a piece which is not held by any peer in the overlay (note that it is different from the notion of rarest pieces used in the Rarest First policy, since a piece selected under the rarest first protocol 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 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 the unacquired pieces is not critical.

A key point in this approach is how to identify missing pieces in the overlay. 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 of the requester, sequence number concerned with the requester, and the TTL, where the sequence number is the number of queries issued by the requester which is used to avoid multiple responses to the same query. More concretely, if it receives a copy of a query, each peer returns a response to the query only when the sequence number of the query is larger than the largest sequence number received from the same requester,

In the original BitTorrent protocol, the notion of *local piece availabiliy (LPA)* is used to identify a set of rarest pieces, where LPA of a peer is an accumulated value of the piece availability in its adjacent peers. In this paper, in order to identify a set of missing pieces as accurately as possible, a new notion of piece availability called *chain piece availability (CPA)* is introduced. 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 CPA are 0 are missing pieces. Finally, in order to reflect the removal of pieces to the CPA, each peer periodically refreshes the CPA.

C. Switching MPR

The second technique developed in the proposed scheme is to switch the mode of peers according to the estimated size of the overlay 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 Section V, the performance of the scheme gradually degrades as increasing the number of peers in the network, while the cost of MPR 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 decrease 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. Consider a peer who is currently playing a video, and let X be the number of pieces constitute the corresponding file. 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/X; i.e., if it is smaller than a predetermined threshold, it switches to the Small_Network mode, and issues CPA request queries and MPRs according to the policy described in the last subsection; otherwise, it switches to the Large_Network mode, and stops to issue the queries and MPRs, where it returns LPA instead of CPA if it receives a request for a CPA under the Large_Network mode (the impact of the threshold to the performance will be evaluated in Section V). 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.

The reader should note that in the resulting scheme, each peer conducts an SMPR by referring to the value of N/X, instead of N or X. By normalizing the estimated number of peers by the number of pieces to be acquired, we can take into account the probability that each piece is held by a peer in the network. More concretely, we can make the following observations:

- For any fixed N, a peer becomes the Small_Network mode if the number of pieces need to be acquired is larger than the threshold. As long as being in the Small_Network mode, it can issue MPRs to the media server *if and only if* it detects that there are no pieces which can be downloaded from the adjacent peers (note that if there exists such a piece, it does *not* issue an MPR and tries to collect all pieces merely through the communication to the adjacent peers).
- The value of the threshold changes in proportion to the estimated size of the network. That is, the threshold on the number of pieces becomes twice if the estimated network size is a twice. This reflects an observation such that the issuing MPRs is particularly effective in small networks.

D. 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 total number of peers 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.

V. Evaluation

A. Setup

We conducted simulations to evaluate the performance of the proposed scheme. In the simulation, we evaluate the amount of pieces directly downloaded from the media server, and the total number of queries exchanged among all peers. In the following, we denote the amount of pieces downloaded from the media server by U_s [Mb], and the total number of queries by Q. Each result is an average over five 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 will call it the "Normal method"). In all methods, we assume that the pieces downloaded from its adjacent peers are selected according to the same rule to the BiToS.

Parameters used in the simulation are determined as follows. In each run of the simulation, the number of peers is fixed to a value selected from $\{50, \ldots, 500\}$. All peers are homogeneous, and each peer can maintain at most 30 connections to the other peers. Communication bandwidth of each peer is fixed to 1024 Kbps in each of the upload/download directions, and the communication bandwidth of the media server is not limited. The ratio of the High Priority Set used in the BiToS is 8% of the entire pieces, as was recommended in [10]. The probability p of selecting such High Priority Set is fixed to 0.8. The propagation speed of queries used in calculating CPA is fixed to 1 peer per second.

We assume that the system contains exactly one video file of length 600 sec, with the playback bit-rate of 512 Kbps. As for the behavior of the peers, we consider the following scenario: All peers simultaneously arrive at the system at time 0, where initially all pieces are held merely by the media server. Thus each peer should download random pieces from the media server upon arrival (although such a bursty request will significantly increase the traffic around the media server, in this paper, we omit such an issue by assuming that the bandwidth of the media server is unlimited). The piece exchange and the playback of the video will start after downloading one random piece from the media server, and the playback continues until it reaches the end of the file. Then, after completing the playback, each peer immediately leaves the system.

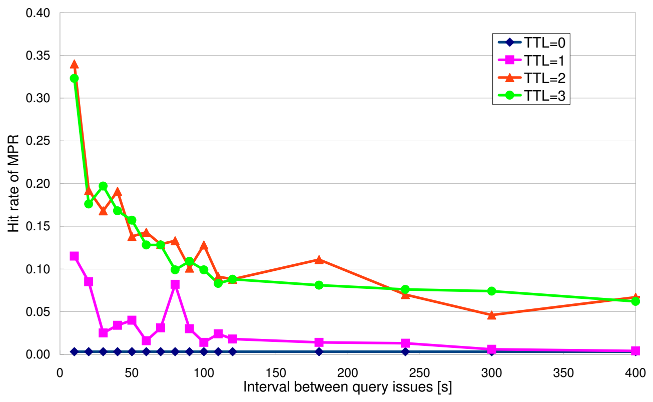

Fig. 3. Hit rate of the MPR (500 peers).

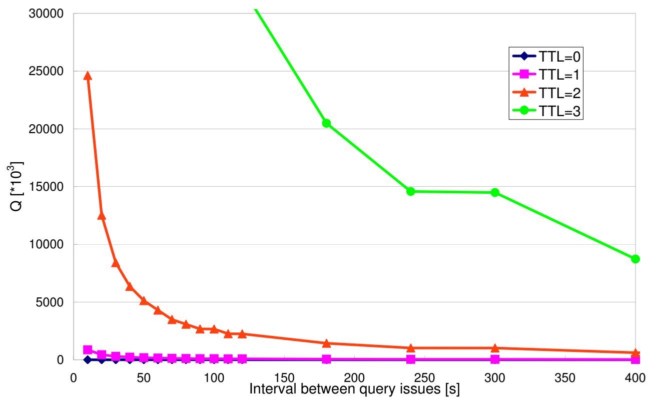

Fig. 4. The total number of queries Q (500 peers).

B. Impact of TTL and the interval between query issues

At first, we evaluate the accuracy of the estimation of missing pieces by measuring the hit rate of MPRs. The hit rate is formally defined as the ratio of the number of (actual) missing pieces to the total number of MPRs issued by all peers. We evaluate the hit rate by varying the TTL and the time interval of consecutive queries, by fixing the number of peers to 500.

Figure 3 illustrates the result. In this figure, "TTL=0"

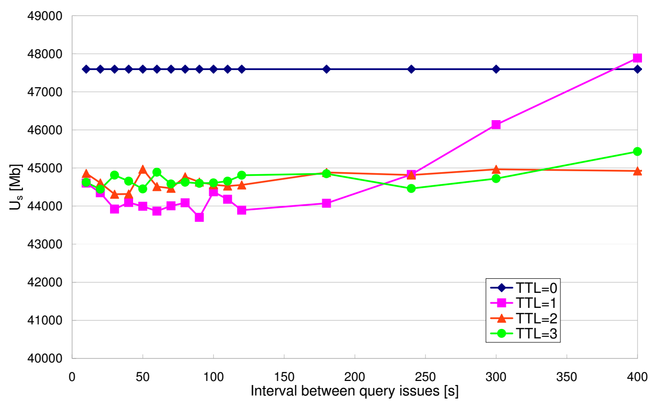

Fig. 5. Total upload U_s of the media server (500 peers).

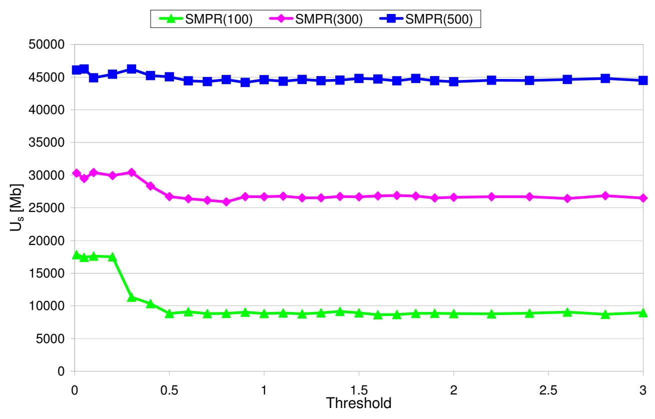

Fig. 7. Total upload U_s of the media server.

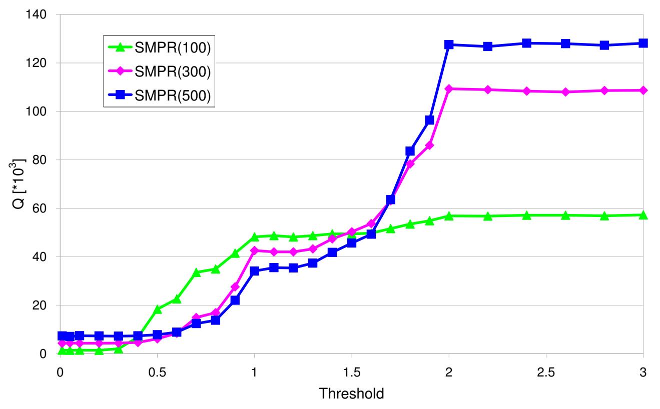

Fig. 6. Total number of queries Q.

indicates that each peer makes a decision merely by referring to its own LPA. We can observe that the hit rate increases as increasing the TTL and by shortening the time interval, while it saturates when TTL=2; i.e., it is not necessary to increase the TTL from two to three to improve the hit rate. Figure 4 shows the total number of queries Q exchanged under the same setting. This result indicates that Q significantly increases as increasing the TTL from one to two, particularly when the time interval is short. Figure 5 shows the total amount of pieces U_s uploaded by the media server. It indicates that the case of TTL=0 is apparently worse than the other cases, and that the case of TTL=1 is slightly better than other positive TTLs for short time intervals, although the hit rate of TTL=1 is lower than other two cases.

According to the above observations, in the following experiments, we fix the TTL of the query to one, and the time interval to 30 sec.

C. *Threshold of SMPR*

Next, we evaluate the impact of the threshold used in the SMPR. More concretely, we measure Q and U_s, by varying the threshold from 0.01 to 3.0. The number of peers in the network is fixed to either 100, 300, or 500. Figure 6 illustrates the result for Q. The value of Q gradually increases as increasing the threshold, and after exceeding 2.0, it reaches an upper bound, i.e., almost all peers become the Small_Network mode under such a high threshold. The result for U_s is shown in Figure 7. The value of U_s sharply changes between 0.2 and 0.5, which corresponds to the range of threshold under which the number of peers in the Small_Network mode gradually increases. In addition, the amount of reduction is more significant for networks of smaller sizes, and the timing of U_s starting to decrease in small networks is slightly early than the timing for large networks. This is because networks of smaller sizes are more sensitive to the number of Small_Network mode peers, since such networks have a lot of missing pieces and require Small_Network mode peers. Considering the balance between Q and U_s, in the following experiments, we fix the threshold for the SMPR to 0.8.

D. *Comparison of methods*

Finally, we compare the performance of three methods, i.e., the Normal method, the MPR method and the SMPR method in terms of metrics Q and U_s. Figure 8 shows the result for U_s, where the horizontal axis is the number of peers in the network. Two proposed methods need less amount of uploads than the Normal method. In particular, when the number of peers is smaller than 150, the amount of reduction is more than 37% of the Normal method. Figure 9 illustrates the result for Q. As shown in

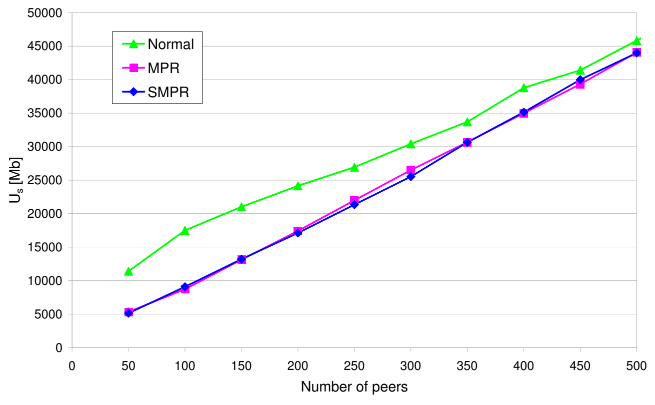

Fig. 8. Total upload U_s of the media server.

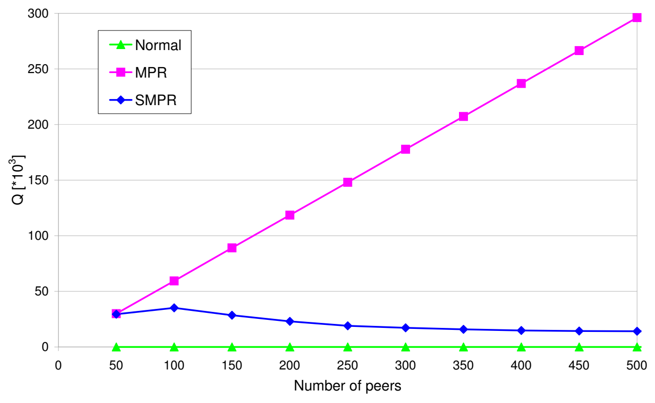

Fig. 9. Total number of queries Q.

the figure, the number of queries exchanged in the MPR method increases in proportion to the number of peers, since all peers periodically issue a query. On the other hand, the number of queries exchanged in the SMPR method is bounded by 36000, and gradually decreases as increasing the number of peers, which is apparently due to the effect of the SMPR. Consequently, the SMPR method could significantly reduce the amount of server upload under the Normal method without significantly increasing the overall cost.

VI. 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 and the number of pieces constituting the downloaded file is sufficiently large. The result of simulation indicates that the proposed scheme reduces the load of the media server by 37%, 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 improve the proposed scheme so as to relax the bursty requests to the media server, and evaluate the performance of the scheme in terms of the traffic around the media server.

References

[1] Wikipedia - Video on demand. *http://en.wikipedia.org/wiki/Video_on_demand*

[2] YouTube. *http://www.youtube.com/*

[3] USTREAM. *http://www.ustream.tv/*

[4] BitTorrent. *http://www.bittorrent.co.jp/*

[5] Y-H. Chu, S. G. Rao, S. Seshan and H. Zhang. A Case for End System Multicast. *In ACM SIGMETRICS*, 1–12, 2002.

[6] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. *In ACM SOSP*, 298–313, 2003.

[7] Y. Guo, K. Suh, J. Kurose and D. Towsley. P2Cast: Peer-to-Peer Patching Scheme for VoD Service. *In WWW Conference*, 301–309, 2003.

[8] 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. *In IEEE INFOCOM*, 2102–2111, 2005.

[9] C. Dana, D. Li, D. Harrison and C-N. Chuah. BASS: BitTorrent Assisted Streaming System for Video-on-Demand. *In IEEE MMSP*, 1–4, 2005.

[10] A. Vlavianos, M. Iliofotou and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. *In IEEE Global Internet Symposium*, 1–6, 2006.