Available online at www.sciencedirect.com

ScienceDirect

Journal of Systems Architecture 54 (2008) 55–69

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

A novel caching mechanism for peer-to-peer based media-on-demand streaming

Ye Tian [*], Di Wu, Kam-Wing Ng

Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong

Received 31 December 2006; received in revised form 23 March 2007; accepted 28 March 2007
Available online 27 April 2007

Abstract

In recent years, peer-to-peer networks and application-level overlays without dedicated infrastructure have been widely proposed to provide on-demand media services on the Internet. However, the scalability issue, which is caused by the asynchronism and the sparsity of the online peers, is a major problem for deploying P2P-based MoD systems, especially when the media server's capacity is limited. In this paper, we propose a novel probabilistic caching mechanism for P2P-based MoD systems. Theoretical analysis is presented to show that by engaging our proposed mechanism with a flexible system parameter, better scalability could be achieved by a MoD system with less workload imposed on the server, and the service capacity of the MoD system could be tradeoff with the peers' gossip cost. We verify these properties with simulation experiments. Moreover, we show by simulation results that our proposed caching mechanism could improve the quality of the streaming service conceived by peers when the capacity of the server is limited, but will not cause notable performance degradation under highly lossy network environments, compared with the conventional continuous caching mechanism.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Peer-to-peer network; Media-on-demand; Overlay

1. Introduction

With the deployment of broadband techniques, peer-to-peer (P2P) based systems have been widely proposed to provide the media-on-demand service (MoD) on the Internet in recent years. The basic idea of a P2P-based MoD system is to allow the peers accessing the same media object to share the media data cached in their buffers with each other, thus reducing the workload imposed on the media server of the system. Typically, for a P2P-based MoD system, an index overlay and a data overlay will be formed by the online peers, as demonstrated in Fig. 1.

The index overlay of a P2P-based MoD system is usually formed by all the online peers, and serves for managing the information such as the peers' memberships and their current playback offsets. With an index overlay, incidents such as peer joining, peer departure and VCR operations (e.g., pause, fast-forward, rewind, etc.) could be handled in a distributed fashion efficiently. Usually some well designed structures and dedicated algorithms (e.g., trees,

[*] Corresponding author. Tel.: +852 3163 4253.
E-mail addresses: ytian@cse.cuhk.edu.hk (Y. Tian), dwu@cse.cuhk.edu.hk (D. Wu), kwng@cse.cuhk.edu.hk (K.-W. Ng).

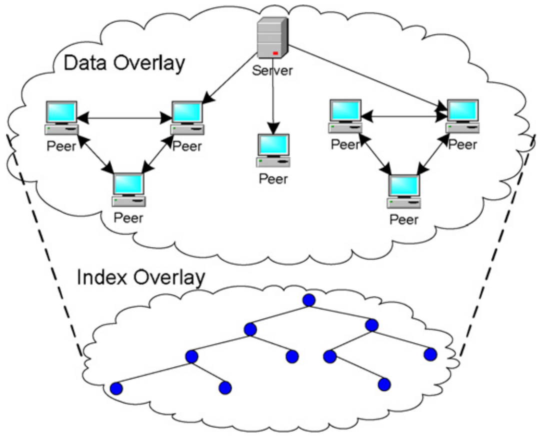

Fig. 1. A typical P2P-based MoD system.

DHTs) are engaged for constructing and maintaining this overlay. While for the data overlay of a P2P-based MoD system, it is formed by the peers which are actually exchanging media data. As a peer would be able to cache in its buffer only limited media content, a data overlay connection could only be setup between two peers when there is media data exchange. In the data overlay, peers could use " push" or "gossip + pull" mechanisms to distribute and obtain the media data. Note that unlike the global structured index overlay, in the data overlay, only a few peers with media data exchanging could have links among themselves, and a cluster is formed by these peers. Usually several disconnected clusters may be formed among all the online peers, as shown in Fig. 1.

In a P2P-based MoD system, the streaming media object is split into segments, which is the smallest playable media content unit, and peers will use the segments as the unit for requesting and caching the media data. Typically, a peer in the P2P-based MoD system will try to pre-fetch and cache some segments which are immediately before its current playback offset in case of network jitters, where the available bandwidth might decrease unpredictably. As these pre-fetched segments are cached some time before their playback deadlines, the peer could ensure the quality of the streaming service under an unstable network environment to some degree. After a segment is played, the peer could either discard it immediately, or keep it in its buffer for some time. When requesting a segment, the peer will first try to obtain it from some other peers which have cached this segment in their buffers; if failed, it will then request the segment from the server. Obviously in such a peer-assisted mechanism, in order to improve the system's service capacity, it is essential to increase the "hit ratio" of the segment requests among the peers, especially when the server's service capacity is limited. In this paper, we present a novel data overlay for P2P-based MoD systems, in particular, we propose a probabilistic caching mechanism for peers to request and discard the segments. We evaluate our proposed mechanism via extensive theoretical analysis as well as simulation experiments, and show that by engaging our proposed mechanism, the following favorable properties could be achieved:

- Scalability: We find that the main performance bottleneck for a P2P-based MoD system is in the media server of the system, and is caused by the asynchronism and the sparsity of the online peers. Without enlarging the peers' buffering space, we show that our proposed caching mechanism could reduce the workload imposed on the server notably, or improve the quality of the streaming service conceived by peers considerably when the server's capacity is limited, thus improving the system's scalability.
- Flexibility: We provide means for system designers to make tradeoffs between the system's scalability and the peers' gossip cost in a P2P-based MoD system. Specifically, by varying the buffering space partition, a P2P-based MoD system could trade better scalability with larger peer gossip cost under the situation of sparse online peers or weak media server; or it could reduce the gossip cost of the peers at the price of sacrificing the system's scalability by imposing more workload on the server of the system.

We also study the performance of the proposed caching mechanism under a lossy network environment, and show that our approach will not cause serious performance degradation, even when not all the buffering space is allocated for pre-fetching; moreover, when the loss rates of the network links are extremely high, our mechanism works even better than the conventional continuous caching mechanism, which is usually considered to be more robust under the lossy network environment.

For the remainder of this paper, we discuss the related work in Section 2; we introduce our proposed caching mechanism in Section 3; and in Section 4, theoretical analysis on the system's performance is given; we present and discuss our simulation results in Section 5; and finally we conclude this paper and discuss the future work in Section 6.

2. Related work

With the advance of P2P technologies, many services which are traditionally provided with a client–server architecture are successfully migrated to P2P networks. For example, file swarming systems such as BitTorrent [2] are widely used on today's Internet, and have become the major source of the network traffic [19,20]; and P2P multicast systems such as ESM [3,4] and CoolStreaming [21] are successfully deployed to provide live media services without the deployment of an IP-multicast infrastructure. In recent years, many works [1,5–7,10, 11,13] have been proposed on providing on-demand media streaming services with P2P approaches.

For data overlays of the P2P-based MoD systems, oStream [1] uses a spanning tree algorithm for peers to construct an overlay for media streaming. Designs of tree-like overlays could also be found in P2Cast [5] and P^2VoD [6]: which differ in their caching strategies and failure recovery mechanisms. To make the system robust and to balance the workload, CoopNet [7] proposes to use a multiple tree overlay to distribute the MDC encoded media data. A tree-assisted mesh overlay is presented in [11], in which mesh links will be formed when the parent–child relationship could not be set up. In [8], a segment scheduling algorithm is proposed when network coding techniques [16,17] are engaged for the on-demand media distribution. In [9], the authors aim to support the on-demand video viewing functionality based on file swarming systems, and a probabilistic pre-fetching technique which has some similarity to our approach is proposed. However, the key difference between our work and [9] is that in our solution, peers are only caching very limited content in their memories, while peers in the system proposed in [9] actually download the entire video file on their hard disks. Moreover, we present a theoretical analysis in our work to enable a more insightful understanding on the probabilistic caching mechanism.

Besides data overlays, many index overlays have also been proposed recently for managing the MoD system. oStream [1] uses a set of centralized servers to record the peers' playback information. Pure distributed solutions such as DHT algorithms, which are widely used in file sharing applications, are also adopted for index overlay construction. In OBN [10], the finger table of Chord [22] is used for peers to maintain links to their neighbors on the overlay; and in [12], CAN [23] is adopted to organize the peers in a proxy-assisted on-demand streaming system. For supporting specific MoD operations such as jump, fast forward and rewind, some dedicated structures are proposed, such as the AVL tree [11] and the Skip List [13], the former is featured with non-sequential accesses and the latter supports the VCR-operations very well. And recently an overlay construction algorithm based on AVL tree is proposed in [8] for improving the searching efficiency by not involving all the peers on the overlay.

To understand the performance of the P2P-based MoD systems, analytical performance evaluation is an effective way. A stochastic process model is proposed in [14], which shows that the service capacity of peer-assisted MoD systems will start to scale after the transition time; and in [1], a modeling framework is used to study the bandwidth required on the server and the overhead engaged under different MoD system designs. In this paper, we adopt the modeling framework in [1] to analyze our proposed caching mechanism.

3. Probabilistic caching mechanism

3.1. The caching problem

Before describing the caching problem in P2P-based MoD systems, we first look at the size of the media objects and the buffering space size of the clients in a MoD system. With today's multimedia techniques, multimedia objects integrating audio and video are getting larger in order to have better audio/video effects. For example, a two-hour movie encoded with a CBR (constant bit rate) playback rate of 512 bps will have a size of 450 MB. Obviously, in an on-demand media application, it is impractical for today's personal computer with a typical sized memory (e.g., 512 MB) to cache the entire media object such as a movie in its memory. On the other hand, to cope with network jitters, where the future available bandwidth decreases unpredictably, many media streaming systems use a fixed length client buffer to pre-fetch and cache the media content ahead of their playback deadlines. For example, Windows Media Player allows a pre-fetching buffer up to 60 s, while Real Player uses a buffer with a default size of 30 s. Obviously, these pre-fetching buffers are relatively small compared with the computer's entire memory: for example, it takes less than 4 MB for caching 60 s of a 512 bps encoded video object, and caching 5 min of the video content requires 18.75 MB, which is a reasonable cost for today's personal computers. In summary, we believe that under today's multimedia techniques and with today's personal computers, peers are incapable of caching an entire media object or a large portion of it with their buffers, but they should be able to cache the media content which is more than just enough to handle network jitters.

With limited sized buffers, which segments a peer should cache is problematic. Following we investigate the caching problem formally. Suppose in a P2P-based MoD system, a media object is divided into M segments, indexed as 0 to $M-1$. Suppose at a time, there are N online peers accessing this object, and each peer has a buffer of B segments, and for a particular segment i ($0 \leqslant i < M$), we assume that it is cached on C_i different peers at the time. Now consider that a peer requests segment i, as it does not know on which peer this segment is cached, it just requests it blindly by sending its request to another peer with a probability p. Then for segment i, the probability that the request is answered by a peer caching it is $1-(1-p)^{C_i}$, and in this case, the request is successful. For all the M segments, our purpose is to maximize the probability of the successful segment requests among the online peers as $\sum_{i=0}^{M-1}(1-(1-p)^{C_i})$ given the buffering space B and the peer population N, or we seek to

$$\text{minimize} \quad \sum_{i=0}^{M-1}(1-p)^{C_i}$$

$$\text{s.t.} \quad \sum_{i=0}^{M-1} C_j \leqslant N \times B$$

It is not difficult to find that this optimization problem has a solution of $C_i = \frac{N\times B}{M}$, which means the segments should be cached evenly among the online peers. However, evenly caching the segments is impractical to implement in a P2P-based MoD system for two reasons: (1) It is hard for peers to know the global segment caching information, as segments are cached and discarded by peers very frequently all the time, thus, a peer can only make its decisions based on local information; (2) another purpose for peers to cache segments in their buffers is to cope with network jitters, which means peers must cache certain number of segments continuously ahead of their playback deadlines; as peers are asynchronous when accessing the media object, from a global view, their cached segments for network jitters are randomized instead of well organized to optimize the global segment availability. In summary, segments are unlikely to be cached evenly among all the online peers in P2P-based MoD systems.

On the other hand, we consider the situation that peers use a simple continuous caching scheme which is widely engaged in many systems to pre-fetch and cache the segments. Specifically, we suppose that each peer caches B segments which are immediately ahead of its current playback offset. For a particular segment, the probability that it is cached by one peer is $\frac{B}{M}$, and the probability that it is not cached by any of the total N online peers is $P = 1-(1-\frac{B}{M})^N$. In other words, given the peers' buffering space of size B, to achieve a segment availability among the online peers as P, the required number of the peers simultaneously online is

$$N = \log_{(1-\frac{B}{M})}(1-P) \tag{1}$$

Fig. 2 shows the relationship between the segment availability P among the online peers and the required peer number N, according to Eq. (1). From the figure, it is observed that a lot of peers are required to be online simultaneously for the system to maintain a high segment availability. For example, when peers have a 4-min buffer and the media object is two hours long, to have a segment availability of 0.98, more than 100 peers should be kept online at any time. Note that under this situation, the total buffering space on all the online peers

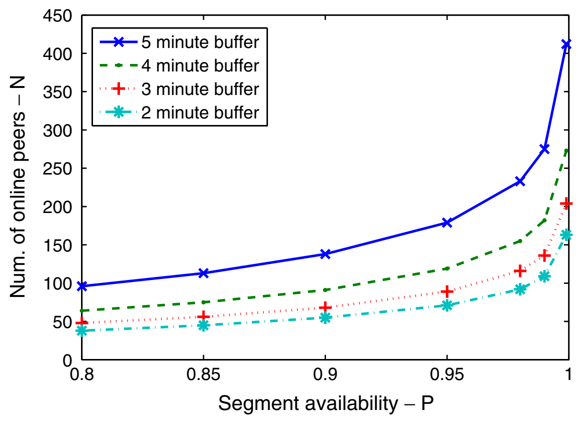

Fig. 2. Relationship between segment availability and required number of online peers, under the continuous caching scheme.

(more than 400 min) is much larger than the entire media object, meaning that if there is a practical evenly caching mechanism, the entire media objects could be served by the online peers completely. Moreover, we can see from the figure that when the target segment availability approaches one, the number of the online peers required increases rapidly to infinity. Obviously for a MoD system, it is hard to maintain too many peers online for just one media object, especially when this object is not very popular. When a segment request can not be answered by any online peer, the server will have to take the workload by providing the segment to the requesting peer. Clearly, under this continuous caching scheme, a server with limited service capacity is likely to become a bottleneck of the system.

In conclusion, we find that the optimized solution of evenly caching the segments cannot be practically implemented in P2P-based MoD systems due to peers' asynchronous behaviors and their incapability of having a global view; meanwhile, with the continuous caching scheme, the poor segment availability of the MoD system caused by the sparsity of the online peers will make the server a performance bottleneck, when its service capacity is limited. Based on this observation, we believe that a caching mechanism, between the extremes of evenly caching and continuous caching, which are practical to be implemented, could improve the system's segment availability and relieve the server's workload, and eventually improve the MoD system's performance effectively.

3.2. The probabilistic caching mechanism

In P2P-based MoD systems, peers cache segments for two reasons: First, segments which are immediately before a peer's playback offset are usually pre-fetched and cached by the peer to handle unpredictable network jitters; second, a peer could use the segments cached in its buffer to serve other peers which are requesting them. And we can see from Fig. 2 that by only caching enough segments (e.g. 2 min) for the first usage will cause a poor system wide segment availability, and from the discussion in the previous section, we find that it is also possible for peers to cache segments which are more than just enough against network jitters. Based on this observation, we propose a novel probabilistic caching mechanism for P2P-based MoD systems. Our simple idea is to let peers request and cache the segments probabilistically and cooperatively to improve the system's service capacity, while keeping enough segments pre-fetched before their playback offsets against network jitters. Our proposed caching mechanism is composed of a probabilistic caching scheme which determines the amount and the location of the segments cached, and a segment requesting and discarding algorithm. We will introduce them separately in the following sections.

3.2.1. The caching scheme

We demonstrate the probabilistic caching scheme in Fig. 3. In this scheme, we divide a peer's buffering space into two parts: a primary buffer with a size of B_P segments and a secondary buffering space with a size of B_S segments. The former is basically designed for handling network jitters, while the latter helps to improve the system's service capacity. Suppose a peer's current playback offset is at O, then for its primary buffer, its range is $[O, O + W_1)$, meaning that the peer pre-fetches and caches the segments within this range continuously. Obviously, W_1 segments will be cached in the primary buffer and $B_P = W_1$. As the objective for the primary buffer is to handle the network jitter, its size should be just enough for this purpose. For the remaining buffering space, it is allocated in a number of secondary buffers. We further divide the secondary buffering space into a series of forward secondary buffers and a series of backward secondary buffers. Counting from the peer's playback offset, we call these

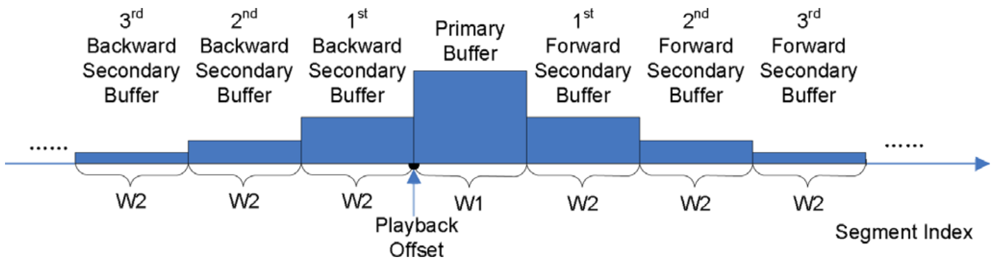
3rd Backward Secondary Buffer
2nd Backward Secondary Buffer
1st Backward Secondary Buffer
Primary Buffer
1st Forward Secondary Buffer
2nd Forward Secondary Buffer
3rd Forward Secondary Buffer
......
......
W2
W2
W2
W1
W2
W2
W2
Playback Offset
Segment Index

Fig. 3. Demonstration of the probabilistic caching scheme.

buffers as the 1st, 2nd,..., forward/backward secondary buffers respectively. As demonstrated in Fig. 3, the forward secondary buffer is composed of the 1st forward secondary buffer caching segments in the range of $[O + W_1, O + W_1 + W_2)$, the 2nd forward secondary buffer caching segments in the range of $[O + W_1 + W_2, O + W_1 + 2W_2), \ldots$, etc. For the ith forward secondary buffer with the range of $[O + W_1 + (i-1)W_2, O + W_1 + iW_2)$, the peer will cache a portion of ρ^i segments in this range, here ρ is a system parameter named caching ratio with a positive value smaller than one. For the backward secondary buffer, we have a similar scheme, in which the peer caches a portion of ρ^i segments in the range of the $(O - iW_2, O - (i-1)W_2]$ for its ith backward secondary buffer. Note that in the practical implementation, for the ith forward/backward secondary buffering range, actually a number of $\lceil \rho^i W_2 \rceil$ segments are cached, and the number of the secondary buffers is constrained by the total secondary buffering space B_S. For the choice of the secondary buffer length W_2, we let $W_2 = \lceil \frac{B_S(1-\rho)}{2\rho} \rceil$, so as to make $2\sum_{i=1}^{\infty} \rho^i W_2 \approx B_S$. The number of the secondary buffers n is determined by the equation of $2\sum_{i=1}^{n} \lceil \rho^i W_2 \rceil = B_S$. For example, suppose the total buffering space is 5 min (300 s), and each segment is a one-second content. If the primary buffer has $B_P = 120$ s, secondary buffering space is $B_S = 180$, and the caching ratio is $\rho = 0.5$, then the specific caching scheme is summarized in Table 1.

From Table 1, we can see that under the caching scheme, 45 of 90 segments should be cached in the first forward/backward secondary buffer; and 23 of 90 segments should be cached for the second forward/backward secondary buffer; and so on. The total secondary buffering space is used up with six forward/backward secondary buffers. From this example we can see that under the probabilistic caching scheme, although segments of 300 s are actually cached, a peer's buffer will cache segments within a range of 1200 s, and the chance of a segment being cached is determined by its distance to the peer's playback offset. On the other hand, as the secondary buffering space is limited, a peer will only have a few forward/backward secondary buffers. Therefore it will only need to be concerned with the system wide segment caching information within its caching range, which is usually much smaller than the entire media object, and could avoid the communication cost caused by obtaining the segment caching information on a large caching range.

Table 1
Example caching scheme

Buffer	Cached/range
Primary buffer	120/120
First forward/backward secondary buffer	45/90
Second forward/backward secondary buffer	23/90
Third forward/backward secondary buffer	12/90
Fourth forward/backward secondary buffer	6/90
Fifth forward/backward secondary buffer	3/90
Sixth forward/backward secondary buffer	1/90

3.2.2. Segment requesting and discarding algorithm

With the probabilistic caching scheme, we could describe the algorithm for peers to request and discard their segments. Our simple idea is that peers should request the least available segments and discard the most available ones. Before describing the algorithm formally, we first discuss the relationship among peers determined by their caching ranges.

In our probabilistic caching scheme, as a peer will cache the segments in its secondary buffers probabilistically, for a forward/backward secondary buffer, we use $W_2' = n \times W_2$ for the range that the secondary buffers cover on the segment index in one direction. For example, for the caching scheme described in Table 1, the range for the forward/backward secondary buffer is $6 \times 90 = 540$ s, as there are six secondary buffers and each covers 90 s. For any two peers, if one peer's entire caching range is overlapping with another peer's caching range, then they are neighbors to each other. Formally, for peer P_A with a playback offset at O_A, if peer P_B's playback offset O_B is within the range of $(O_A - W_1 - 2W_2', O_A + W_1 + 2W_2')$, then P_A and P_B are neighbors. Note that with this definition, the neighboring relationship between any two peers is bidirectional.

We assume an efficient index overlay (e.g., Skip List [13]) is available for peers to discover and maintain their neighboring peers, and for each peer, it manages to keep a set of its neighboring peers in the set of *Neighbor_Set*. Note that as all the peers have the same playback rate, the neighboring relationship between any two peers will not get changed unless one of them departs or performs some VCR-operations such as pause, fast forward, etc. Thus, it is inexpensive for the index overlay to maintain the neighboring relationships among the online peers. For each peer, it obtains its local caching information by exchanging gossips periodically with its neighbors, specifically, it will send a gossip message indicating the segments cached in its buffers to all its neighbors, and will receive the gossip messages from all its neighbors periodically. For constructing the gossip message, the bit-map scheme used in CoolStreaming [21] or a Bloom filter [24] could be engaged. After having a complete view on all the segments cached by its neighbors on its concerned range, which is the caching range of the primary buffer plus all the forward/backward secondary buffers, the peer will request and discard segments following the procedure below:

- The peer will request all the missing segments in its primary buffer from the neighboring peers which are caching them; if a segment is unavailable on the neighbors, it is requested from the server;
- For the ith forward secondary buffer, if the number of the actually cached segments are smaller than the regulated number $\lceil \rho^i W_2 \rceil$ of the probabilistic caching scheme, the peer will request the segments which are least cached by its neighbors from a neighboring peer caching it, until it caches exactly $\lceil \rho^i W_2 \rceil$ segments for this forward secondary buffer; however, if there is not enough segments to be requested for satisfying the number of $\lceil \rho^i W_2 \rceil$, the peer will not request from the server, but will just go to the $(i+1)$th forward secondary buffer;
- For the ith backward secondary buffer, if the number of the actually cached segments are larger than the regulated number $\lceil \rho^i W_2 \rceil$ of the probabilistic caching scheme, the peer will discard the segments which are most cached by its neighbors, until it caches exactly $\lceil \rho^i W_2 \rceil$ segments for this backward secondary buffer.

Fig. 4 presents the segment requesting and discarding algorithm in pseudo-code.

4. Performance analysis

4.1. Analytical methodology

We extend the analytical model in [1] to study the performance of the MoD system with our proposed probabilistic caching mechanism. Especially, we are interested in the workload imposed on the server of

```
Segment Requesting and Discarding Algorithm (Neighbor_Set)

1.  sends gossips to all the neighboring peers in Neighbor_Set;
2.  receives gossips from all the neighboring peers in Neighbor_Set;
3.  for each missing segment s in primary buffer
4.    if (s is cached by a neighboring peer $P_N$)
5.      requests s from $P_N$;
6.    if (s is not cached by any neighboring peer)
7.      requests s from server;
8.  for the $i^{th}$ forward secondary buffer
9.    while (the segment cached < $\lceil \rho^i W_2 \rceil$) and (segment available to request)
10.     requests the segment least cached by all the neighboring peers;
11. for the $i^{th}$ backward secondary buffer
12.   while (the segment cached > $\lceil \rho^i W_2 \rceil$)
13.     discards the segment most cached by all the neighboring peers;
```

Fig. 4. Segment requesting and discarding algorithm for the probabilistic caching mechanism.

the system, as the server is usually the performance bottleneck of the entire system. The analytical model is presented in the following.

We consider that under a P2P-based MoD system, for a particular random segment x, how frequently it must be served by the server. As in [1], we use X for the events that x is being requested, and denote its requesting rate as Λ_X. We use Z for the events that x is served by the server. Obviously, $\{Z\} \subseteq \{X\}$. We further use a random variable w to denote the interval length between two consecutive events in $\{X\}$, and use τ to denote the average length of the intervals between two consecutive events in $\{Z\}$. We let $E[\tau|x]$ be the conditional expectation of τ given x. If a media object contains T segments as $[0, T]$, then the total workload on the server could be calculated as

$$B = \int_0^T \frac{\mathrm{d}x}{E[\tau|x]}$$

We assume that peers request the media object following a Poisson process with an arrival rate of λ, and for a peer request, on average its session is of the length S. We further assume that each segment in the media object is requested with an equal chance. Then, the arrival rate for the events in $\{X\}$ could be expressed as $\Lambda_X = \frac{\lambda S}{T}$. With the Poisson process assumption, for w, the interval length between two consecutive events in $\{X\}$, its conditional distribution function is

$$F_w(t|x) = \Pr(w \leqslant t) = 1 - \mathrm{e}^{-t\Lambda_x}$$

and its conditional density function is

$$f_w(t|x) = \Lambda_x \mathrm{e}^{-t\Lambda_x}$$

With w's density function, it is easy to show that for a particular length W, the length expectation of all the intervals between two consecutive events in $\{X\}$ with their length no longer than W could be expressed as

$$E_{w\leqslant W}(w|x) = \frac{\int_0^W tf_w(t|x)}{\Pr(w \leqslant W)} \tag{2}$$

and the length expectation for all the intervals longer than W could be expressed as

$$E_{w>W}(w|x) = \frac{\int_W^\infty tf_w(t|x)}{\Pr(w > W)} \tag{3}$$

We now consider $E[\tau|x]$, which is the length expectation of the intervals between two consecutive events in $\{Z\}$, under the probabilistic caching mechanism.

For the segment x, when it is first requested from the server, it must be cached in the requesting peer's primary buffer, as there is no replica of it among the online peers. After being cached on the requesting peer, it will be discarded at some position of the peer's backward secondary buffers. If we assume that each segment is discarded with an equal chance, then it will be discarded at the 1st backward secondary buffer with a probability of $(1-\rho)$, it will be discarded at the 2nd backward secondary buffer with a probability of $\rho(1-\rho), \ldots$, and it will be discarded at the ith backward secondary buffer with a probability of $\rho^{i-1}(1-\rho)$. If we assume that a segment is discarded at the center of a backward secondary buffer, then on average, the segment will stay in a peer's backward secondary buffer for a time of

$$W_2'' = \sum_{i=1}^{\infty} \rho^{i-1}(1-\rho)\left(\frac{2i-1}{2}\right)W_2 = \frac{W_2}{1-\rho} - \frac{W_2}{2}$$

after it is played. Similarly, the time that the segment will stay in a peer's forward secondary buffer is also W_2'', as it could be requested in different forward secondary buffers with different probabilities.

Summarizing all the discussions, we can see that if x is requested by a peer from the server, on average it will stay for a time of $W_1 + W_2''$ in the peer's buffer, and if it is requested from another online peer, it will stay for a time of $W_1 + 2W_2''$. If we view the series of the requests that x is first requested by a peer from the server, and is passed from the caching peer to the next requesting peer repeatedly, until the time that x is lost among all the online peers and must be served by the server again, as a request chain. Then, the interval between the first two requests of the chain should be no longer than $W_1 + W_2''$, and if the chain is broken immediately after the first request, the interval between the last two consecutive requests of the chain should be longer than $W_1 + W_2''$; while for the case that the chain is successfully prolonged after the first request, then the intervals between any two following consecutive requests which successfully prolong the chain should not be longer than $W_1 + 2W_2''$, and the interval between the last two consecutive requests should be longer than $W_1 + 2W_2''$ so as to break the chain. Applying Eqs. (2) and (3), we have

$$E_1 = E_{w\leqslant W_1+W_2''}(w|x) = \frac{\int_0^{W_1+W_2''} tf_w(t|x)}{\Pr(w \leqslant W_1 + W_2'')}$$

$$E_2 = E_{w>W_1+W_2''}(w|x) = \frac{\int_{W_1+W_2''}^\infty tf_w(t|x)}{\Pr(w > W_1 + W_2'')}$$

$$E_3 = E_{w \leqslant W_1+2W''_2}(w|x) = \frac{\int_0^{W_1+2W''_2} t f_w(t|x)}{\Pr(w \leqslant W_1 + 2W''_2)}$$

$$E_4 = E_{w > W_1+2W''_2}(w|x) = \frac{\int_{W_1+2W''_2}^{\infty} t f_w(t|x)}{\Pr(w > W_1 + 2W''_2)}$$

for the expectations of different intervals' lengths.

Let $P_1 = \Pr(w \leqslant W_1 + W''_2)$ and $P_2 = \Pr(w \leqslant W_1 + 2W''_2)$, then we could obtain the length of the chain, which is also the length expectation of the intervals between two consecutive requests of x from the server, $E[\tau|x]$, as

$$E[\tau|x] = (1 - P_1)E_2 + P_1 \sum_{i=1}^{\infty} P_2^{i-1}(1 - P_2) \times (E_1 + (i-1)E_3 + E_4)$$

and the workload on the server could be expressed as

$$B(\lambda) = \int_0^T \frac{\mathrm{d}x}{E[\tau|x]} = \frac{\lambda S}{\mathrm{e}^{\lambda S(W_1+2W''_2)/T} - \mathrm{e}^{\lambda S W''_2/T} + 1} \quad (4)$$

It is also important to investigate the gossip cost engaged in our probabilistic caching mechanism. As neighboring peers exchange gossip messages on their segment caching information with each other regularly, the cost incurred for a peer should be proportional to the number of its neighboring peers. According to the neighboring condition, for a peer with the playback offset O, another peer with its playback offset in the range of $(O - W_1 - 2W'_2, O + W_1 + 2W'_2)$ will be its neighbor, then there are on average $\lambda(2W_1 + 4W'_2)S/T$ neighboring peers for it. If any two peers exchange gossip messages periodically, then the gossip cost for a peer in the MoD system with the probabilistic caching mechanism per gossip period is

$$O(\lambda) = \frac{\lambda(2W_1 + 4W'_2)S}{T} \quad (5)$$

4.2. Analytical results and discussions

We numerically study the MoD system's performance based on the analysis in this section. Fig. 5 presents the server's workload (in the unit of number of segments served per second) calculated by Eq. (4) under different peer arrival rate λ (in the unit of number of peers arrived per second). The media object has 7200 segments, and for each peer, its buffer size is 300 segments. If the segment is one second playable content, then the media object is two hours long and a peer could cache 5 min of content in its buffer. We use "a:b" to denote a caching scheme, in which the size of the primary buffer is a segments and the total size of all the secondary buffers is b segments. For example "300:0" is the continuous caching scheme. And for the schemes with $b > 0$, we set the caching ratio ρ as *0.5*, then the scheme denoted as "120:180" is actually the scheme we have described in Table 1. The average session length of peers to access the media object is 1187 s. The values of the system parameters are listed in Table 2.

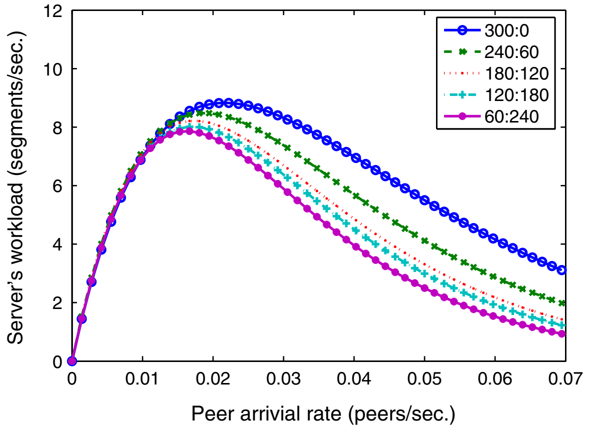

Fig. 5. Analytical results on server's workload and with varying peer arrival rates under different segment caching schemes.

From the figure we can see that clearly all the probabilistic caching schemes impose less workload on the server, and the more buffering space is allocated for the secondary buffers, the less workload the server will have, and consequently the better scalability could be achieved by the MoD system. Moreover, it is observed that all the curves have a peak regarding the server's workload, and after the peak, the server's burden will decrease with the increasing of the peers' arriving rate, meaning that the MoD system starts to scale and benefit from more users accessing the on-demand media object.

Table 2
Parameter values for numerical study

Parameter	Value
T	7200
S	1187
ρ	0.5
W_1, W_2, W'_2, W''_2	300, 0, 0, 0 (for scheme 300:0)
	240, 30, 150, 45 (for scheme 240:60)
	180, 60, 360, 90 (for scheme 180:120)
	120, 90, 540, 135 (for scheme 120:180)
	60, 120, 840, 180 (for scheme 60:240)

We call the arrival rate that the system reaches its peak as the system's turning point. From the figure, we can see that with more buffering space allocated for the secondary buffers, the MoD system's turning point appears at a lower peer arrival rate, meaning that the system could start to scale at a lower threshold.

Fig. 6 presents the gossip cost (in the unit of number of gossip messages per gossip interval, we assume that the gossip interval is 30 s) engaged by each peer in different caching schemes according to Eq. (5). Note that even under the continuous caching scheme of "300:0", there is still some gossip cost, as peers need to know the segments which are actually cached instead of supposed to be cached on their neighboring peers. Together with Fig. 5, we can see that to achieve a better scalability, i.e., less workload imposed on the server, the peers in the MoD system must maintain and communicate with more neighboring peers, at a higher gossip cost. Moreover, the analytical results in Figs. 5 and 6 show that by caching segments probabilistically, we provide a means to tradeoff the system's performance with the peers' gossip cost in designing P2P-based MoD systems.

Besides varying the buffering space partition "*a*:*b*", another tunable parameter in the probabilistic caching mechanism is the caching ratio ρ, which determines the portions of the segments cached in the secondary buffering ranges. In the following study, we consider the caching schemes with a fixed buffering space partition of "180:120", but with different values of ρ as 0.2, 0.5 and 0.8. The workload imposed on the server and the gossip cost for each peer are presented in Figs. 7 and 8 respectively.

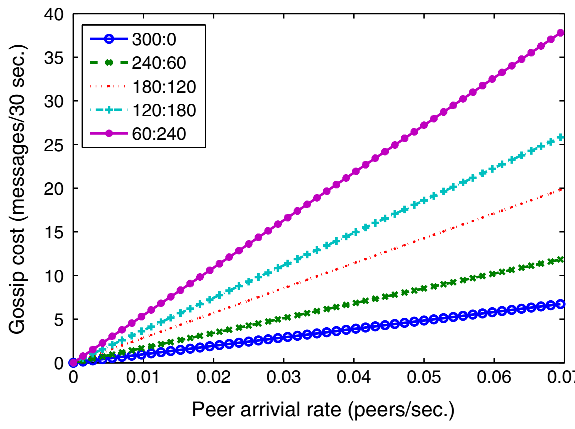

Fig. 6. Gossip cost per peer with varying peer arrival rates under different segment caching schemes.

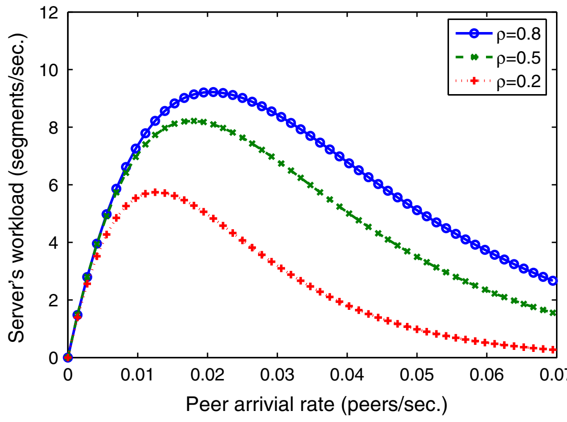

Fig. 7. Analytical results on server's workload and with varying peer arrival rates under different caching ratios.

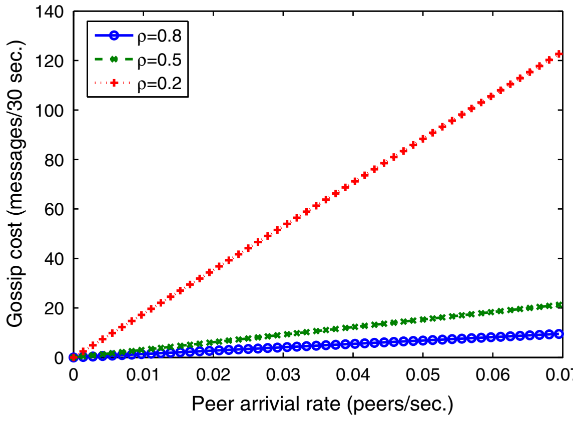

Fig. 8. Gossip cost per peer with varying peer arrival rates under different caching ratios.

From the figures we can see that with a smaller value of ρ, a better scalability could be achieved at the price of a larger gossip cost, as the peers under the probabilistic caching mechanism have a relatively larger buffering range on the segment index by having more secondary buffers; while with a larger valued ρ, the system works with a smaller peer gossip cost but imposing heavier workload on the server, for the reason that the peers cache the segments on a relatively smaller buffering range. For example, when $\rho = 0.2$, there are as many as 46 forward/backward secondary buffers on each direction; and with $\rho = 0.8$, each peer has only one forward secondary buffer and one backward secondary buffer. To avoid complexity, in this paper we fix the value of ρ as 0.5, and make the buffering space partition "*a*:*b*" as the only means to trade off the service capacity with the gossip cost in a P2P-based MoD system.

5. Performance evaluation

In this section, we use simulation-based experiments to evaluate the proposed probabilistic caching mechanism for P2P-based MoD systems. A time-driven simulator is developed for this purpose. The simulator operates in discrete intervals of time, where each interval is of the length equal to the time that one segment is played. We call the intervals the simulation seconds (referred to seconds for briefness). In our experiments, the media object served by the MoD system is CBR-encoded, and is of the length equal to 7200 s (i.e., 7200 segments). We allow each peer to have a buffering space of 300 s (i.e., 300 segments). In our simulator, a peer will try to play the segment at its individual playback offset in each second, if the segment is missing, a segment loss event is recorded for this peer, and the peer will play the next segment in the next second. Every 30 s, a peer will gossip with its neighboring peers, and requests/discards the segments following the algorithm presented in Fig. 4.

For the peers' behaviors, we assume that peers enter the MoD system and access the media object following a Poisson process, and their beginning offsets are uniformly randomized between 0 and 7200 on the segment index. For the lengths of the sessions, we assume that each peer will access the media object for a random period of time, and the session lengths follow a Pareto distribution reported in [15]. We generate a synthetic trace for the events of peer joinings and departures under the assumed models, and use the trace as the input to the simulator. The average session length for a peer to access the media derived from the trace is approximately 1187 s. We do not explicitly simulate the VCR jumps as it could be viewed as a departure followed by a rejoining of the peer. Finally, as mentioned in previous sections, we assume an efficient index overlay (e.g., Skip List [13]) is available for peers to discover and maintain their neighboring peers.

5.1. Experimental results

Fig. 9a presents the workload of the server for the simulated MoD system with different caching schemes engaged and under varying peer arrival rates. We also use the denotation of "*a*:*b*" for different caching schemes as in the previous section. It could be observed that the simulated results in the figure has the same trends with the analytical results in Fig. 5, with the following features: (1) The probabilistic caching schemes impose less workload on the server, and the more buffering space is allocated for the secondary buffers, the less workload the server will have; (2) with more buffering space allocated for the secondary buffers, the MoD system's turning point appears at a lower peer arrival rate. However, we find that the simulation results in Fig. 9a are obviously better than the analytical ones in Fig. 5, with less workload on the server observed. We believe this could be explained as in our analysis, we do not consider the fact that peers tend to request the least available segments and discard the most available ones, but simply assume that all the segments are cached with an equal chance. To validate our point, we run the simulation again under the same settings, but this time the peers request and discard segments just randomly. The experimental outputs are plotted in Fig. 9b. From the figure, we can see that the benefits brought by simply expanding peers' caching range are very lim-

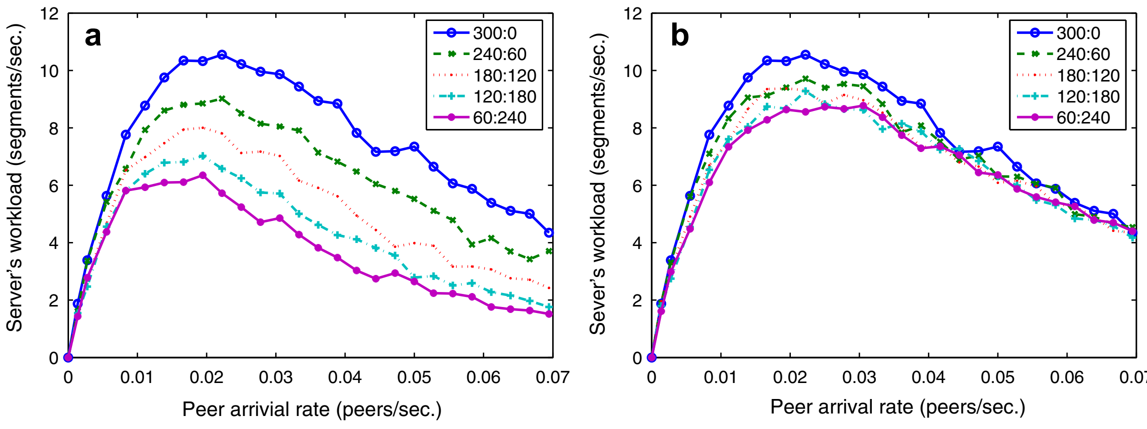

Fig. 9. Simulation results on server's workload with varying peer arrival rates under different segment caching schemes.

ited, while the algorithm of requesting/discarding the least/most available segments is essential in our caching mechanism for P2P-Based MoD systems.

Fig. 10 presents the numbers of the gossip messages a peer sends out per 30 s with different caching schemes. We can see that the simulation results on the gossip cost are very close to the analytical results in Fig. 6. In conclusion, we have verified our analysis in the previous section and have illustrated improved performance regarding the server's workload and the system's scalability, and we have also demonstrated the inherent tradeoff between the system's performance and the peers' gossip cost via simulation.

In previous experiments we assume that the capacity of the server in the MoD system is unlimited, however, usually a MoD server could only serve a limited number of requests simultaneously. In this experiment, we investigate the quality of the service received by peers when the server's capacity is limited in the simulated MoD system. We assume that a server could only upload a limited number of segments per second, and measure the segment missing ratio, which is defined as the ratio of the segments which are missed on their playback deadlines, compared with all the segments requested. As we let the peers' uploading/downloading capacities to be unlimited in the system, the only reason that a segment is missed lies in the fact that it is not cached among the online peers, and the server is not able to serve it as its service capacity is fulfilled. Fig. 11 presents the segment missing ratios of the peers in the MoD system under different caching schemes with varying capacities of the server,

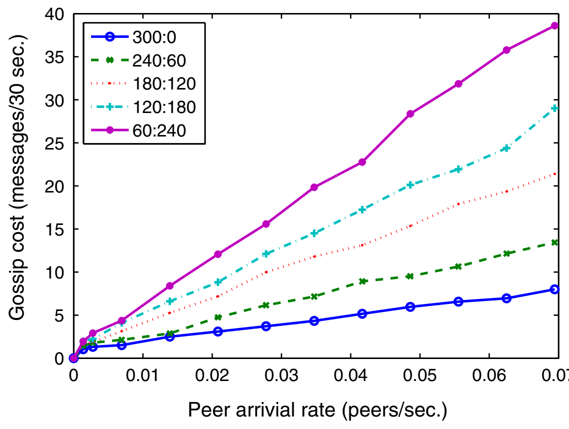

Fig. 10. Simulation results on gossip cost per peer with varying peer arrival rates under different segment caching schemes.

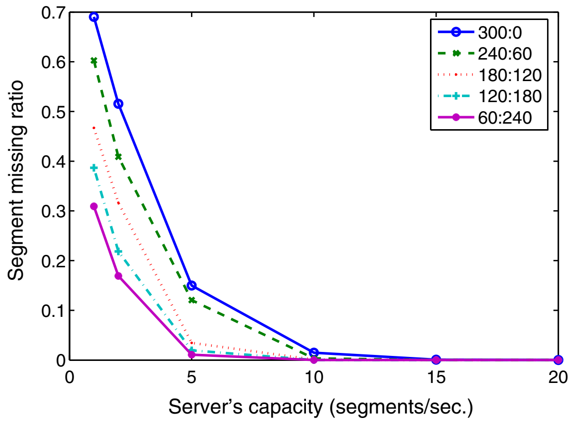

Fig. 11. Segment missing ratio with varying server's capacities under different segment caching schemes.

which is denoted as the number of the segments it could upload per second. The peers' arrival rate is fixed as 0.03 s^{-1}. It is observed from the figure that a considerable portion of the segments requested are missing due to the very limited service capacity of the server, but with the increase of the server's capacity, the missing ratio decreases rapidly under any caching scheme. Moreover, we find that for the caching schemes allocating some buffering space for probabilistic caching, the segment missing ratios are lower than the continuous scheme of "300:0", and the more space is allocated for probabilistic caching, the lower segment missing ratio could be achieved.

Our last experiment focuses on the performance of the simulated MoD system when the underlying network is lossy. As we have discussed before, continuously pre-fetching and caching segments before a peer's playback offset is an effective way to handle unpredictable network jitters, thus the "300:0" scheme using all the buffering space as the primary buffer is expected to have a better performance than the schemes with secondary buffers and caching segments probabilistically. On the other hand, by caching segments in the forward secondary buffers, a peer is allowed to request segments with an increasing probability from the furthest forward secondary buffer to its primary buffer, and has relatively more chances to request one segment if it is lost in the previous downloading. In this way, probabilistic caching schemes should perform better than a continuous one under lossy network environments. To study how the two different factors influence the MoD system, in our experiment, we deploy the MoD system on a simulated network topology

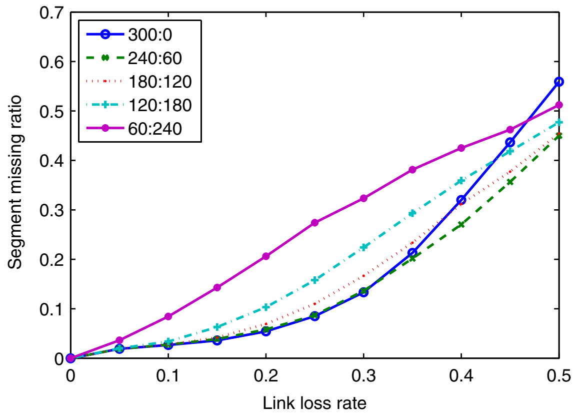

Fig. 12. Segment missing ratio with varying link loss rates under different segment caching schemes.

created by the GT-ITM transit-stub model [25]. We deploy the server and all the peers on stub domains, and vary the loss rate of the links to simulate a lossy network environment. We present the results in Fig. 12. It is interesting to see from the figure that the continuous caching scheme behaves differently compared to the probabilistic schemes: when the link loss rate is low, the continuous scheme outperforms all the probabilistic schemes, but it has a higher segment missing ratio than the probabilistic caching schemes when the loss rate is high enough. Our observation indicates that when the network is not very lossy, the continuous scheme is better regarding the segment missing ratio due to its larger primary buffer, but when the link loss rate is high enough to overwhelm the benefit caused by prefetching in the primary buffer, some probabilistic schemes perform better because peers have more chances to download a segment which was unsuccessful in its previous request. Moreover, we can find that there is no obvious performance gap between the "300:0" scheme and the "240:60" or the "180:120" schemes even under low link loss rates, indicating that by engaging a probabilistic caching scheme with a proper buffering space partition, there is just trivial performance degradation regarding the segment missing ratio under lossy network environments for P2P-based MoD systems.

5.2. Discussions

The experimental results indicate that our proposed caching mechanism for P2P-based MoD systems could reduce the server's workload, thus improving the system scalability greatly, at the cost of the gossip communication among the peers. However, we believe that the gossiping cost is trivial for the following reasons: (1) a peer only exchanges gossip messages with its neighboring peers, which is much fewer than the total online peers; (2) a gossip message is much smaller in size compared with a media segment, and for the secondary buffers, delay is tolerable as the segments requested in them are not very urgent. For example, suppose we use one bit for the existence of a segment in a peer's buffer, then under our simulation setting, the gossip message could be as small as 300 bits, which is sent out every 30 s.

Another related issue is the selfish behaviors under the P2P-based MoD applications: a selfish peer may avoid uploading the cached media data by declaring in its gossip messages with very few segments cached. However, it is easy to detect those peers which are frequently declaring obviously fewer segments than they are supposed to cache according to the caching scheme, and other peers may choose not to upload to these selfish lying peers any more. Another possible selfish behavior is that a selfish peer simply does not upload the requested segments, although it has declared to hold them in its gossip message. To combat this behavior, we propose that a peer will keep a record of the successful/failed segment requests for each of its neighboring peers. If the ratio between the two numbers for a neighboring peer is lower than a threshold, this neighboring peer may be believed as selfish and other peers may choose not to upload segments to it any more.

6. Conclusion and future work

Peer-assisted application-level overlay is a promising direction to deploy the interactive on-demand media services on the Internet. However, the scalability of the P2P-based MoD systems are usually constrained by the bottleneck on the server, due to the asynchronism and the sparsity of the online peers accessing the same media object. To improve the system's scalability, we propose a novel probabilistic caching mechanism for MoD systems on P2P networks in this paper. We show via theoretical analysis as well as experimental studies that without enlarging the buffering space on clients, the MoD system with our proposed caching mechanism imposes less workload on the server, thus having a better scalability compared with the system engaging the conventional continuous caching mechanism. Moreover, by adjusting the buffering space partition, we provide means for people to tradeoff the scalability of the system with the peers' gossip cost, thus making the mechanism flexible under various application contexts. We also show that by engaging our proposed caching mechanism, a higher quality of service could be conceived by peers when the server's capacity is limited, and our proposed caching mechanism will not degrade the service quality under lossy network environments.

For future work, as we have discussed in Section 5.2, a mechanism is necessary to be integrated into the P2P-based MoD systems against selfish peers. Furthermore, our proposed probabilistic caching mechanism could be improved from the following aspects: First, recall that when the server's service capacity is limited, usually it is not able to upload all the segments requested by the peers, thus it is necessary for the server to have a scheduling mechanism for determining which request should be responded with priority. Generally speaking, the server should upload the segments which are likely to be forwarded among the peers for the most times, in order to lower the server's workload. Another direction is to investigate the feasibility of incorporating the network coding technique with our caching mechanism. Although with network coding, higher throughput is expected, other issues arise, such as the segment deadline problem and the malicious block problem addressed in [8,18] respectively. Finally, we plan to validate the proposed probabilistic caching mechanism under real-world deployments.

References

[1] Y. Cui, B. Li, K. Nahrstedt, oStream: asynchronous streaming multicast in application-layer overlay networks, IEEE Journal on Selected Areas of Communications 22 (1) (2004) 91–106.

[2] BitTorrent. <http://www.bittorrent.com>.

[3] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, IEEE Journal on Selected Areas in Communication 20(8) (8) (2002) 1456–1471.

[4] Y. Chu, A. Ganjam, T.S. Ng, S.G. Rao, K. Sripanidkulchai, J. Zhan, Hui Zhang, Early experience with an Internet broadcast system based on overlay multicast, in: Proceedings of USENIX'04, Boston, MA, June 2004.

[5] Y. Guo, K. Suh, J. Kurose, D. Towsley, P2Cast: Peer-to-peer patching scheme for VoD service, in: Proceedings of WWW'03, Budapest, Hungary, May 2003.

[6] T.T. Do, K.A. Hua, M.A. Tantaoui, P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment, in: Proceedings of ICC'04, Paris, France, June 2004.

[7] V.N. Padmanabhan, H.J. Wang, P.A. Chou, K. Sripanidkulchai, Distributing streaming media content using cooperative networking, in: Proceedings of NOSSDAV'02, Miami, FL, May 2002.

[8] H. Chi, Q. Zhang, Efficient search and scheduling in P2P-based media-on-demand streaming service, IEEE Journal on Selected Areas of Communications 25 (1) (2007) 119–130.

[9] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, P. Rodriguez, Is high-quality vod feasible using p2p swarming? in: Proceedings of WWW'07, Banff, Alberta, Canada, May, 2007.

[10] C.S. Liao, W.H. Sun, C.T. King, H.C. Hsiao, OBN: peering for finding suppliers in P2P on-demand streaming systems, in: Proceedings of ICPADS'06, Minneapolis, MN, July 2006.

[11] M. Zhou, J. Liu, Tree-assisted gossiping for overlay video distribution, Multimedia Tools and Applications 29 (3) (2006) 211–232.

[12] L. Guo, S. Chen, X. Zhang, Design and evaluation of a scalable and reliable P2P assisted proxy for on-demand streaming media delivery, IEEE Transactions on Knowledge and Data Engineering 18 (5) (2006) 669–682.

[13] D. Wang, J. Liu, Peer to peer asynchronous video streaming using skip list, in: Proceedings of ICME'06, Toronto, Canada, July 2006.

[14] Y. Tu, J. Sun, M. Hefeeda, Y. Xia, S. Prabhakar, An analytical study of peer-to-peer media streaming systems, ACM Transactions on Multimedia Computing, Communications, and Applications 1 (4) (2005) 354–376.

[15] H. Yu, D. Zheng, B.Y. Zhao, W. Zheng, Understanding user behavior in large scale video-on-demand systems, in: Proceedings of EuroSys'06, Leuven, Belgium, April 2006.

[16] R. Ahlswede, N. Cai, S.R. Li, R.W. Yeung, Network information flow, IEEE Transactions on Information Theory 46 (4) (2000) 1204–1216.

[17] C. Gkantsidis, P. Rodriguez, Network coding for large scale content distribution, in: Proceedings of the IEEE INFOCOM'05, Miami, FL, March 2005.

[18] C. Gkantsidis, P. Rodriguez, Cooperative security for network coding file distribution, in: Proceedings of the IEEE INFOCOM'06, Barcelona, Spain, April 2006.

[19] A. Parker, The true picture of peer-to-peer file-sharing, CacheLogic Presentation, CacheLogic, 2004.

[20] K. Cho, K. Fukuda, H. Esaki, A. Kato, The impact and implications of the growth in residential user-to-user traffic, in: Proceedings of ACM SIGCOMM'06, Pisa, Itlay, Sepember 2006.

[21] X. Zhang, J. Liu, B. Li, T.-S.P. Yum, CoolStreaming/DONet: a data-driven overlay network for live media streaming, in: Proceedings of IEEE INFOCOM'05, Miami, FL, March 2005.

[22] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for Internet applications, in: Proceedings of ACM SIGCOMM'01, San Deigo, CA, August 2001.

[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: Proceedings of ACM SIGCOMM'01, San Deigo, CA, August 2001.

[24] B. Bloom, Space/time tradeoffs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422–426.

[25] E.W. Zegura, K. Calvert, S. Bhattacharjee, How to model an internetwork, in: Proceedings of IEEE INFOCOM '96, San Francisco, CA, March 1996.

Ye Tian is a Ph.D. candidate in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include peer-to-peer networks and distributed systems, especially P2P-based content distribution networks.

Di Wu is a Ph.D. candidate with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests are in the areas of peer-to-peer systems and overlay networks.

Kam-Wing Ng is currently a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include P2P networks and grid computing.