ARTICLE IN PRESS

Computer Communications xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Peer-to-peer video-on-demand with scalable video coding

Yan Ding[a], Jiangchuan Liu[a], Dan Wang[b], Hongbo Jiang[c,*]

[a] *School of Computing Science, Simon Fraser University, Burnaby, BC, Canada*
[b] *Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*
[c] *Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China*

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Peer-to-peer
Video-on-demand
Scalable video coding
Network heterogeneity
Scheduling

ABSTRACT

Peer-to-peer has evolved into a promising communication paradigm for large-scale content sharing. It has recently been suggested for on-demand video streaming as well. Such peer-to-peer Video-on-Demand (VoD) makes effective use of local disk space and upload bandwidth distributed across peers to relieve the server load, which has long been a bottleneck of conventional VoD systems that demand enormous storage and network resources.

It is well-known that the startup delay of state-of-the-art peer-to-peer VoD remains much longer than powerful client/server-based systems. In this paper, we present a novel peer-to-peer VoD system that utilizes Scalable Video Coding (SVC) for delay minimization, and to deal with heterogeneous user capabilities as well as dynamic end-to-end resources availability. It brings two tangible benefits: first, starting from the SVC base layer only, the startup delay for a peer to join the system and successfully initialize video playback can be reduced; second, by dynamically adding or dropping SVC enhancement layers, the occurrences of frame freezing due to the temporal network congestion or the insufficient peer bandwidths can be minimized, and quick recovery from such freezing can be expected too. We mathematically formulate the transmission scheduling problem for peer-to-peer VoD with SVC. We strike a balance between startup delay and playback quality, trying to maximize the overall playback quality that all peers experience. We develop a practical scheduling strategy that allows each peer operate locally and efficiently. It implements a zigzag like importance allocation mechanism to determine the transmission order, taking advantage of the supplying peers with more layers and larger bandwidth.

We have extensively evaluated our proposed system under diverse network and peer configurations. Our simulation results demonstrate that it makes effective utilization of network resources, and outperforms conventional P2P VoD systems in terms of startup delay and playback quality.

1. Introduction

Peer-to-peer has evolved into a promising communication paradigm for large-scale content sharing. With each peer contributing its bandwidth to serve others, a peer-to-peer overlay scales extremely well with larger user bases. Besides file sharing, it has been quite successful in supporting large-scale live streaming (e.g., CoolStreaming [1], PPLive [2], and UUSee [3]), and has recently been suggested for on-demand video streaming (e.g., GridCast [4] and Vanderbilt VoD [5]) as well. Such peer-to-peer Video-on-Demand (VoD) makes effective use of local disk space and upload bandwidth distributed across peers to relieve the server load, which has long been a bottleneck of conventional VoD systems that demand enormous storage and network resources.

Compared with file sharing and live streaming, VoD generally supports richer user interactions, and users switch among videos more frequently. Peer-to-peer VoD thus faces much more challenges, and existing solutions are yet to be perfected to compete with the conventional client/server model. In particular, it is well-known that the startup delay of state-of-the-art peer-to-peer VoD remains much longer than powerful client/server-based systems, particularly those with Content Distribution Network (CDN) support. For example, for the on-demand mode in PPLive, the startup delays are often longer than 10 s, and sometimes go beyond half a minute, while the average delay of YouTube is about 6.5 s [6], though the latter has already been ranked as a slow site. The situation is only getting worse when rich VCR operations such as fast forward, rewind, and random seek are introduced.

There have been pioneer works on constructing efficient overlays for peer-to-peer VoD [7–9]. They optimize the indexing structures for organizing the peers of similar playback progresses, so as to speed up the video segment discovery. Yet the video buffering time also constitutes an important part in the startup delay, which

* Corresponding author.
E-mail addresses: yda15@cs.sfu.ca (Y. Ding), jcliu@cs.sfu.ca (J. Liu), csdwang@comp.polyu.edu.hk (D. Wang), hongbojiang@hust.edu.cn (H. Jiang).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.04.025

can hardly be minimized even with an advanced indexing scheme. This is mainly because, in the existing systems, the video stream structures are not flexible and generally with a fixed rate while users have heterogenous capacities, i.e., the upload and download bandwidths. We believe that this can be improved by enabling adaptive rate control during seeking within a video or switching between videos. A key observation is that, in practice, a quick start with a reasonable quality enables better user experience than a slow start, even with high picture quality. This is particularly true for VoD, as users frequently switch among different videos, and poor experience often leads to even more frequent switching.

To this end, we present a novel peer-to-peer VoD system that utilizes Scalable Video Coding (SVC). The SVC codec in H.264/MPEG-4 AVC [10] enables rate adaptation in terms of temporal, spatial, and SNR/quality salabilities. A video stream can be encoded into a base layer and several enhancement layers, which progressively improve the reconstructed video quality. It brings two tangible benefits to peer-to-peer VoD: First, if starting from the base layer only, the startup delay for a peer to join the system and successfully initialize video playback can be reduced; Second, by dynamically adding or dropping layers, the occurrences of frame freezing due to temporal network congestion or the insufficient peer bandwidths can be minimized, and quick recovery from such freezing can be expected too. The rate adaption however faces a series of challenges. First, though fewer segments are to be delivered with lower layers only, a user generally expects higher quality video as soon as possible if it does have enough bandwidth capacity. Also given the layer dependency, different layers are of uneven importance, i.e., of different weights. Such dependency must be considered in the calculation of a streaming schedule, beyond the bandwidth and time constraints in existing systems.

In this paper, we mathematically formulate the transmission scheduling problem for peer-to-peer VoD with SVC. We strike a balance between startup delay and playback quality, trying to maximize the overall playback quality that all peers experience, that is, the total weight value of all the received data segments before the playback deadline. This scheduling problem, in its general form, is difficult even with global knowledge. Also peers may join and leave frequently and their buffer status thus would change quickly as well. Therefore, we resort to a practical scheduling that allows each peer operate locally and efficiently. It implements a zigzag like importance allocation mechanism to determine the transmission order, taking advantage of the supplying peers with more layers and larger bandwidth.

We have extensively evaluated our proposed system under diverse network and peer configurations. Our simulation results demonstrate that by using SVC, our system can efficiently deal with the network congestion and the heterogenous network resources, and ensure satisfactory service for every user in terms of startup delay and playback quality.

The rest of this paper is organized as follows. Section 2 presents the related work. In Section 3, we overview the system design. Section 4 formulates the scheduling problem with scalable video and identifies its difficulty. We then present an efficient local scheduling algorithm using zigzag ordering in Section 5. The performance of our solution is evaluated in Section 6. Finally, Section 7 concludes the paper.

2. Related work

Peer-to-peer streaming has attracted significant attention both from academia and industry in the past decade; see a survey in [11]. It has also recently been applied in video-on-demand services [12–15,7–9]. There have been great efforts in constructing efficient overlays for peer-to-peer VoD, e.g., RINDY [7], DSL [8], and InstantLeap [9], to name but a few. They optimize the indexing structures for organizing the peers of similar playback progresses, so as to speed up video segment discovery. Our work complements them by introducing adaptive scalable video to minimize playback delay, which has long been a critical challenge for peer-to-peer streaming systems.

There have been some existing works using adaptive or layered video coding for peer-to-peer streaming. Cui and Nahrstedt [16] employ layered video to accommodate asynchronous requests from users of heterogenous bandwidths. Liu et al. [17] design a distributed protocol that allows peers enjoy better video quality if they contribute more uplink bandwidth. Baccichet et al. [18] develop a mathematical framework to quantify the advantage of using scalable codec for tree-based overlays, particularly during network congestions. Rejaie et al. [19,20] present an adaptive layered streaming mechanism to enable quality adaptive streaming of a layer-encoded video from multiple congestion controlled senders. We however consider a more general mesh overlay that has been widely used in today's commercial products. Our focus also differs from these previous studies in that, besides accommodating bandwidth heterogeneity, the role of scalable video in reducing playback delay is explicitly considered. In particular, we present an efficient scheduling strategy that works with scalable video to minimize the delay and delay jitter during playback.

Diverse scheduling strategies for peer-to-peer streaming have been examined in the literature. Typical mesh protocols implement a hybrid earliest deadline first and rarest first policy for segment fetching in a sliding window. Bandwidth heterogeneity has also been considered, with preference to high bandwidth supplying peers [1,4]. Recently, RedCarpet [21] further addresses joint server and client scheduling. Our work is motivated by these studies. Yet, we not only consider network-level measures, but also perceptual quality measures to ensure satisfactory service for every user. With prioritized video layers, our scheduling strategy leverages peers with rich available resources, so as to provide the best possible playback quality and minimized delay to all peers.

3. System overview

We consider a Video-on-Demand (VoD) system that consists of a video server and a number of clients interested in different videos originated from the server. Different from the traditional client/server model, our system implements the peer-to-peer communication model; that is, each client not only downloads its video of interest, but can also forward the video to other clients interested in the same video.

We advocate a *mesh* overlay design as used in many model peer-to-peer streaming systems [1,22]. Each new peer, when joining the system, issues a request to the server, and will be informed with a list of active peers interested in the same video. The new peer will randomly select some of them (about 8–10, as suggested by previous studies [23]) to establish partnerships. It will then exchange bandwidth and data availability information with these partners and fetch video data through a scheduler. The active peer list will be periodically gossiped through the overlay, so that existing peers can update their partners to accommodate peer dynamics and possibly achieve better performance.

Note that in a VoD system, a peer can join at any time. Minimizing the startup delay thus becomes a critical concern. We address this problem through the incorporation of Scalable Video Coding (SVC), which also better accommodates peer heterogeneity and reduces playback delays. We first give an overview of SVC, and then detail the benefits and challenges toward incorporating SVC in peer-to-peer VoD.

3.1. Scalable video coding

In H.264 AVC, a video stream can be encoded into a base layer and several enhancement layers, known as Scalable Video Coding (SVC). Each encoded layer will be further divided into segments, each with one unit playback time (Fig. 1). As such, a layered segment (*LS*) should be represented by two identifiers: its layer number and its sequence number. The layer number starts from 1, the base layer, and the sequence number gives the time instance (in unit time) when this segment should be played. For example, LS_{23} represents the 3rd segment in layer 2, i.e., the first enhancement layer. Given the layer dependencies, higher layers can only be decoded if all lower layers are received. Hence, at a time instance, the reconstructed video quality is determined by how many cumulative *LS*s starting from the base layer are received.

SVC codec enables rate adaptation in terms of temporal, spatial, and SNR/quality salabilities. Fig. 2 shows a simple example of how SVC works, which has two quality layers (one base layer and one enhancement layer) with a GOP (Group of Pictures) size of 4. This example also realizes temporal scalability by introducing B-directional predicted frames,where frames 2 and 6 forms the first level of temporal enhancement and frames 1, 3, 5 and 7 form the second level. The spatial scalability can be further added by changing the resolution between layers, e.g., by over-sampling frames in higher layers using resolution formats such as QCIF, CIF and 4CIF. GOP is the basic unit that can be decoded individually. To eliminate prediction dependency among segments in the same layer, we divide a video stream into segments, each of which consists of an integer number of GOPs (Fig. 2). For example, for a typical video with frame rate of 30 frames/s, we use a GOP size of 15 frames and therefore, each an 1-s *LS* will contain two GOPs.

3.2. Window adjustment

We assume that each peer maintains a *sliding window* that indicates the segments currently of interest in its local buffer. The width of the sliding window is fixed, e.g., Fig. 3 illustrates a window of five segments. The height of the window however various over time, ranging from base layer only to all the layers. When a peer just joins the system, the height of the window will be initialized to an initial value, for example, only 1, i.e., the base layer. This would reduce the startup time for less data is to be fetched. Similarly to the TCP AIMD congestion control mechanism [24], the window height will then increase linearly to explore the available capacity of the peer, until all layers are included (Fig. 4: Case 1) or certain layers cannot be fetched without affecting lower layers. During network congestion or when the peer bandwidths are insufficient, the height can also be reduced (Fig. 4: Case 2 and 3), so as to ensure the lower important layers can be successfully received. In short, there is a dynamic window height adjustment mechanism that tries to meet the segment arrival curve (see Fig. 5).

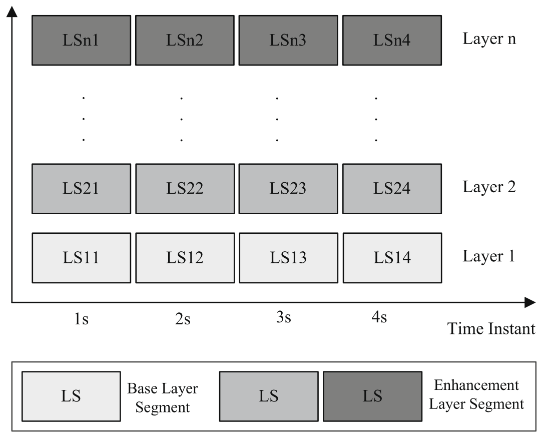

Fig. 1. Layered segment in video streams.

The window also slides along with the playback progress of the peer. For each time instance, the peer, after obtaining the segment availability in the windows of its partners and their bandwidths, will decide a segment fetching schedule; that is, which segment in which layer should be transmitted from which partner, and their orders. We will discuss the optimal strategy that best balances reconstructed video quality and playback delay in the next section.

4. Scheduling with SVC: problem formulation

The use of SVC brings two tangible benefits: first, starting from the base layer only, the startup delay for a peer to join the system and successfully initialize video playback can be reduced; Second, by dynamic adding or dropping layers, the occurrences of frame freezing due to temporal network congestion or the insufficient peer bandwidths can be minimized, and quick recovery from such freezing can be expected too. It is known that a quick start with a reasonable quality enables better user experience than a slow start, even with high picture quality [25], and frame freezing has been the most frequent complaint for Internet video streaming. These are particularly true for VoD, as users frequently switch among different videos, and poor experience often leads to even more frequent switching.

The rate adjustment however faces a series of challenges. First, though fewer segments are to be delivered with lower layers only, a user generally expects higher quality video as soon as possible if it does have enough bandwidth capacity. Also given the layer dependency, different layers are of uneven importance, i.e., of different weights. Such dependency must be considered in the calculation of a schedule, beyond the bandwidth and time constraints in the existing systems. In short, given the heterogeneous bandwidths and segment availability, our objective is to achieve the best overall video quality by maximizing the number of weighted layered segments.

We now formulate this scheduling problem with SVC. Suppose there are N active peers in the overlay, $P=\{P_1,P_2,\ldots,P_N\}$, and their corresponding upload and download bandwidths are $U=\{U_1, U_2,\ldots,U_N\}$ and $D=\{D_1,D_2,\ldots,D_N\}$, respectively. We use $a_{ij}=1$ ($i=1,2,\ldots,N$, $j=1,2,\ldots,M$) (M is the total number of segments of the video) to indicate whether P_i has segment j in its window ($a_{ij}=0$ means it does not). For each P_k ($k=1,2,\ldots,N$), h_j^k represents whether P_k needs segment j ($h_j^k=1$ means it does). We also use variable x_{ij}^k={0,1} to indicate whether P_k will fetch segment j from P_i ($x_{ij}^k=1$ represents P_k will).

Let $q(j,k)$ be the weight of segment j for peer P_k. The optimal scheduling problem is to determine the receiving (or sending) matrix $X=\{x_{ij}^k\}$, as follows:

$$\text{Maximize} \quad \sum_{k=1}^{N} q(j,k)\cdot x_{ij}^k \tag{1}$$

$$\text{s.t.} \quad x_{ij}^k \leqslant \min\{a_{ij}, h_j^k\} \quad \forall i,j,k \tag{2}$$

$$\sum_{i=1}^{N} x_{ij}^k \leqslant 1 \quad \forall j,k \tag{3}$$

$$\sum_{j=1}^{M}\sum_{i=1}^{N} x_{ij}^k \leqslant D_k \quad \forall k \tag{4}$$

$$\sum_{k=1}^{N}\sum_{j=1}^{M} x_{ij}^k \leqslant U_i \quad \forall i \tag{5}$$

$$x_{ij}^k = \{0,1\} \quad \forall i,j,k \tag{6}$$

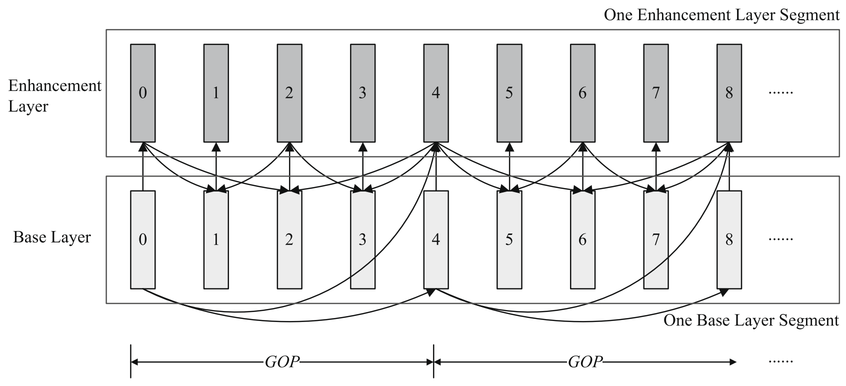

Fig. 2. Frame structure in SVC with two quality layers and GOP size of 4.

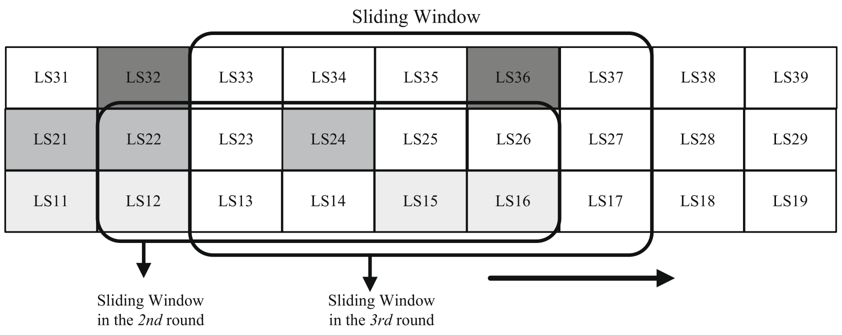
Sliding Window
LS31 LS32 LS33 LS34 LS35 LS36 LS37 LS38 LS39
LS21 LS22 LS23 LS24 LS25 LS26 LS27 LS28 LS29
LS11 LS12 LS13 LS14 LS15 LS16 LS17 LS18 LS19
Sliding Window in the 2nd round
Sliding Window in the 3rd round

Fig. 3. Local buffer (three layers with sliding window size of 5); blank *LS* indicates unavailable.

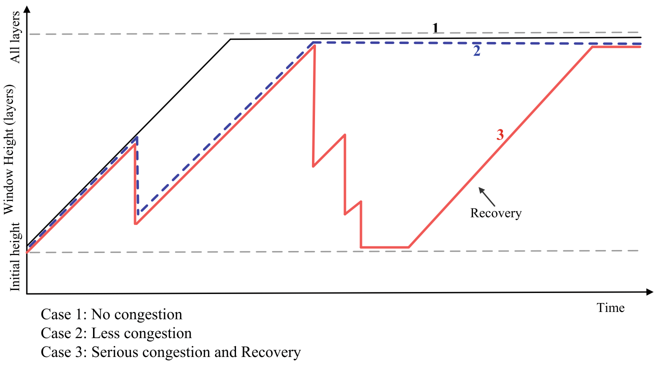

Fig. 4. Window adjustment mechanism.

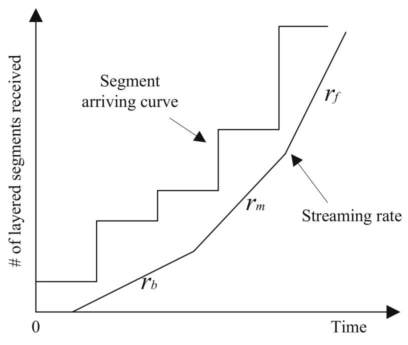

Fig. 5. Segment arrival rate and streaming rate.

Here, the objective function is to maximize the social welfare, i.e., the overall playback quality that all peers experience, which is calculated as total weight value of all the received segments before the playback deadline. The value of $q(j,k)$ is constant for P_k in each round according to the weight allocation mechanism (Section 5). For example, if P_k has buffered segment j, the value of $q(j,k)$ equals to zero since segment j makes no contribution to increase the playback quality of P_k. The first constraint indicates that the segments the peer needs are available in its partners. The second constraint suggests that fecthing duplicated segments from different partners should be avoided. The third shows the maximum number of segments that each peer can download, and the fourth is that the number of total providing segments can not beyond the upload bandwidth of each supplying peer.

The above scheduling problem requires global knowledge. Its Integer Linear Programming (ILP) problem form also implies that an efficient solution can hardly be derived. We thus resort to a local and efficient scheduling algorithm, as will be presented in details in Section 5.

5. Zigzag scheduling

We resort to a simple scheduling strategy as an optimal global scheduling can hardly be derived even for the single-layer streaming [1], not mention to the multi-layer one, particularly with large overlays. Moreover, peers may join and leave frequently and their buffer status thus would change quickly as well, which has yet to be addressed.

In our scheduling strategy, each peer can compute locally and effectively. Our scheduling objectives include: (1) well utilizing available network bandwidth; (2) reducing the playback delay as well as improving playback quality. In our solution, we allow each user to firstly determine the transmission order of the requested segments, in case not all the segments can be transmitted with bandwidth constraints. In particular, we implement a zigzag-like order to determine the importance of each segment. Fig. 6

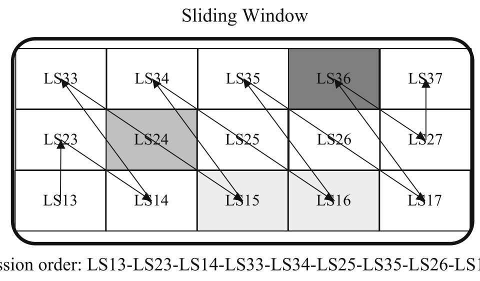
Sliding Window
LS33
LS34
LS35
LS36
LS37
LS23
LS24
LS25
LS26
LS27
LS13
LS14
LS15
LS16
LS17
Transmission order: LS13-LS23-LS14-LS33-LS34-LS25-LS35-LS26-LS17-LS27-LS37

Fig. 6. Zigzag importance allocation for segments within sliding window in the *3rd* round in Fig. 3.

illustrates this ordering for the segments in Fig. 3. The key idea of the zigzag ordering comes in three fold: (1) it follows the prediction dependency in SVC. That is, in the same time instant, higher layer segments are more important than lower layer ones; (2) playback deadline rule is obeyed, in which the required segments in nearer time instants are easier to approach the playback deadline, and thus, are allocated with higher importance values; (3) the key observation is that a higher layer segment at a nearer time instant is in practice more important than a lower layer segment at a farther time instant. For example, LS_{23} is considered more important than LS_{14}, because during normal playback the former one delivers better perceptual experience.

The zigzag-ordered segments are then scheduled to be transmitted from the most important to the least. For each one, we first find out all the partners that have buffered this segment. Among them, the one with the least number of the requested layered segments is selected to transmit this segment. This is because partners with a larger number of layered segments have more higher layer segments that can be transmitted later to enhance video quality and better utilize the available upload bandwidth. If several partners have the same number of requested layered segments buffered, the one with the highest bandwidth is chosen. If no partner has the requested segment, peers will finally skip it. Such operations are summarized in Algorithm 1.

Algorithm 1. FindBestCandidate (*candiates*,*j*)

```
1   // find out the best candidate to fetch segment j from
2   For i ∈ candidates do
3     bestPartners ← candidate with fewest LSs that are
      less important than j;
4   end
5   if sizeof bestPartners > 1 then
6     bestPartners ← candidate with highest bandwidth;
7     if sizeof bestPartners > 1 then
8       bestCandidate ← select one from bestPartners;
9     end
10    else
11      bestCandidate ← bestPartners;
12    end
13  end
14  else
15    bestCandidate ← bestPartners;
16  end
17  return bestCandidate;
```

Before scheduling, each peer will firstly generate the missing segments that are nearest to the current playback position. The sliding window in each peer determines the maximum time instants of segments that the peer can download in each round. However, the number of segments can be maximally download is not fixed and depends on the buffer state of each peer. A pseudo code of the scheduler at each peer is showed in Algorithm 2. Its complexity is bounded by $O(M \cdot P \cdot C)$, where M, P and C are the number of requested missing segments, the number of partners, and the buffer size (in number of segments), respectively. Given the limited sizes of partner set and buffer, the computation time of this scheduling algorithm is quite low.

Algorithm 2. Scheduling Strategy

```
    Input:
        partners:            set of partners;
        OrderedMissingSeg:   set of missing segments;
        partnersBW[i]:       bandwidth from partner i;
        bmp[i]:              buffer map of partner i;
1   for j ∈ OrderedMissingSeg do
2     // find out all partners that cached segment j;
3     for i ∈ partners do
4       if bmp[i,j] = true and partnersBW[i] > 0 then
5           add partner i to candiates;
6       end
7     end
8   // fetch segment j from one partner or skip it;
9   if candiates = Ø then
10      skip segment j;
11    end
12    else
13      i ← FindBestCandidate (candiates,j);
14      transStrategy ← fetch segment j from partner i;
15      partnersBW[i] ← partnersBW[i] − 1;
16    end
17  clear candidates;
18  end
```

We give out a toy example in Fig. 7 to illustrate our algorithm. Let $P_i(b_i)$ be the ith partner whose bandwidth is b_i. We only list the first four requested segments and assume that the receiving peer o has totally five partners in this example. The allocated upload bandwidth and available segments of each partner are shown in the figure, where the blank segments are not buffered and thus unavailable. We first schedule the most important segment LS_{13}. The candidates set of LS_{13} includes P_1, P_2, P_3 and P_4. Among them, P_1 has the fewest number of LSs that are less important than LS_{13}. Thus, LS_{13} is fetched from P_1. For the second important segment LS_{23}, P_3 and P_5 have buffered it and they both have one more less important LSs than LS_{23}. In this case, we break the tie by selecting the peer with higher bandwidth, that is, P_5. When scheduling LS_{33}, both P_2 and P_3 have the same number of less important LSs and the same bandwidth. In this case, either of them can be chosen to transmit segment LS_{33}.

6. Performance evaluation

6.1. Simulation setup

We evaluate the performance of our scheme through simulations. In our simulation, we have three classes of peers for our peer-to-peer VoD system, namely, Cable/Ethernet, DSL2 and Modem/ISDN/DSL1. We adopt similar parameters as [26]; and the download/upload bandwidths and peer population for each class are summarized in Table 1. We assume that peer arrival follows Poisson process with rate λ; this has been widely used in [27–31]. In our simulation, we set $\lambda = 20$. This indicates that on average 20 peers will arrive every minute. We also use Weibull distribution [32] to model the lifetime of the participating peers. Our simulation lasts for 60 min with varied cross traffic to present the

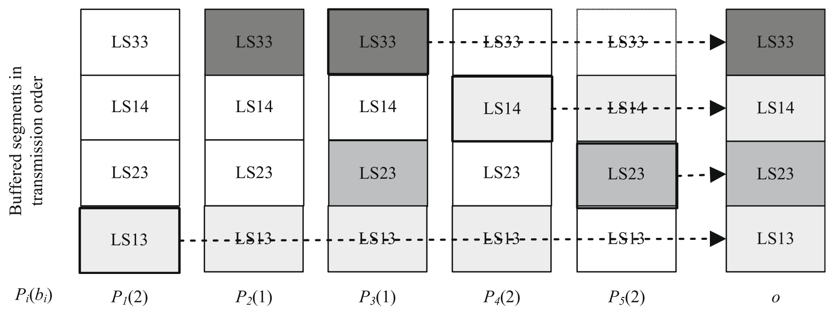

Fig. 7. Transmission strategy with five partners.

Table 1
Peer classes.

Class	Population	Download	Upload (Kbps)
Cable/Ethernet	60%	3 Mbps	768
DSL2	20%	1.5 Mbps	384
Modem/ISDN/DSL1	20%	768 Kbps	128

dynamic end-to-end resources. Each peer can join the system at different time, and will depart after its lifetime.

Without losing generality, we consider one video source with length of 15 min. By using SVC, the video source is encoded into a total of 50 layers with one base layer and 49 enhancement layers. It is observed in [23] that the source video rate is usually between 381 Kbps and 450 Kbps in the existing VoD systems. Thus, we set the streaming rate in our system to 400 Kbps. All layers are encoded with the same streaming rate, i.e., 8 Kbps. The frame rate is 30 frames per second and we set the GOP size to 15. Therefore, each layered segment (one second playback time) contains two GOPs and has a size of 8 Kbits (1 KB). This is a reasonable unit for efficient VoD transmission [23]. Each chunk cached in the local disk consists of one second of layered segments. The chunk size varies from one layered segment to 50 layered segments, which is 1–50 KB.

We implement a conventional peer-to-peer VoD system that uses single layer coding for comparison. Bandwidths are set as the same, and peers follow the same access and departure pattern with the same parameters. The different is that it uses one layer to encode the video and it does not have the zigzag scheduling strategy. To better understand how our system benefits the user experience with heterogeneous resources, we assume that both systems enable peers to cache all the viewed video chunks in their local disks.

6.2. User experience

We focus on user experience of startup delay, playback discontinuity and playback quality. In our simulation, the initial buffering time is set to 10 s. This means that peers can only start playing videos when 10 s of video segments are downloaded. Fig. 8 shows the cumulative distribution function (CDF) of the startup delay. We see that more than 95% of peers enjoy a startup delay of six seconds or less. Without SVC, only 85% peers can achieve a similar startup delay. With SVC, peers can fetch fewer layered segments; this leads to less time to fill the initial 10-s buffer. Both startup delays are relatively small since we assume peers have cached all the beginning part of the video to start watching quickly.

We use playback delay as the measurement for playback discontinuity. Thus, a smaller delay during playback will lead to a better viewing experience. Fig. 9 plots the average playback delay of all peers as a function of the number of peers. When the overlay size is small, the system without SVC has a high playback delay

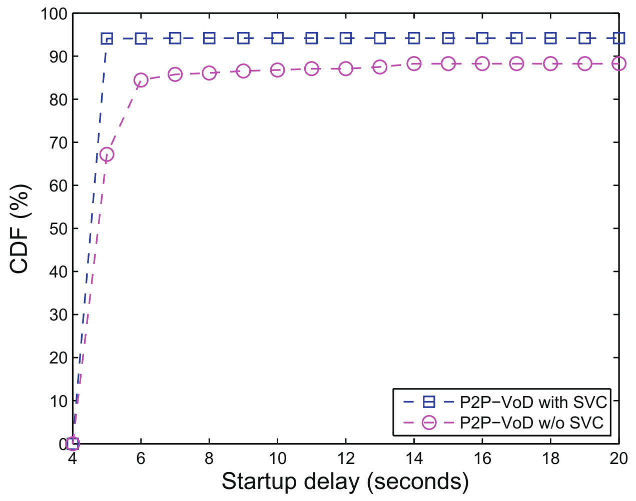

Fig. 8. CDF of startup delay.

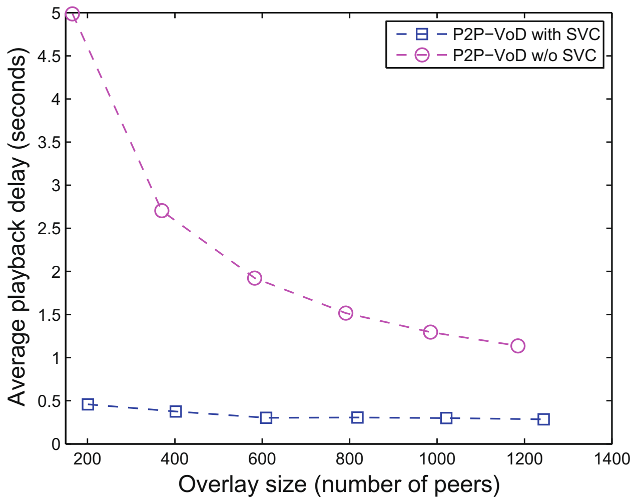

Fig. 9. Average playback delay.

of five seconds. The playback delay decreases when the number of peers is large. However, even when the number of peers is 1000, we still see a 1.5 s playback delay. As a sharp contrast, our system has an average of 0.5 s playback delay and is stable when the system scales.

We next define the user playback quality (*PQ*) as

$$PQ = \frac{\bar{r}}{d+1} \qquad (7)$$

here $\bar{r}$ denotes the average streaming rate and d denotes the playback delay; we use $d + 1$ to prevent the case where d arbitrarily approaches zero. Our argument is that a good user playback quality is affected not only by a high streaming rate, but also, a small playback delay. In other words, a high streaming rate may not deliver the best playback experience for users because it is more likely to have network congestion with more video data, and therefore, a larger playback delay.

Ideally, the *PQ* of the peers is equal to 400 Kbps/s when they receive all layers without any playback delays. Fig. 10 compares the *PQ* of our system and the conventional peer-to-peer VoD system. We can see that our user playback quality is much better with an improvement of at least 70% or more.

6.3. System performance

We next consider utilization of the network bandwidth, the system control overhead and the server load. Fig. 11 shows the total traffic that the server has to support as a function of the overlay size. Initially, only the server has the entire video source, and thus, the load gradually increases when overlay size is small. After the video has been distributed over the network, peers start to assist video distribution, and the bandwidth requirement on the serve becomes stable. System with SVC stabilizes more quickly, which indicates that video data is better distributed in our overlay. In addition, the server load of the system with SVC is only one half to that of the system without SVC.

We see in Fig. 12 the bandwidth utilization increases when overlay size increases. For both systems with or without SVC, 95% of the bandwidth is utilized when the overlay size is greater than 600 peers. This shows that our system with SVC can well utilize the available network bandwidth.

Fig. 13 shows the average control overhead of each peer as a function of the overlay size. The average control overhead decreases when more peers join the system. We see that SVC does not incur more control messages and the overhead is about 100 KB when the overlay size is 400. Such overhead is negligible comparing to the data traffic.

6.4. Optimization on window adjustment

Our window adjustment mechanism is based on three parameters, namely, the initial layers (number of layers a new peer firstly requests, or the initial window height), the window width, and the increasing step of the window for each round. In this section, we

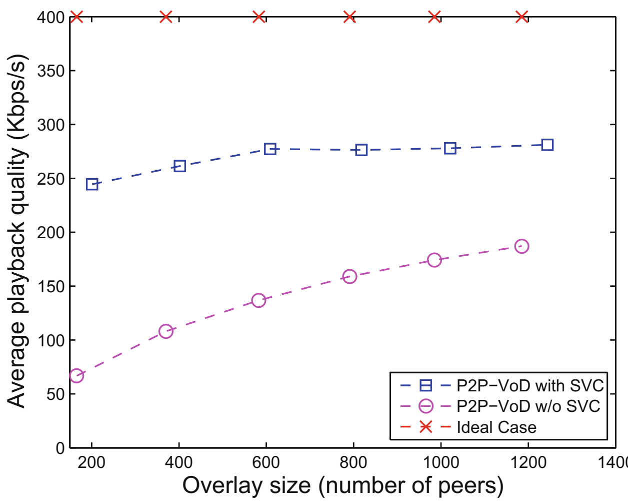

Fig. 10. Average playback quality.

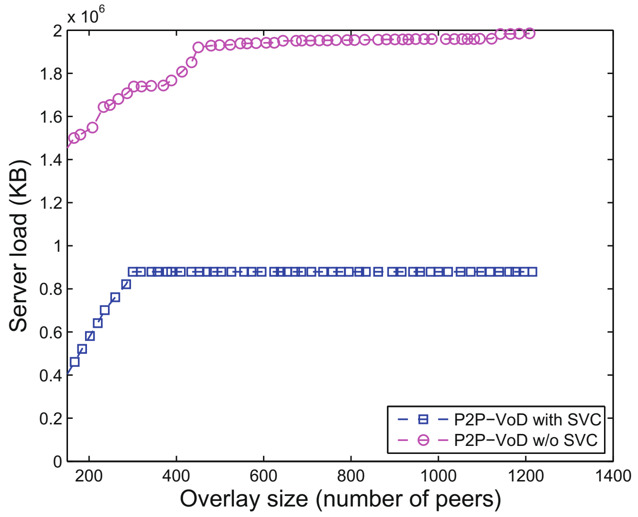

Fig. 11. Server load.

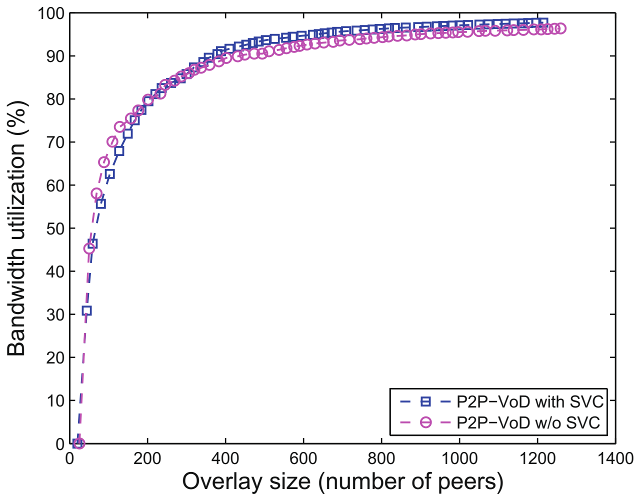

Fig. 12. Bandwidth utilization.

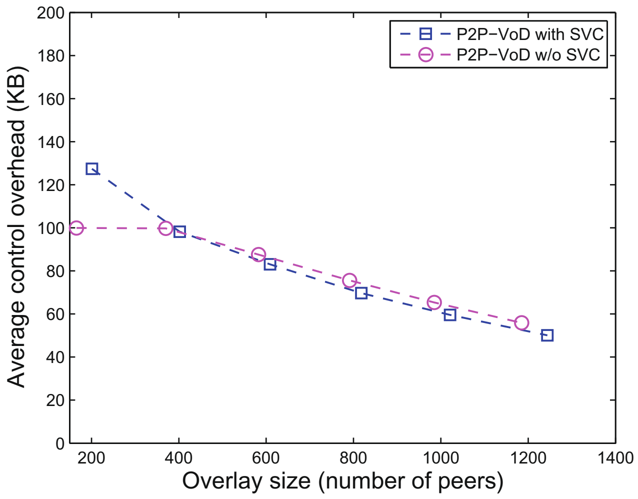

Fig. 13. Average peer overhead.

try to investigate how to set these parameters to optimize user playback experience. The simulation setup follows Section 6.1

except that we do not allow peers to cache all the viewed video content. According to [33], an optimal cache replacement algorithm in peer-to-peer VoD brings only marginal improvement, as compared with the simple Least Recently Used (LRU) and Least Frequently Used (LFU) schemes. Thus, we implement the LRU cache replacement in our simulation and assume that all peers have the same buffer size. We set the cache to be 10% (1.5 min) of the total video playback time. We further cache the beginning 10% of the video stream during the lifetime of a peer.

Fig. 14 compares the total delays, the playback quality and number of failed peers for different initial layers settings. We set the window increasing step to be 1 layer and the window width to be 5 s. User experience is the worst when the initial layer is set to 1, that is, peers start watching the video immediately when they receive the base layer segments. Setting the initial layer to 5 brings a better playback quality, but introduces more failed peers. Clearly there is a tradeoff between the two. A higher initial layer has a higher streaming rate, but at the same time, peers may not have sufficient bandwidths to collect the segments in the future playback positions. Thus, it is more likely to have playback delays which will reduce user playback quality (*PQ*).

Fig. 15 shows user playback experience with different values of window increasing step. We can see that both the total delays and the playback quality become slightly worse when the step increases. This indicates that a small increasing step is preferred. We have yet to find an optimal value for failed peers, however. The impact of window width is also studied in Fig. 16. We observe that the window width of 5 s is the best choice with smallest delays, highest quality and fewest failed peers.

7. Conclusion and discussion

Peer-to-peer has been recently suggested to be a promising architecture to stream on-demand videos. These peer-to-peer Video-on-Demand (VoD) systems take advantage of local disk space as well as the contributed upload bandwidth of peers to potentially relieve the server load. However, reducing the startup delay in peer-to-peer VoD has been yet to be well addressed, which is critically important since VoD users frequently switch among different videos. This paper designed a novel peer-to-peer VoD system that utilized Scalable Video Coding (SVC) to minimize startup delay and improve playback experience. SVC enables quick start by allowing newly joined peers to successfully initialize the video playback when receive base layer video segments only. Further, the occurrences of frame freezing due to the network congestion or the insufficient peer bandwidths can be minimized by dynamically dropping the enhancement layer video segments. To best balance the startup delay and playback quality, we developed a practical scheduling strategy to maximize the total weight value of received layered segments, each of which was allocated an importance value by a proposed zigzag like mechanism to indicate its contribution to improve playback experience. Our evaluation results showed that the proposed system well utilizes the network bandwidth and outperforms conventional peer-to-peer VoD systems in terms of startup delay and playback quality.

Further, the benefit that SVC brings is orthogonal to that of many existing peer-to-peer VoD systems that focus on the overlay construction, i.e., the neighbor selection. Although some of these systems such as [9] are not specifically designed for layered streaming, the key ideas are still useful for constructing overlays for layered streaming. Their proposed mechanisms select good neighbors, and in turn provide better underlying support for our SVC based data scheduling. However, it is more difficult to implement scalable video coding for the VoD systems practically than the traditional coding such as H.264/AVC [10] due to its higher coding complexity. The novel scalable video coding extension of the H.264/AVC standard [34] has achieved significant

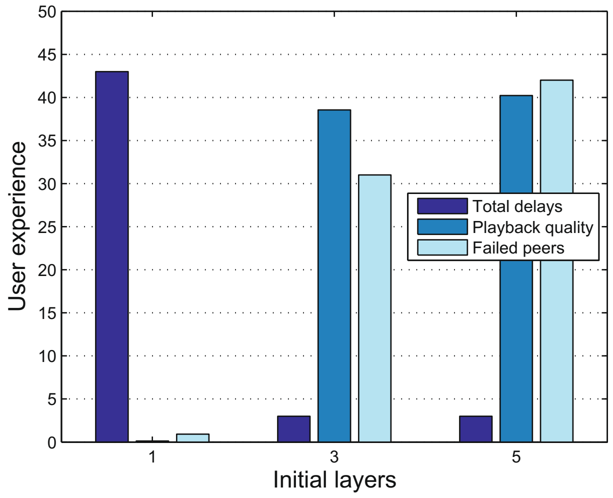

Fig. 14. Initial playback layers. The total delays, playback quality and failed peers are in terms of seconds, KBps/s and numbers, respectively.

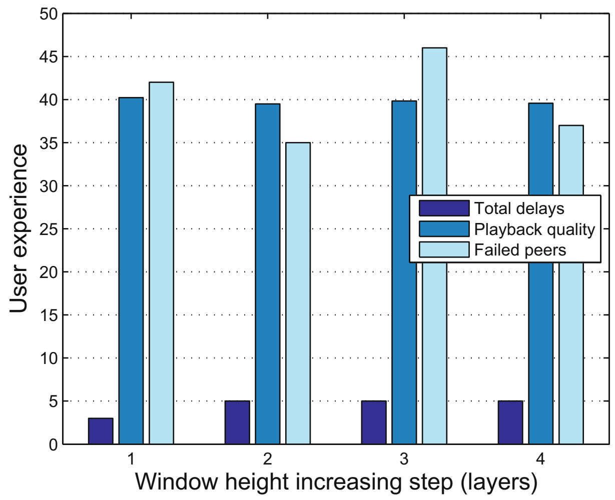

Fig. 15. Window increasing step. The total delays, playback quality and failed peers are in terms of seconds, KBps/s and numbers, respectively.

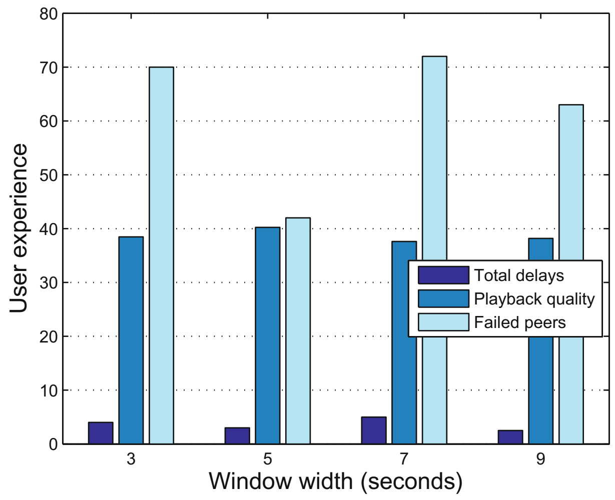

Fig. 16. Window width. The total delays, playback quality and failed peers are in terms of seconds, KBps/s and numbers, respectively.

improvement in coding efficiency as well as providing scalability functionalities, and thus makes it possible to implement efficient layered VoD streaming on the Internet.

Acknowledgment

This research is supported by a Canada NSERC Strategic Grant, an NSERC Discovery Grant, an NSERC DAS grant, and an MITACS Project Grant. This research is also supported by grant Hong Kong PolyU/G-YG78, A-PB0R, A-PJ19, 1-ZV5W, and RGC/GRF PolyU 5305/08E. In addition, this work is supported in part through National Natural Science Foundation of China (No. 60803115), the Fundamental Research Funds for the Central Universities (No. M2009022), the Youth Chenguang Project of Wuhan City (No. 201050231080), the CHUTIAN Scholar Project of Hubei Province, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

References

[1] X. Zhang, J. Liu, B. Li, T. Yum, CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming, in: Proc. of IEEE INFOCOM'05, vol. 3, Miami, FL, USA, 2005, pp. 2102–2111.
[2] PPLive. Available from: <http://www.pplive.com/>.
[3] UUSee. Available from: <http://www.uusee.com/>.
[4] B. Cheng, L. Stein, H. Jin, X. Liao, Z. Zhang, GridCast: improving peer sharing for P2P VoD, ACM Transactions on Multimedia Computing, Communications and Applications 4 (4) (2008) 1–31.
[5] B. Chang, L. Dai, Y. Cui, Y. Xue, On feasibility of P2P on-demand streaming via empirical VoD user behavior analysis, in: Proc. of the 28th International Conference on Distributed Computing Systems Workshops (ICDCSW'08), Beijing, China, 2008, pp. 7–11.
[6] M. Saxena, U. Sharan, S. Fahmy, Analyzing video services in Web 2.0: a global perspective, in: Proc. of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'08), Braunschweig, Germany, 2008, pp. 39–44.
[7] B. Cheng, H. Jin, X. Liao, Supporting VCR functions in P2P VoD services using ring-assisted overlays, in: Proc. of IEEE International Conference on Communications (ICC'07), Glasgow, Scotland, 2007, pp. 1698–1703.
[8] D. Wang, J. Liu, A dynamic skip list-based overlay for on-demand media streaming with VCR interactions, IEEE Transactions on Parallel and Distributed Systems 19 (4) (2008) 503–514.
[9] X. Qiu, C. Wu, X. Lin, F. Lau, InstantLeap: fast neighbor discovery in P2P VoD streaming, in: Proc. of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'09), Williamsburg, Virginia, 2009, pp. 19–24.
[10] Advanced video coding for generic audiovisual services, ITU-T recommendation H.264-ISO/IEC 14496-10(AVC), ITU-T and ISO/IEC JTC 1, 2003.
[11] J. Liu, S. Rao, B. Li, H. Zhang, Opportunities and challenges of peer-to-peer internet video broadcast, in: Proc. of the IEEE (Invited paper), Special Issue on Recent Advances in Distributed Multimedia Communications 96 (1) (2008) 11–24.
[12] M. Hefeeda, A. Habib, B. Botev, D. Xu, B. Bhargava, PROMISE: peer-to-peer media streaming using CollectCast, in: Proc. of the 11th ACM International Conference on Multimedia (ICM'03), ACM, New York, NY, USA, 2003, pp. 45–54.
[13] Y. Guo, K. Suh, J. Kurose, D. Towsley, P2Cast: peer-to-peer patching scheme for VoD service, in: Proc. of the 12th International Conference on World Wide Web (WWW'03), Budapest, Hungary, 2003, pp. 301–309.
[14] N. Vratonjic, P. Gupta, N. Knezevic, D. Kostic, A. Rowstron, Enabling DVD-like features in P2P video-on-demand systems, in: ACM P2P-TV'07, Kyoto, Japan, 2007, pp. 329–334.
[15] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, M. Varvello, Push-to-peer video-on-demand system: design and evaluation, IEEE Journal on Selected Areas in Communications 25 (9) (2007) 1706–1716.
[16] Y. Cui, K. Nahrstedt, Layered peer-to-peer streaming, in: Proc. of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), Monterey, CA, USA, 2003, pp. 162–171.
[17] Z. Liu, Y. Shen, S. Panwar, K. Ross, Y. Wang, Using layered video to provide incentives in P2P live streaming, in: Proc. of the 2007 Workshop on Peer-to-peer Streaming and IP-TV'07, Kyoto, Japan, 2007, pp. 311–316.
[18] P. Baccichet, T. Schierl, T. Wiegand, B. Girod, Low-delay peer-to-peer streaming using scalable video coding, in: Proc. of Packet Video (PV'07), Lausanne, Switzerland, 2007, pp. 173–181.
[19] R. Rejaie, A. Ortega, PALS: peer-to-peer adaptive layered streaming, in: Proc. of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), Monterey, CA, USA, 2003, pp. 153–161.
[20] N. Magharei, R. Rejaie, Adaptive receiver-driven streaming from multiple senders, ACM Multimedia Systems 11 (6) (2006) 550–567.
[21] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, P. Rodriguez, Exploring VoD in P2P swarming systems, in: Proc. of IEEE INFOCOM'07, Anchorage, Alaska, USA, 2007, pp. 2571–2575.
[22] Y. Tang, J.-G. Luo, Q. Zhang, M. Zhang, S.-Q. Yang, Deploying P2P networks for large-scale live video-streaming service, IEEE Communications Magazine 45 (6) (2007) 100–106.
[23] Y. Huang, T. Fu, D. Chiu, J. Lui, C. Huang, Challenges, design and analysis of a large-scale P2O-VoD system, ACM SIGCOMM Computer Communication Review 38 (4) (2008) 375–388.
[24] S. Floyd, Congestion Control Principles, Tech. Rep., BCP 41, RFC 2914, September 2000.
[25] B. Cheng, X. Liu, Z. Zhang, H. Jin, A measurement study of a peer-to-peer video-on-semand system, in: Proc. of 6th International Workshop on Peer-to-Peer Systems (IPTPS'07), Bellevue, WA, USA, 2007.
[26] C. Huang, J. Li, K. Ross, Can internet video-on-demand be profitable? in: Proc. of ACM SIGCOMM'07, Kyoto, Japan, 2007, pp. 133–144.
[27] D. Liben-Nowell, H. Balakrishnan, D. Karger, Analysis of the evolution of peer-to-peer systems, in: Proc. of the 21th Annual ACM Symposium on Principles of Distributed Computing (PODC'02), Monterey, CA, USA, 2002, pp. 233–242.
[28] D. Qiu, R. Srikant, Modeling and performance analysis of BitTorrent-like peer-to-peer networks, ACM SIGCOMM Computer Communication Review 34 (4) (2004) 367–378.
[29] T. Li, M. Chen, D.-M. Chiu, M. Chen, Queuing models for peer-to-peer systems, in: Proc. of 8th International Workshop on Peer-to-Peer Systems (IPTPS'09), Boston, MA, USA, 2009.
[30] J. Wang, C. Yeo, V. Prabhakaran, K. Ramchandran, On the role of helpers in peer-to-peer file download systems: design, analysis and simulation, in: Proc. of International Workshop on Peer-to-Peer Systems (IPTPS'07), Bellevue, WA, USA, 2007.
[31] R. Mahajan, M. Castro, A. Rowstron, Controlling the cost of reliability in peer-to-peer overlays, Lecture Notes in Computer Science (2003) 21–32.
[32] Y. He, G. Siganos, M. Faloutsos, S. Krishnamurthy, A systematic framework for unearthing the missing links: measurements and impact, in: Proc. of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI'07), Cambridge, MA, USA, 2007.
[33] J. Wu, B. Li, Keep Cache replacement simple in peer-assisted VoD systems, in: Proc. of IEEE INFOCOM'09 Mini-Conference, Rio de Janeiro, Brazil, 2009, pp. 2591–2595.
[34] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard, IEEE Transactions on Circuits and Systems for Video Technology 17 (9) (2007) 1103–1120.