Incentivized Peer-assisted Streaming for On-demand Services

Chao Liang[†], Zhenghua Fu[‡], Yong Liu[†], and Chai Wah Wu[‡]
[†]Polytechnic Institute of NYU, Brooklyn, NY, USA 11201
[‡]IBM T.J.Watson Research Center, Hawthorne, NY, USA 10532
Email: cliang@photon.poly.edu, zfu@us.ibm.com, yongliu@poly.edu, cwwu@us.ibm.com

***Abstract*—As an efficient distribution mechanism, Peer-to-Peer (P2P) technology has become a tremendously attractive solution to offload servers in large-scale video streaming applications. However, in providing on-demand asynchronous streaming services, P2P streaming design faces two major challenges: how to schedule efficient video sharing between peers with asynchronous playback progresses? how to provide incentives for peers to contribute their resources to achieve a high level of system-wide Quality-of-Experience (QoE)? In this paper, we present iPASS, a novel mesh-based P2P VoD system, to address these challenges. Specifically, iPASS adopts a dynamic buffering-progress-based peering strategy to achieve high peer bandwidth utilization with low system maintenance cost. To provide incentives for peer uploading, iPASS employs a differentiated pre-fetching design that enables peers with higher contribution pre-fetch content at higher speed. A distributed adaptive taxation algorithm is developed to balance the system-wide QoE and service differentiations among heterogeneous peers. To assess the performance of iPASS, we built a detailed packet-level P2P VoD simulator and conducted extensive simulations. It was demonstrated that iPASS can completely offload server when the average peer upload bandwidth is more than 1.2 times the streaming rate. Furthermore, we showed that the distributed incentive algorithm motivates peers to contribute and collaboratively achieve a high level of system wide QoE.**

***Index Terms*—peer-to-peer, video streaming, on-demand, incentive**

I. Introduction

Video-on-Demand (VoD) services enable users to watch their favorite videos at any time. YouTube, an extremely popular VoD application on the Internet, serves 100 million distinct videos daily [1]. Traditional VoD solutions employ video servers and content distribution networks (CDNs) to stream video to viewers. The infrastructure cost grows linearly with the user population and the video quality. It will become very expensive for YouTube to stream higher resolution videos with TV or even high-definition quality. On the other hand, P2P technology utilizes available resources of peers and effectively offloads servers in large scale content distribution, such as file sharing [2] and live video streaming [3]. Recently, P2P technology has also been adopted to provide VoD services. In providing VoD services, P2P streaming design faces two major challenges: how to schedule efficient video sharing between peers with asynchronous playback progresses? how to provide incentives for peers to contribute their resources to achieve a high level of system-wide Quality-of-Experience (QoE)?

To address the asynchronous user playback issue, the *Cache-and-Relay* approach has been proposed. Peers store downloaded video in memory or hard disk, and relay the cached video to other peers in future, leading to asynchronous P2P video sharing. Early Cache-and-Relay based systems assume a small amount of video cache on peers and exploit asynchronous sharing between peers with close playback progresses. Through batching, peers are organized into groups according to their playback time and a tree-like topology is formed for peers in the same group to exchange video [4], [5]. Unfortunately, small video caching results in low P2P sharing efficiency. The structured P2P topology incurs high management overhead and is vulnerable to dynamic peer arrivals and departures. Recent advances in computer hardware technology make low-priced computers increasingly equipped with abundant memory and storage. New P2P VoD systems fully exploit the largely improved peer video caching capability for higher P2P sharing efficiency. In [6], [7], peers are effectively turned into distributed "video seeds" by caching a large volume of video clips on their hard disks. Longer video caching also makes it possible for P2P VoD systems to adopt mesh-based topology. Mesh-topology is robust to peer churn and easy to manage. It has demonstrated its successes in many large scale file sharing [2] and live streaming systems [3]. Inspired by the success of mesh structure, several mesh-based P2P VoD systems have been proposed [8], [9], [10]. In those systems, peers form one or multiple meshes randomly and exchange data with neighbors. Unlike in file sharing, data sharing in VoD systems is commonly uni-directional between peers. Data flows from a peer to its neighbors with smaller playback progresses. We will show that random peering leads to poor peer resource utilization under this data flow directionality. How to design P2P VoD systems with high peer bandwidth utilization and low maintenance cost remains to be a challenging research problem.

Meanwhile, providing incentives for peers to contribute their resources is another essential design component for P2P systems. In file sharing systems, peers are motivated to upload content to other peers in order to achieve a higher download rate from the system. By employing the *tit-for-tat* policy, BitTorrent punishes free-riders who do not contribute bandwidth to the system. In live streaming systems, peers are motivated to contribute more in order to get better playback quality. It was proposed in [11], [12] that, with scalable video coding, peers uploading more will be rewarded with higher video quality. Due to the asynchronous peer playback progress and the data flow directionality, tit-for-tat type of direct reciprocity incentive mechanism is not applicable to P2P VoD systems. In addition, to maintain the playback continuity, each peer needs to download video data before their playback deadlines. It is critical to design incentive mechanism for P2P VoD systems to balance the system-wide QoE and service differentiations among heterogeneous peers.

In this paper, we present iPASS, a novel mesh-based P2P VoD system, to simultaneously address the previously described efficiency and incentive issues. iPASS adopts a dynamic Buffering-Progress-Based (BPB) peering strategy to achieve high peer bandwidth utilization with low system maintenance cost. To provide incentives for peer uploading, iPASS employs a differentiated pre-fetching design that enables peers with higher contributions to pre-fetch content at higher speeds. A distributed adaptive taxation algorithm is developed to balance the system-wide QoE and service differentiations among heterogeneous peers. The contributions of this paper can be summarized as follows:

1) We analytically study the impact of asynchronous peer playback progresses on the efficiency of mesh-based P2P sharing, and propose a distributed BPB peering strategy. Through analysis and simulation, we show that the BPB peering can enable mesh-based P2P VoD systems to achieve high peer bandwidth utilization, low maintenance cost and high peer churn robustness. Moreover, BPB also facilitates the support of VCR interactions.
2) To the best of our knowledge, we are the first to use differentiated pre-fetching as an incentive mechanism to motivate capable peers to contribute in P2P VoD systems. We demonstrate that pre-fetchings on peers can be coordinated by an adaptive taxation algorithm to simultaneously maintain system-wide QoE and provide service differentiations among peers with different contributions.
3) To assess the performance of iPASS, we analytically investigate iPASS's system resource utilization and the equilibrium state using a fluid modeling approach. We built a detailed packet-level P2P VoD simulator and conducted extensive simulations. Compared with previous P2P streaming simulators, our simulator can examine the packet-level details. In addition, it can prolong the simulation duration to hours in order to study long-term system behaviors under a rich set of simulated scenarios.

The remaining of this paper is organized as follows. We briefly discuss the related work in Section II. The main design components are outlined in Section III. The detailed system implementation is presented in Section IV. The system properties and analysis are described in Section V. The performance evaluation with numerical simulation results is presented in Section VI. Finally, the paper is concluded in Section VII.

II. Related work

P2P sharing can greatly reduce server bandwidth cost to provide on-demand streaming service [13]. Recently, various P2P on-demand systems have been deployed [6], [14] and various schemes to support VCR interactions have been proposed [15], [16]. The design of VoD systems keeps evolving. Early P2P VoD systems adopt structured streaming topologies and require delicate system management. P2Cast [5] groups peers according to their arrival time. Peers in the same group are organized into a multicast tree. Peers retrieve video content through a combination of streaming along the tree and patching from peers who arrived earlier. dPAM [4] employs distributed pre-fetching to improve system performance. oStream [17] constructs media distribution trees at the application layer to realize asynchronous media delivery.

Recent advances in computer hardware technology largely improve peers' video caching capability and broaden the design spaces of VoD systems. Inspired by P2P file sharing and live streaming systems, *mesh-based* data swarming has been adopted by new P2P VoD systems. BiTos [10] customized the Bittorrent protocol for on-demand video streaming. PONDER [9] divides video into multiple sub-clips and forms multiple meshes, one for each sub-clip. Peer selection and measurement based admission control was proposed to manage swarms. BASS [18] combines streaming from the server with Bittorrent-assisted downloading. The impact of segment scheduling, overlay management and network coding on the performance of swarming-based VoD systems has been discussed in [8]. In the above mesh-based P2P VoD systems ([8], [9], [10], [18]), each peer randomly connects others to form one or multiple overlay meshes, which are scalable and robust to peer churn. However, the peering strategy commonly does not take into account neighbors' buffering or playback progresses, and this status-oblivious peering leads to inefficient peer bandwidth usage. In contrast, the buffering progress based peering strategy of iPASS can substantially enhance the system resource utilization. Meanwhile, the structured mesh formed by the peers is amenable to provide VCR functions.

On the other hand, more and more attentions have been attracted to the design of incentive mechanisms in P2P systems to discourage free-riders. Various schemes have been proposed for file sharing and live streaming systems. Models are developed to study the phenomenon of free-riding in P2P systems [19], [20]. A *tit-for-tat* type of substream trading algorithm was developed in [12] to provide incentive in live streaming systems with layered video coding. A taxation scheme [21] was proposed to improve the overall social welfare through subsidizing resource-poor peers by exploiting resource-rich peers, which has been applied in [11] to encourage peers to contribute to obtain higher playback quality. However, previous incentive schemes are generally based on direct reciprocity, providing incentive in on-demand systems remains a challenging problem. Due to the asynchronous playback progress and the directional data flow, it rarely takes place in on-demand systems that, between a pair of hosts, each host has the content the other host needs. By motivating peers to contribute for higher pre-fretching speeds, an effective adaptive taxation scheme is proposed in iPASS to address the incentive issue in on-demand system from a new perspective.

III. iPASS: Design Overview

In this section, we present the two major design components of iPASS: *Buffering-Progress-Based (BPB) peering* and *Adaptive-Taxation-Based (ATB) pre-fetching*.

A. System Model

In *peer-assisted* VoD systems, servers host publishers' videos and stream them to peers upon requests. To save bandwidth consumption of servers, peers viewing the same video form a P2P overlay network and redistribute videos among themselves. Severs are responsible for maintaining peers' playback continuity. If a peer cannot download video data from other peers before the playback deadline, it will download the missing data from the server directly, which increases the server bandwidth cost consequently. A key design issue of peer-assisted VoD systems is to minimize the server bandwidth cost by efficiently utilizing peers' upload bandwidth.

Peers start to watch from the beginning of the video after they join the system. Peers are allowed to pre-fetch content beyond its playback range, and the pre-fetching speeds of peers are subject to the regulation of incentive mechanism. In addition, peers can conduct VCR interactions as *random seek*, *fast-forward* and *rewind*.

There are two unique features in VoD systems: the playback progresses on peers are *asynchronous*; peers can download content beyond its current playback range. To cope with bandwidth variations and peer churn, a peer normally buffers a certain amount of video beyond its playback progress. To describe the status of a peer, we introduce the following notations for peer i in the system:

- *Playback progress* p_i: the current playback position of peer i, indexed by the sequence number of the video chunk being played.
- *Buffering progress* b_i: the sequence number of the first missing chunk beyond current playback position p_i.
- *Buffering level* τ_i: the number of continuous buffered chunks beyond the current playback progress point. By definition, $\tau_i = b_i - p_i$.
- *Playback buffering threshold* w_{rd}: the number of buffered chunks necessary for smoothing playback. We call the sliding window $[p_i, p_i + w_{rd}]$ peer i's continuous playback range.
- *Contribution level* c_i: the number of chunks that peer i has uploaded to other peers since it joins the system.

Fig. 1 illustrates two different peer buffer statuses. On Peer 1, buffering level τ_1 is lower than the playback buffering threshold w_{rd}. It is downloading the missing chunks in the continuous playback range. We say that peer 1 is in the *normal playback mode*. On peer 2, buffering level τ_2 is higher than the playback buffering threshold w_{rd}. Peer 2 is downloading chunks outside of the playback range. We say that peer 2 is in the *pre-fetch mode*.

Peers are assumed to have enough storage to cache what they ever watched. Due to copyright issues, the content in the cache of a peer will be eliminated once the peer leaves the system, and the peer cannot serve as a seed anymore. We also

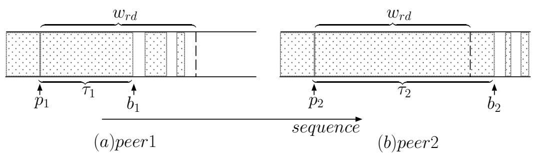


Fig. 1. Peer buffer status

assume peers are obedient to unveil their truthful information to each other (There exist reputation mechanisms [22], [23] to ensure peers truthfully reporting their information.)

B. Buffering Progress Based Peering

We first investigate the impact of asynchronous peer playback progresses on the efficiency of content sharing among peers. We then present the Buffering-Progress-Based peering strategy and study the impact of various peering strategies on server cost.

1) *Impact of Asynchronous Playback on P2P Sharing*. Let's start with a peer-assisted VoD system with homogenous N peers, each of them with upload bandwidth u. Suppose each peer randomly selects k peers as its neighbors. The video length is L. Peers store the content they have already played. Asynchronous playback leads to asynchronous buffering progresses among peers. Obviously, a peer can only serve peers with buffering progress behind it. In addition, a peer divides its upload bandwidth equally to all its receivers.

Proposition 1: With random peering and equal bandwidth sharing, the expected download rate of a peer from other peers drops as the peer buffering progress increases.

Proof: Let the random variable X denote the buffering progress of a selected peer, and f_x denotes the density function of its distribution. Each peer selects k neighbors randomly and independently, and only the neighbors with larger buffering progress can be the suppliers. Given a peer with buffering progress x, the expected number of neighbors with smaller buffering progress is $kP(X < x)$. Let d_x represent the expected distribution rate from this peer with buffering progress x to each of its receivers. With equal bandwidth sharing, the distribution rate can be approximated by

$$d_x = \frac{u}{kP(X < x)} \tag{1}$$

For peer with buffering progress y, we can obtain the expected aggregate download rate D_y from its k neighbors

$$\begin{aligned} E[D_y] &= k \int_0^L d_x f_x I(x > y) dx \\ &= k \int_y^L d_x f_x dx \end{aligned} \tag{2}$$

From the above equation, we can deduce that the expected download rate D_y decreases as the buffering progress y increases. ■

Specifically, when X has uniform distribution, we can get $E[D_y] = u(lnL - lny)$, which shows that the expected possible download rate drops logarithmically as the buffering progress increases. For peers with larger buffering progress,

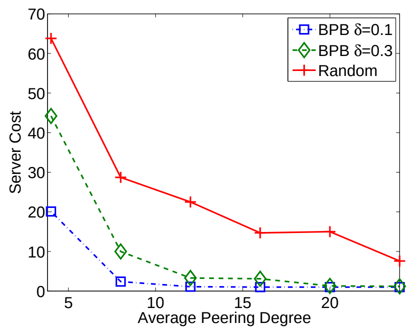


Fig. 2. Normalized Server Cost with $\rho = 1.2$

due to the random neighbor selection, they will find fewer suppliers in their random neighbor set, from which they can download video. In addition, a supplier with larger progress will be able to serve more download requests. Due to the equal bandwidth sharing, it will upload to each of its receivers at a lower rate. These two factors conspire and lead to low download rates for peers with large buffering progress. This shows that random peering and equal bandwidth sharing lead to low P2P bandwidth sharing efficiency.

2) *Buffering-Progress-Based Peering*. The bandwidth sharing efficiency in P2P systems is mainly determined by two factors: how peers are connected and how a peer allocates its upload bandwidth to all its neighbors. In the previous section, we have demonstrated that random peering and equal bandwidth sharing is not efficient for asynchronous P2P VoD systems. Peers with larger buffering progresses have less opportunity to download from the P2P network. Intuitively, this suggests some heuristic peering policy: *to increase the download rate of peers with large progress, the upload bandwidth of peers close to the end of the streaming session should not be invested to peers who just joined the session*. More generally, we propose the Buffering Progress Based (**BPB**) peering to let peers connect to peers with close buffering progress. Peers form one structured mesh overlay with BPB peering strategy. In the mesh topology constructed under BPB, peers with similar buffering progresses are preferentially connected. A fraction of neighbors of a peer are suppliers with larger buffering progress. Another fraction of neighbors are receivers with buffering progress lagging behind the peer. And remaining neighbors have very close progress and overlapping download interests, they may act as either supplier or receiver. On top of the BPB mesh, peers adaptively allocate their upload bandwidth to their neighbors to maximally reduce the complementary streaming requests to servers. Moreover, as we would present in the following section, the BPB peering can adjust peers' connection relationship dynamically to accommodate their differentiated pre-fetching speeds. Meanwhile, the structured mesh can be maintained well in the face of peer churn.

To study the impact of peering and bandwidth allocation on server bandwidth, we formulate the following Linear Programming model. There are a set V of peers in a peer-assisted on-demand system. The video streaming rate is r. For peer i, let $nb(i)$ be its neighbor set, and u_{ji} be the download rate from peer j. The aggregate download rate from all its neighbors is $\sum_{j \in nb(i)} u_{ji}$, and then the complementary streaming rate needed from the server is $\max(0, r - \sum_{j \in nb(i)} u_{ji})$. The goal is to find the optimal peer bandwidth allocation to minimize the aggregate server cost, then we have

$$\min_{\{u_{ji}\}} \sum_{i \in V} (r - \sum_{j \in nb(i)} u_{ji}) \tag{3}$$

subject to

$$\sum_{j \in nb(i)} u_{ij} \leq u_i, \quad i \in V \tag{4}$$

$$u_{ij} \leq I_{ij} u_i, \quad i, j \in V \tag{5}$$

$$\sum_{j \in nb(i)} u_{ji} \leq r, \quad i \in V. \tag{6}$$

In the above formulation, I_{ij} denotes the buffering progress relationship between peer i and j, $I_{ij} = 1$ when $p_i > p_j$, otherwise equal to 0. Eq. (4) states the bandwidth constraint for each peer respectively. And Eq. (5) shows the content constraint among peers. Eq. (6) presents the download speed constraint without pre-fetching.

The above optimal bandwidth allocation formulation can be applied to general topology. We now use it to compare the server bandwidth saving of random peering and BPB peering. Towards this goal, we generate an instance of a peer-assisted video-on-demand system using a discrete time simulation. During the simulated session with duration $T = 100$, peers arrive at the system according to a Poisson process with rate $\lambda = 2$. We assume all peers' download bandwidth is greater than r. There are two types of peers with upload bandwidth $1.5r$ and $0.5r$ respectively. The normalized average peer upload bandwidth is $\rho = \bar{u}/r = 1.2$.

With random peering, upon its arrival, a peer randomly picks k peers already in the system as its neighbors. With BPB peering, peers are firstly ordered in the increasing order of their arrival times. A peer who arrived at the system with rank i will randomly pick k neighbors from peers with arrival ranks in the range of $[i - \delta * N, i]$ given a total of N online peers. By changing δ, we manipulate the playback progress closeness of neighbors in the constructed BPB graph. We then compare the normalized server cost under BPB and random peering strategies with five snapshots of the system. With each snapshot, we solve the optimal bandwidth allocation problem defined in (3). Figure 2 shows the minimum server cost can be achieved with different peering strategies. The results indicate that with limited peering degree, BPB-peering can significantly reduce the server cost compared with random peering. Due to more efficient system resource distribution and partnership setup among peers, BPB-peering can achieve higher system resource utilization. The results obtained here is only the lower bounds on the server bandwidth cost. In Section VI, we will compare the server bandwidth saving of random peering and BPB peering via detailed packet-level simulations.

C. Adaptive Taxation Based Pre-fetching

Providing incentive in asynchronous VoD system is challenging. The asymmetric data flows between peers with different playback progresses make direct reciprocity incentive mechanisms infeasible, such as tit-for-tat of BitTorrent. In our design, we use pre-fetching as an incentive to motivate peers to contribute more to obtain higher download rate from the system. To maintain the playback continuity in the face of peer churn and network dynamics in P2P streaming systems, peers normally buffer certain amount of data ahead of the playback progress. Furthermore, in P2P VoD systems, peers with high download rate can *pre-fetch* content beyond their playback points and potentially become *seeds*, namely, nodes with the whole content, long before their playbacks finish. From the system point of view, more seeds in the system, more efficient the content sharing among peers. As will be shown through simulations in Section VI : *with large enough peer bandwidth resources and high scheduling efficiency, the seeds that evolved from regular peers with pre-fetching may completely take the place of the servers and results in zero server cost.* From individual peers' point of view, with pre-fetched content in the buffer, they can enjoy smooth non-linear viewing operations, such as fast-forwarding and jumping. Moreover, peers can finish the download process of the whole content before they finish the playback, and they have options to leave the system to proceed other Internet applications without interference. Hence, pre-fetching can be leveraged to motivate peers to contribute and facilitate the system distribution process.

To coordinate the asynchronous demands of peers and maintain system-wide quality, we propose a *Adaptive Taxation* scheme to regulate the pre-fetching on heterogeneous peers. The bandwidth of a peer can be treated as its *wealth*. And the video pre-fetching can be regarded as a peer's *income*. Resource-rich peers contribute more bandwidth to the system, and subsidize for the resource-poor peers. The tax regulated redistribution of peer wealth helps improve the social welfare and then reduce server cost. The tax ratio is fixed in the original taxation scheme [21] and applied for providing incentive in live streaming system. To balance the budget, the *demogrant* rate (i.e., one peer who does not contribute anything, still receives the demogrant rate) is adaptive. On the contrary, in peer-assisted systems, although peers may have different download rates, the base playback rates for all peers are guaranteed. Therefore, in our adaptive taxation method, the demogrant rate is fixed to be equal to the playback rate and the tax ratio would be adaptive. (In peer-assisted systems, the pure free-riders should also be discouraged. In practice, the system can enforce peers to contribute with at least certain upload rates.)

Suppose we pose a taxation ratio t on peers. All peers have download rates no less than the base playback rate r. Then peer i with contribution level c_i and lifetime T_i, could get the average download rate r_i to accumulate the expected buffering level

$$\tau_i = (r_i - r)T_i = \frac{c_i}{t}. \tag{7}$$

In a resource rich system, peers accumulate different amount of buffering levels proportional to their contributions. In a resource deficit system with small peer average bandwidth $\bar{u} < r$, the bandwidth supplies are not enough to sustain their normal playback demands and they need help from the server. In this case, it could be difficult for any peer to accumulate large buffering level and $t \to \infty$. How to adapt t with the system resource availability is crucial in the adaptive taxation scheme. The aggregate tax revenue $\sum r_i$ would be balanced with the budget expenditure $\sum c_i/T_i$. And the taxation ratio t would be determined by the system-wide resource availability. With the summation on both sides of Eq. (7), we obtain

$$t = \sum c_i / \sum \tau_i. \tag{8}$$

Due to peer dynamics and resource imbalance, it could be infeasible to collect all the information and tackle the issue in a centralized manner. A distribution protocol with the adaptive taxation is to be presented in the following sections.

IV. iPASS: System Design

In this section we present the detailed design of iPASS system.

A. Architecture

Similar to most deployed large scale P2P streaming systems, iPASS employs a *tracker* to keep track of peer arrivals and departures. The tracker maintains a list of active peers in the system. When a new peer joins in, it first contacts the tracker for an initial peer list. Then the new peer makes connections with peers on the returned list and starts to exchange signaling information and video data with them. Through signaling, peers exchange with their neighbors information about their buffering progresses, contribution levels and neighbor lists.

iPASS adopts the state-of-art pull based data exchange mechanism. A peer pulls video chunks from its neighbors by sending download requests. Furthermore, to avoid contention due to uncoordinated requests to the same peer, we introduce *pull tokens* for peers. Each peer periodically sends out pull tokens to its neighbors to give them permissions to pull chunks from it. The total number of tokens that one peer sends out is determined by the number of chunks that it can serve in each round. The number of tokens that a peer sends to a particular neighbor is determined by the contribution level of the neighbor, and is calculated by a distributed implementation of the ATB pre-fetching algorithm.

Due to asynchronous pre-fetching, a peer may become out-of-sync with its neighbors. Moreover, idle seeds and peers lacking enough number of suppliers may turn to find complementary neighbors. If so, to maintain the BPB peering, it needs to update its neighbors. A peer will find new neighbors by querying the tracker or searching through its neighbors' neighbor lists.

B. BPB Peering Implementation

The BPB peering makes the structured mesh scalable, robust, and also facilitates the support of VCR interactions.

1) *New Peer Join.* The key to BPB peering is to find peers with close buffering progresses. To facilitate BPB peering, the tracker sorts the list of active peers according to their arrival times. When a new peer joins in, the tracker records its arrival time and appends it to the end of the peer list. Then the tracker will return the new peer with an initial peer list consisting of

a number of random peers at the end of the list. Those peers will be the suppliers for the new peer.

2) *Dynamic BPB Peering*. When there is no pre-fetching, buffering on peers advances roughly at the same pace, namely the playback rate. Peers who arrive close in time will remain close in buffering progress. During the session, when a peer needs to connect to new neighbors, either due to neighbor departures or unsatisfactory peering connections, it can contact the tracker for additional peers. The tracker can quickly search through the sorted list to find peers with close buffering progresses for the requesting peer. In addition, due to BPB peering, a peer's neighbors' neighbors should also have close buffering progresses with the peer. Without going to the tracker, a peer can find new "close" neighbors in the neighbor lists returned by its neighbors.

With pre-fetching, bufferings on peers advance at different rates. A peer joins the system later can possibly download video faster than its neighbors who arrived earlier and gain larger buffering progress. Once this happens, the download rate of the peer will be slowed down due to the lack of enough suppliers. The peer should then trigger *dynamic BPB peering* to find more suppliers satisfying the BPB peering criterion. Fig. 3 shows a simple example of dynamic BPB peering. Towards the goal of downloading the whole video, node n_a runs on the "express track" with higher download speed, while its neighbors runs on the "local track" with lower download speed. As time evolves, it catches up with the buffering progress of its neighbors. To maintain its download rate, it then connects with peer n_1 with larger buffering progress and disconnects from peer n_5 with the smallest buffering progress.

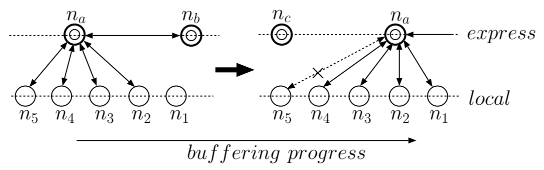


Fig. 3. Dynamic BPB peering

To facilitate this dynamic BPB peering, a distributed solution can be implemented as follows. Peers constantly exchange their buffering progresses with their neighbors. Due to dynamic BPB buffering, there is a good chance that a peer, even doing fast pre-fetching, can find peers ahead of it by searching through the neighbor lists returned by its neighbors. Then instead of requesting from the tracker, peers can request complementary peer lists from neighbors and choose appropriate peers with close buffering progresses to connect with. An alternative centralized solution is also possible, given a tracker with powerful process capability. The tracker needs to keep track of peers' buffering progresses and help peers to find new neighbors with close buffering progresses. This will incur a large volume of signaling and processing overhead on the tracker and peers.

3) *VCR Interaction Support*. In on-demand systems, users can freely trigger VCR operations. Next, we will show how to support VCR operations with BPB peering strategy.

- *Fast-forward and Rewind*. These two functions could be fulfilled by playing one out of v content segments, assuming that the fast-forward or rewind speed is v. Hence, peers in fast-forward mode stop pre-fetching sequentially and download one every v segments. As peers may have large download rate to pre-fetch content beyond its current playback range, peers can fast-forward without extra operations if the video content to be played has already been pre-fetched. On the other hand, peers have to locate new suppliers of the expected content in time when the content is beyond the current buffering progress and has not been downloaded yet. Unquestionably, peers can still effectively search suppliers with larger buffering progress along the structured mesh under dynamic BPB peering, as illustrated in Fig. 3. As for the rewind function, peers can certainly go back to playback what they have already watched previously since the video content would be stored in the cache. And peers can also conduct the BPB peering in reverse direction to locate suppliers for missing part, which has not been stored in cache due to previous random seeks or fast-forward interactions.
- *Pause and Resume*. Pre-fetching actually decouples the playback and download processes. In pause status, peers can simply stop playback but keep downloading the content, since the video content would be stored in the cache. And the playback can be continued after the resume command is requested.
- *Random Seek*. Peers can freely jump to any position of the video to begin playback via the random seek interaction, which calls for an indexing functionality of the system so that a peer can efficiently locate suppliers far away in time. As we discussed previously, a centralized solution is possible with a powerful tracker, which keeps track of peers' buffering progress and returns information of corresponding peers. In practice, we can set up multiple landmarks of playback progress positions with equal distance. When the buffering progress of a peer advances and passes such a landmark, the peer would report to the tracker with certain probability p. The system can adjust the probability to control the overhead of the cross-landmark signaling. In this way, the tracker can maintain and always update such a list of peers who just pass certain landmark playback positions. Whenever a peer conducts a random seek operation and the position it wants to jump to is too far away that the suppliers cannot be located quickly just by searching neighbors' peer list along the mesh, it would inquiry the tracker about peers in the list with associated landmark nearest to the position. And then it would search suppliers along the structured mesh via the returned peer as a portal. Finally, it would establish new connections with peers with close buffering progress corresponding to the specified position and tear down old connections.

The video content stored in a peer's cache might be discontinuous due to fast-forward and random seek interactions. However, it could hardly cause data availability problem in iPASS because peers exchange data only with neighbors with close buffering progresses. On the other hand, content segments may have also been pre-fetched ahead of peers' current buffering progress due to non-sequential playback with VCR interactions. After the missing part in between being downloaded, a peer adjust its buffering progress and buffering level to prevent duplicate download and incorrect token distribution in incentive mechanism, which is based on the buffering and contribution of peers. Fig. 4 illustrates how peer

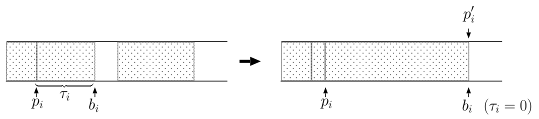


Fig. 4. Adjustment of progress indicators

i adjusts these progress indicators at the moment of the finish of the missing part download. The peer changes buffering progress b_i accordingly and locates appropriate suppliers via dynamic BPB peering. Meanwhile, the playback progress p_i is replaced with a *virtual playback progress* indicator p'_i. And the buffering level τ_i is reset to zero by letting p'_i equal to b_i at that moment. Furthermore, the contribution level c_i is also set to zero. By such settings, we can prevent the disturbance of the neighbors' pull token distribution determined by incentive mechanism. Then the peer would act as a new peer in the incentive mechanism. Similarly, whenever peers return to normal playback after any VCR interactions, the contribution levels should be reset to zero. And if the content at the new playback progress has been downloaded, the virtual playback indicator would be exploited and set to the same as the buffering progress. Especially, when users pause the playback, a virtual playback indicator would keep moving and the download process would not be interrupted.

C. Signaling between Neighbors

In iPASS, peers would frequently collect information from their neighbors and exchange data availability using *buffer-maps*. A buffer-map of a peer consists of a sequence of binary bits, each of which indicates the availability of one specific chunk on that peer. In live P2P streaming systems, due to the synchronous peer playback, at any time instant, the chunks that peers need to download fall into a small moving window covering several minutes worth of video. The buffer-map length can be kept short. In P2P file sharing, peers randomly download different portions of files. Buffer-maps have to indicate the data availability for the whole file. Similar to file sharing, peers in VoD systems are asynchronous. It is challenging to design VoD buffer-map to simultaneously achieve high utilization and low system overhead.

To address this issue, we define the *interested area* of a peer as the range of chunks that the peer is currently downloading. In the normal playback mode, peer i needs to retrieve chunks in their current playback range $[p_i, p_i + w_{rd}]$. Once all chunks in the playback range have been retrieved, it enters into the pre-fetch mode and starts to download chunks falling into its pre-fetching window. Therefore, the interested area of a peer is either its current playback range, if it is in the normal playback mode, or the pre-fetching window, if it is in the pre-fetch mode. Peers generate buffer-maps only for chunks in their interested areas. Furthermore, peers could only send the buffer maps to neighbors who have overlapping interested areas in order to reduce the signalling overhead. In addition, peer i sends to its neighbors the information on its buffering progress b_i, buffering level τ_i, and contribution level c_i.

D. Chunk Scheduling between Neighbors

Chunk scheduling determines the data flows among neighbors. iPASS employs pull-based scheduling design. We let $\psi(i)$ denote the set of receivers of peer i. After obtaining buffer-maps from its neighbors, a peer sends out pull requests to download missing chunks from its neighbors who have them. Due to distributed scheduling, peer i may receive multiple requests from peers in its receiver set $\psi(i)$. Some of the requests will be delayed or even disposed if peer i cannot fulfill all of them in time. To avoid contention, we introduce tokens to regulate pull requests from peers. Specifically, peer i periodically sends tokens to peers in set $\psi(i)$ to give them permission to pull data from it. The number of tokens that peer i sends is determined by how many chunks it can serve within each round. In the strategy without pre-fetching, the tokens of peer i is randomly distributed to peers in $\psi(i)$. In the pre-fetch mode, the token distribution should be conducted to maintain normal playback on all peers and enable differentiated pre-fetching based on peers' contribution. The ATB pre-fetching algorithm described in Section III-C is an ideal centralized solution, which cannot be implemented in a large system. Alternatively, we developed a distributed token distribution algorithm to realize ATB pre-fetching.

Algorithm 1: ATB Token Distribution on Peer i

```
  input : $\tau_k$, $c_k$, $\forall k \in \psi(i)$
  output: $P(k)$: fraction of tokens to peer $k$
1 $sum \leftarrow 0$
2 $t \leftarrow \sum_{k\in\psi(i)} c_k / \sum_{k\in\psi(i)} \tau_k$
3 for $k \in \psi(i)$ do
4     if $\tau_k \le w_{rd}$ then $e_k \leftarrow \max(w_{rd}, c_k/t) - \tau_k + 1$
5     else $e_k \leftarrow max(c_k/t - \tau_k, 1)$
6     $sum \leftarrow sum + e_k$
7 end
8 for $k \in \psi(i)$ do $P(k) \leftarrow e_k/sum$
```

The ATB token distribution algorithm is presented in Algorithm 1. Instead of assessing a universal tax ratio based on global information, peers deduce it locally based on information exchange with their neighbors. The tax ratio t calculated by peer i is the ratio between the aggregate buffering levels and the aggregate contribution levels of peer i's neighbors. The target buffering level $\bar{\tau}_k$ of a neighbor k is its contribution level c_k divided by t. Then peer i determines the *expected tokens* e_k to peer k as $\bar{\tau}_k - \tau_k$. ATB scheduling gives neighbors in the normal playback mode priority in access tokens. If a neighbor k's buffer progress τ_k falls behind the playback buffering threshold w_{rd}, peer i will give at least $w_{rd} - \tau_k$ tokens to peer k so that it can download chunks in the playback range. After

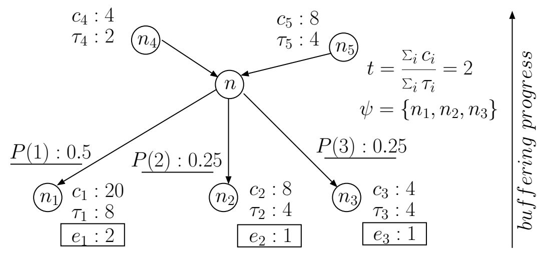


Fig. 5. Illustration of token distribution

calculating e_k for all its neighbors, the peer can determine the fraction of tokens for each neighbor in this round and then assign tokens according to the distribution. Fig. 5 illustrates an example of ATB token distribution, where w_{rd} is set to 1. Peer n has five neighbors and only n_1, n_2 and n_3 are its receivers. First, the tax ratio is calculated to be $t = 2$. Then, based on the buffering level and contribution level, the expected number of tokens e_k is calculated for each peer. Finally, the fractions of tokens sent to n_1, n_2 and n_3 are decided as 0.5, 0.25 and 0.25, respectively.

After a peer receives pull tokens from all its neighbors, it will decide which chunk is to be pulled from which neighbor. Various chunk requesting algorithms in live streaming can be applied. In a simplified manner, one can request missing chunks randomly from the neighbors which hold the chunk and also send the token. Tokens from a neighbor will be disposed if the peer does not send pull request to that neighbor in this round. This is to avoid disturbances to the efficiency of scheduling in future rounds.

V. Performance Analysis

In this section, we analyze the performance of iPASS system. First we try to derive the upper bound of server cost in iPASS system and then attempt to characterize the system behavior at steady state with a fluid modeling based approach.

To make the analysis tractable, we assume there are only two types of peers in the following analysis. The two types of peers have upload bandwidth u_a and u_b respectively, $u_a = \alpha r, u_b = \beta r$. And the node distributions of them are p_a and p_b. The average peer bandwidth is $\bar{u} = \sum u_i p_i$. The streaming rate is r and the playback duration of the video file on the channel is L. Without loss of generality, we assume $\bar{u} > r$, given the fact that the streaming rate of current IPTV system is commonly less than the average peer bandwidth.

A. Server Bandwidth Cost at Initial Startup Phase

During the initial startup phase, there is no seed in the system. The server has to provide content to the leading peers, who arrive at the system earliest, for their real-time playback demand. The server is the only source which can provide the content to the leading peers. As for the peers with smaller playback progress, they can pull content from the preceding peers. We are interested in how much server bandwidth is needed in iPASS system given certain peer distributions.

Property 1: At the system initial startup phase, the server bandwidth is required to boost up the system. The cost is almost surely bounded and determined by the distribution of the leading peers.

With BPB peering, peers can dynamically search the nearby preceding peers for appropriate suppliers. Given a reasonably large neighbor set size and rich system resources, peers can always find suppliers sufficient for their basic real-time playback demand except the leading peers. The peer bandwidth can be fully utilized by pre-fetching. The data flow backwards from the peers with larger buffering progress. Therefore, the aggregate bandwidth requirement of leading peers accounts for the server cost during the initial startup phase. And the distribution of the leading peers determines the amount of aggregate demand from the server.

We try to find an upper bound for the server cost in the following asymptotic way. We define a *supply safe* region formed by the leading Γ number of peers. The probability $P_\Gamma = P\{$the average bandwidth of the first Γ peers is larger than streaming rate $r\}$. If the average upload bandwidth of the peers in supply safe region is larger than the streaming rate, the server cost would be only incurred by the peers in the supply safe region, which would be less than Γr, the aggregate bandwidth necessary for directly streaming to them.

Suppose the numbers of leading peers with type a and b in the region are m and n, respectively, and $m + n = \Gamma$. We assume $mu_a + nu_b > \Gamma r$, which yields

$$m > \frac{1-\beta}{\alpha-\beta}\Gamma,$$

where $u_a = \alpha r$ and $u_b = \beta r$. The arrivals of two type of peers are assumed to be independent Poisson processes. Then we obtain

$$\begin{aligned} P_\Gamma &= \sum_{m > \frac{1-\beta}{\alpha-\beta}\Gamma} \binom{\Gamma}{m} (p_a)^m (p_b)^{\Gamma-m} \\ &= 1 - \sum_{m=0}^{\lfloor \frac{1-\beta}{\alpha-\beta}\Gamma \rfloor} \binom{\Gamma}{m} (p_a)^m (p_b)^{\Gamma-m} \\ &\simeq 1 - G\left(\frac{\lfloor \frac{1-\beta}{\alpha-\beta}\Gamma \rfloor - \Gamma p_a}{\sqrt{\Gamma p_a p_b}}\right). \end{aligned} \tag{9}$$

Given a Γ which lets $P_\Gamma > 1 - \epsilon$ with $\epsilon \to 0$, we can almost be sure that the average bandwidth of leading Γ peers would be larger than the streaming rate. Let $\epsilon = 0.05$, with 95% confidence interval, the maximum server cost is bounded by Γr. Assume $p_a = p_b = 0.5$ and $\alpha = 1.4, \beta = 0.8$, the average peer bandwidth is sightly larger than the steaming rate ($\bar{u} = 1.1r$), the value of Γ needs only to be larger than 30. The cost is small considering thousands of peers would join in typical IPTV systems.

In a resource rich system, the estimated bound for the system initialization phase may also be applicable to other system phases. Furthermore, peers that finish downloading the whole video content and still stay at the system, would become seeds to contribute. With the help from seeds, the system would become self-sustained without any need for server bandwidth. As to be presented in the simulation section, the server cost in the simulations is almost bounded at the initial phase and then drops to nearly zero as seeds appear.

B. System Equilibrium State

To characterize the system behavior in steady state, Qiu and Srikant [24] proposed a fluid modeling approach for file sharing applications. Taking into account the heterogenous peer types, our approach investigates the system from the perspective of the resource balance.

To make the analysis tractable, we assume peers will neither leave until the whole file is successfully downloaded nor conduct VCR interactions. Peers have option to leave immediately once they finish the downloading. And there are altruistic peers who would stay at the system to contribute until they finish the playback. We define the *selfish ratio* ρ^o to denote the probability of peers choosing to leave as soon as they finish download the whole video.

Suppose when the system enters the stable state, the expected numbers of seeds are W_a and W_b for type a and b, respectively. For the peers with types $V = \{a, b\}$, the expected time needed to download the whole file are assumed to be T^a and T^b. Let $\delta_i(x)$ denote the net contribution of a type i peer to the system, who stays at the system for duration x. The net contribution is determined by the contribution minus the consumption due to video download, which yields

$$\delta_i(x) = u_i x - rL.$$

Let θ^s and θ^u represent the expected net contribution of a selfish and unselfish peer, respectively. Then, we have $\theta^s = \sum_{i\in V} p_i \delta_i(T^i)$ and $\theta^u = \sum_{i\in V} p_i \delta_i(L)$. To balance the system resource, the net contribution of peers should follow

$$\rho^o \theta^s + (1-\rho^o)\theta^u = 0,$$

which yields

$$\rho^o = \frac{rL - \sum_{i\in V} u_i p_i L}{\sum_{i\in V} u_i p_i (T^i - L)}. \tag{10}$$

In light of the taxation incentive mechanism, the amount of buffered future content corresponds to the peer's contribution so far. For a peer who accomplished the file download with time T^i, the amount of buffered content is $(L - T^i)r$ and the amount of contribution is $u_i T^i$. For the two types of peers with homogenous taxation ratio among the system, we have

$$\frac{L - T^a}{L - T^b} = \frac{u_a T^a}{u_b T^b}. \tag{11}$$

With Equation(10)(11), we can deduce the expected peer file download time T^a and T^b provided with certain ρ^o.

Next we discuss the impact of ρ^o on the number of seeds in the system. We define the *dummy node* as the mirrors in the system corresponding to those leave the system as soon as they finish the video download. When one peer leaves the system before the playback finishes, one corresponding dummy node will be added to the system. These dummy nodes will stay at the system without contribution until the playback finishes. Let W_f^i denote the subset of type i peers those accomplished file downloading while still in playback at some time, which includes the dummy nodes. And W_u^i denotes the subset of type i peers, who are still in downloading process. Given that the number of type i seeds is W_i, we have

$$E[|W_f^i|] = W_i/(1-\rho^o).$$

Suppose the peer arrival rate is λ. According to the Little's Law, the expected number of peers in current system is $N = \lambda L$. For type i peer, we have

$$|W_f^i| + |W_u^i| = N p_i.$$

Since peers of type i can finish downloading in expected T^i and peers have steady download speed when the system is stable, the set of W_u^i and W_f^i peers can be regarded as the arrival peers in the time interval $[0, T^i]$ and $[T^i, L]$, respectively. The expected degree of the sets should follow

$$\frac{|W_f^i|}{|W_u^i|} = \frac{L - T^i}{T^i}. \tag{12}$$

Hence, we obtain the expected number of seeds of type i in the system

$$\begin{aligned} E[W_i] &= (1-\rho^o)|W_f^i| \\ &= (1-\rho^o)\frac{L - T^i}{L} N p_i. \end{aligned} \tag{13}$$

We validate the above results from the perspective of seed evolution and resource requirement. First, when the system is stable, the rate of turning into seed and seed departure should be equal. For type i node, there are $|W_u^i|$ number of peers possibly turning to seeds and the expect time for this process is T^i. Then the rate for peers to become seeds can be approximated by $|W_u^i|/T^i$. Similarly, the seed departure rate can be denoted by $W_i/(L - T^i)$. Taking into account the selfish ratio ρ^o, we obtain

$$(1-\rho^o)\sum_{i\in V} \frac{|W_u^i|}{T^i} = \sum_{i\in V} \frac{W_i}{L - T^i}. \tag{14}$$

The above equation holds true provided the calculated set degree and seed number based on Equation (12)(13). Second, the system should own enough resource to facilitate peers accomplish file downloading in the specified time. For type i node, the expected file download time needs T^i, then the average download rate would be rL/T^i. Therefore, we should have

$$N\bar{u} - \sum_{i\in V} \frac{\rho^o}{1-\rho^o} W_i u_i = \sum_{i\in V} \frac{rL}{T^i} |W_u^i|. \tag{15}$$

The left item denotes the summation of system resource excluding that of the dummy nodes. The right item denotes the current system demand, the product of peer number and download rate. By replacing the number with Equation (13), it is equivalent to Equation (10).

The above analysis provides several insights of the system behavior:

1) *The system has very good scalability*. From Equation (10)(11), we can observe that the average download time T^i is not related with either the peer arrival rate or the aggregate number of online peers.
2) *The system resource reduces as ρ^o increases*. From Equation (10)(13), we can deduce that the download time T^i increases and the number of seeds W_i decreases correspondingly as ρ^o increases.

To represent the aggregate seed upload bandwidth, we define the function $f_b(\rho^o)$ in terms of the selfish ratio

$$\begin{aligned} f_b(\rho^o) &= \sum_{i \in V} W_i u_i \\ &= \sum_{i \in V} (1-\rho^o) \frac{L - T^i}{L} N p_i u_i. \end{aligned} \tag{16}$$

Based on Equation (10)(11), we can deduce the aggregate bandwidth of seeds. This suggests that *at equilibrium state, the system can self-sustain itself without server bandwidth if the selfish ratio satisfies the constraint,* $\rho^o < f_b^{-1}(\Gamma r)$. Γr is the upper bound of the server resource demand in terms of the Property 1. If the aggregate seed bandwidth is larger than this demand, then no server bandwidth is necessary. Figure 6 plots the aggregate seed bandwidth under

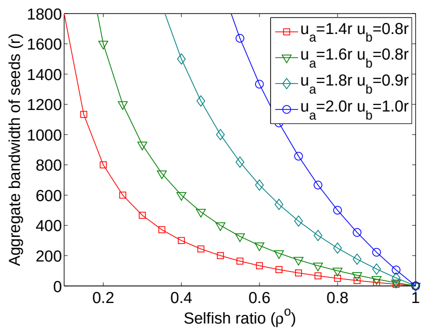


Fig. 6. Aggregate seed bandwidth vs. selfish ratio

two different node bandwidth distribution with various selfish ratios, where $p_a = p_b = 0.5, L = 1000, \lambda = 2$. We can observe as ρ^o increases, the resource of seeds decrease. The larger the average peer bandwidth, the more the corresponding seed resource we will obtain. When $\rho^o < 0.85$, the system can sustain itself without server bandwidth given $u_a = 1.4r$, $u_b = 0.8r$ and $\Gamma = 30$.

VI. Simulation Results

We use simulations to evaluate the performance of the proposed peering and pre-fetching strategies. *bpbp_np* and *ranp_np* refer to the BPB-peering and random peering strategies without pre-fetching respectively. *bpbp_inc* refers to our iPASS strategy, the combination of the BPB-peering with ATB pre-fetching. A random peering strategy with pre-fetching, denoted by *ranp_wp*, is also developed to make the comparison comprehensive.

A. Simulation Setup

We developed a packet-level event-driven simulator in C++ to study the performance of iPASS. Our simulator adopts the simulator engine of [25] simulating the end-to-end latency in terms of real-world latency measurement results. Two 4-CPU servers are employed to accelerate the simulations.

We follow the common consumption that peer download bandwidth is larger than the streaming rate and peer upload bandwidth is the only bottleneck. Three DSL types of nodes are assigned with upload bandwidth 1Mbps, 384kbps and

TABLE I
Normalized Peer Average BW and the Corresponding Fraction of Peer Types

ρ	Fraction of Peers (1M,384k,128k)	ρ	Fraction of Peers (1M,384k,128k)
0.90	$0.15, 0.39, 0.46$	1.40	$0.34, 0.52, 0.14$
1.00	$0.20, 0.40, 0.40$	1.50	$0.43, 0.38, 0.19$
1.12	$0.23, 0.46, 0.31$	1.60	$0.49, 0.36, 0.15$
1.20	$0.25, 0.53, 0.22$	1.70	$0.54, 0.32, 0.14$
1.30	$0.30, 0.50, 0.20$	1.80	$0.60, 0.30, 0.10$

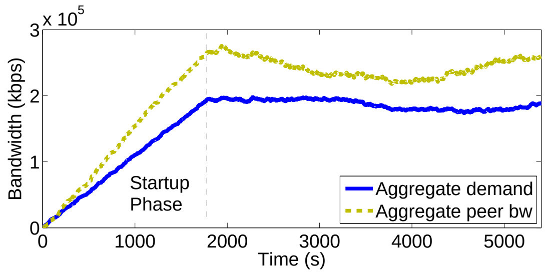
128kbps based on the measurement study [26]. The video streaming rate is 400kbps and each chunk has 5 KB size. We vary the fractions of three types of nodes to adjust the normalized peer average bandwidth, as shown in Table I. In the simulation, we use a single video with 30mins length. One single simulation round lasts for 90mins to get a better view of the system behavior. We believe that the video length and the simulation duration are already long enough to demonstrate the features of different strategies. The peer arrivals follow a Poisson process with arriving rate $\lambda = 1/4$ per second. The average number of online peers maintains around 500 after the startup phase and there are around $1,500$ peers joining the system during the whole session. The default number of neighbors of each peer is 15. The size of the playback buffering threshold and pre-fetching window are both 4 seconds. Peers broadcast buffer-map messages every 0.5 second and the token number information is piggybacked within the message. The server bandwidth cost consists of two parts, due to the complementary pull from peers for missing chunks and request scheduled from peers who receive the tokens from server respectively. The number of tokens sent out periodically from server corresponds to 1Mbps. To make the comparison fair, we generate the peer arrivals and upload bandwidth configuration beforehand and use the same setting to compare different strategies.

B. Numerical Results

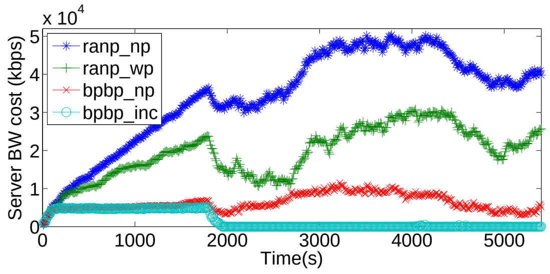
We first show the performance of various strategies on server bandwidth saving in the *linear viewing* mode. Peers will not leave the system before they finish the whole video playback. The results on differentiated pre-fetching are presented next. Then we study the performance with batch peer joins and early peer departures. At last, we compare the performance of iPASS with other P2P VoD systems.

1) Effectiveness on server cost saving: The server bandwidth saving is the most important performance metric to evaluate the different P2P strategies.

●**Server cost evolution illustration.** We begin by showing the evolution of server cost during one simulation session. Fig. 7(a) shows the instant aggregate user demand and the peer bandwidth when the normalized average peer bandwidth(ρ) equals to 1.3. There are no peers in the system at the beginning. The first peer finished playback and leave the system at $1,800$ second. The time period $[0, 1,800]$ is the *system startup phase*. Fig. 7(b) presents the instant server cost under the different strategies. We can observe that the server cost of random peering strategies increase almost linearly at



(a) Demand vs. resource



(b) Instant server cost

Fig. 7. Server cost under different peering strategies

the startup phase as the number of peers increases, then the curves oscillate closely with the instant peer average bandwidth. However, for BPB-peering strategies, it is interesting to observe that the server cost increases in a short period and maintains almost constant at the startup phase. Peers join the system early have limited data to share with each other. The server has to stream data to them directly. When more peers get into the system, peers start to download data from each other. When the startup phase is over, the server cost drops nearly to zero in $bpbp\_inc$ strategy. Later simulation results show that a certain amount of peers evolve into seeds can take the place of the server. Without pre-fetching, $bpbp\_np$ is also sensitive to the average peer upload bandwidth. It successfully controls the server cost at low level. We can find when $\rho = 1.3$, in the comparison of original streaming solution without P2P support, the random-peering without pre-fetching strategy ($ranp\_np$) can save at least around 75% server bandwidth. The saving can be improved to 85% with pre-fetching. With BPB-peering, the $bpbp\_np$ can enhance the saving further to around 95%. Moreover, $bpbp\_inc$ can sustain the system without server cost after the startup phase.

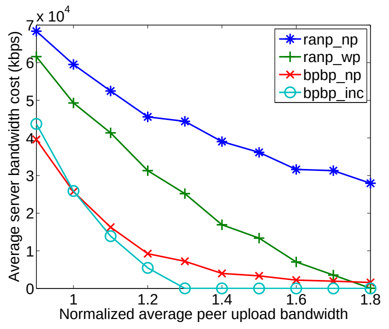
•**Performance with various peer distribution.** Next we examine the server cost savings with different normalized peer average bandwidth. Fig. 8(a) shows the average server cost after the first 50 mins. As the system resource increases, the cost of all strategies drops. $bpbp\_np$ and $bpbp\_inc$ both achieve most bandwidth saving. Especially $bpbp\_inc$ can sustain itself without server when $\rho > 1.2$. The system does not need server bandwidth with the assistance of seeds. *The BPB-peering can effectively improve the scheduling efficiency, which results in more server bandwidth saving.* Pre-fetching enables peers to download future content with extra bandwidth, thus reduces the possibility of data pull from the server in the future. The $ranp\_wp$ strategy with pre-fetching can also work without server when $\rho = 1.8$. When the normalized average bandwidth is 0.9, $bpbp\_np$ slightly outperforms $bpbp\_inc$. We believe this is because that pre-fetching potentially impairs some peers' normal playback when the whole system is in a bandwidth resource deficit status. This disadvantage can be conquered in iPASS by giving more preference to neighbors who haven't fill up the playback window during the scheduling.

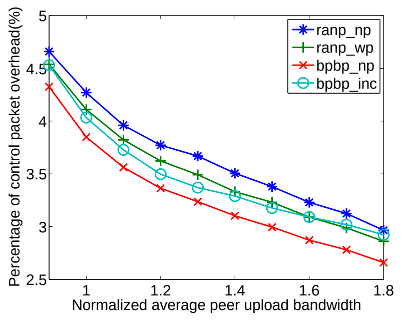
Although $bpbp\_np$ and $bpbp\_inc$ perform closely in terms of server bandwidth saving, pre-fetching of $bpbp\_inc$ produces seeds in the system. Fig. 8(c) illustrates the number of seeds during the simulation with normalized bandwidth equal to 1.5. It is very impressive that for $bpbp\_inc$ the seed number can even reach nearly 40% of all peers. In contrast, the ineffectiveness of random peering leads to fewer number of seeds in $ranp\_wp$. Seeds make the system resource allocation more flexible and thus more robust to peer dynamics. Furthermore, seeds can completely take the place of the server.

Peers only exchange the interested area information, which is efficient to keep the overhead low. Fig.8(b) shows the control traffic throughput compared with data traffic. The overhead contributes less than 5% percentage for all cases. As the resource increases, the exchange between peers become more effective with large enough bandwidth, which leads to less control overhead in return. The same phenomena can be observed between random peeing and BPB-peering strategies, because the latter is more effective than the former.

2) Impact of Differentiated Pre-fetching: To study the differentiated pre-fetching of peers, we collect the following peer information: the time to finish the whole video download and the amount of contribution at the moment of finish downloading. Fig. 9(a) plots the correlation between peer's download rate and contribution level as $\rho = 1.4$. The crosses closely scatter along the linear fitting line, which indicates larger contribution peers can finish download faster. Due to the limited simulation duration, the system does not enter the equilibrium state. That is the reason why the crosses do not form a strictly linear line. Peers in the deficit region are believed to be among the earliest batch of peers which can hardly find other suppliers to maintain the deserved download rate although they contribute a lot. As more and more peer become seeds, the download time of all peers decrease correspondingly. But the peers with larger contribution still finish sooner. The contributions of peers are limited by their upload bandwidth. Fig. 9(b) plots the cumulative distribution of the *seeding time* of different types of peers, which is defined as the duration from the time a peer finishes video downloading till its departure. The peers with zero seeding time are not counted. We can observe that larger bandwidth peers get longer seeding time. Peers with 1Mbps bandwidth have average seeding time of 18.6% of the video length, while the average seeding time for peers with 384kbps and 128kbps are 9.9% and 6.3% respectively. Differentiated pre-fetching enlarges the seed capability further by encouraging peers with larger bandwidth to become seeds earlier.

When the average peer bandwidth increases, peers can averagely spend less time to accomplish the whole file download. Meanwhile, due to the increased number and prolonged seeding time of seeds, peers would expect to contribute less to finish downloading. Fig. 9(c) presents the average download


(a) Average server cost in stable period


(b) Control traffic overhead

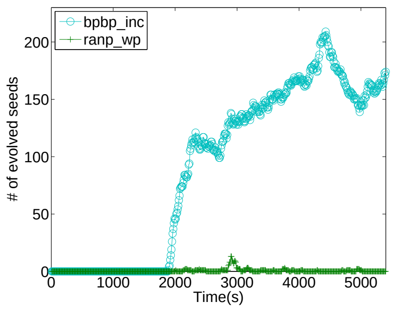
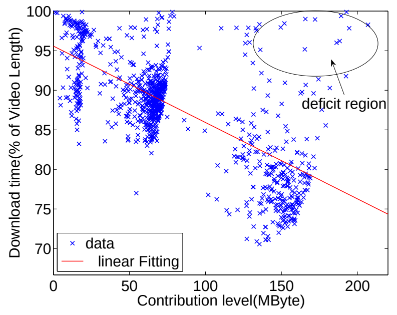
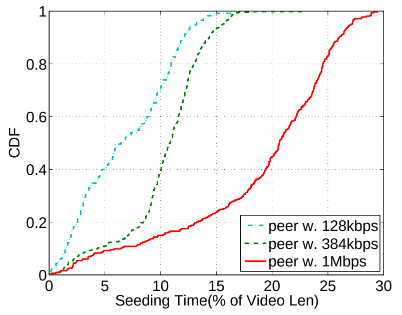

(c) Number of evolved seed

Fig. 8. System performance under various normalized peer bandwidth distributions


(a) Download time vs. contribution amount


(b) CDF of seed lifetime

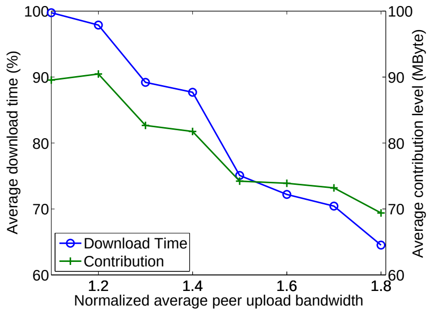

(c) Demand of peer contribution

Fig. 9. Impact of differentiated pre-fetching on peer download time

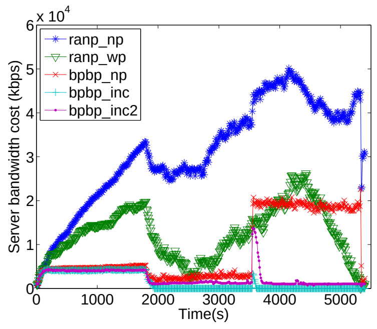
time and amount of corresponding contribution level of peers.

3) Robustness against System Dynamics: Simulations in previous sections assume peers are in linear viewing mode and only leave the system after they finish their playback. We now study the system's performance under different peer churn models. We start with the flash-crowd scenario where a batch of peers joins the system at the same time. As $\rho = 1.5$, 100 peers, almost 20% of the maximum number of online peers, suddenly join the system simultaneously around $3,600$ second. Fig. 10(a) shows the server cost evolution. Pre-fetching prevents the bandwidth cost of $ranp\_wp$ from jumping up significantly. However, there are big jumps in both non-pre-fetching schemes $ranp\_np$ and $bpbp\_np$. At the same time, $bpbp\_inc$ is highly robust against batch peer arrivals. There is only a small pulse in the server cost after the batch arrival. The server cost quickly goes back to zero afterward.

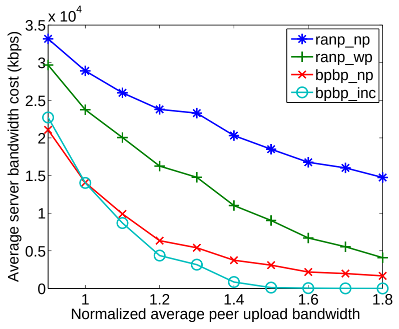
As an incentive in iPASS, peers are allowed to leave the system after they finish downloading the whole video. In that case, it cannot continue to stay as a seed to serve others. To verify the impact, we assume all peers are selfish and they will leave the system as soon as they finish the downloading. The system performance under this assumption is plotted in Fig. 10(a) using the curve denoted as $bpbp\_inc2$. The performance is close to $bpbp\_inc$. The peak value of the pulse due to batch join at around $3,600$ second is also less than $bpbp\_np$ and the pulse soon disappears as time evolves. The adaptive taxation scheme subsidizes large peers' bandwidth for small peers. Therefore the system still benefits a lot from the large peers' contribution before they finish the whole download.

Different from the linear viewing scenario, peers may also leave the system without finishing playback. We also study the impact on the system performance due to peer early departures. In this simulation, the peer lifetime follows a Weibull distribution. With Weibull distribution parameters $(1400, 4)$, peers leave the session gradually starting from the 271th second. And there are only 5.7% peers who will watch the whole video without early departure. Fig.10(b) shows the average server cost under various peer bandwidth distributions. The server cost of all strategies decreases almost half compared with Fig.8(a). This is because the number of simultaneously online peers decreases due to the early peer departures. But the relative performance order among different strategies remains similar. The $bpbp\_inc$ still achieves the best performance, and no server cost is needed when $\rho > 1.4$.

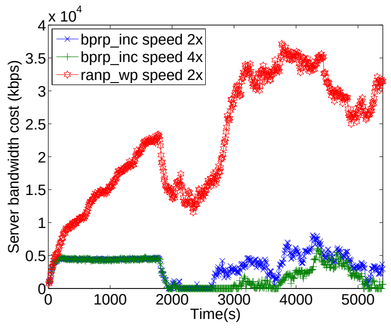
We also examine the system performance when users conduct VCR interactions, in particular, fast-forward. After the system startup phase (the first $1,800$ seconds), 10% of the new joining users are randomly picked to conduct the fast-forward operations until they leave the system. Fig. 10(c) presents the instant server cost at various fast-forward speeds when $\rho = 1.3$. Because peers randomly connect to others with $ranp\_wp$ strategy, part of the peer neighbors may have far large buffering progress and can serve them always. The VCR interactions do not bring much impact, and the peak value of server cost only increases by around 23% compared with that in Fig. 7(b) without VCR interactions. With dynamic BPB support, $bprp\_inc$ always maintains low server bandwidth cost. When those peers fast-forward faster with 4x speed,



(a) Batch join



(b) Early departure



(c) VCR interaction

Fig. 10. System performance under various peer dynamic scenarios

the number and demand of online fast-forward peers decrease accordingly, which leads to the decrease of server cost.

4) Comparisons with other VoD Systems: Without detailed system implementation and parameter settings, we cannot conduct head-to-head comparisons between iPASS and other P2P VoD systems. We qualitatively compare iPASS with some known mesh-based P2P VoD systems using performance numbers reported by the authors. Simulations of BiTos[10] assume all users arrive at almost the same time. They are not comparable with common asynchronous peer simulation setting. In PONDER[9] system, the server cost saving can reach 95% when $\rho = 2.9$. In [8], even the best approach with network coding cannot let chunk delivery ratio exceed 70%. The saving in BASS[18] can only reach 34% with their own setting. The results are all simulated in linear viewing scenario. While in iPASS, the system can sustain itself without server when $\rho > 1.2$. Moreover, with early departure, the system can also sustain itself when $\rho > 1.4$ with only 5.7% linear viewing peers.

VII. Conclusion

In this paper, we present the design of iPASS, a novel mesh-based P2P VoD system. iPASS achieves high peer bandwidth utilization at low system maintenance cost by adopting a dynamic buffering-progress-based peering strategy. To provide incentives for peer uploading, iPASS employs a differentiated pre-fetching design that enables peers with higher contributions pre-fetch content at higher speed. We further demonstrated that pre-fetching on peers can be coordinated by an adaptive taxation algorithm to simultaneously maintain system-wide QoE and provide service differentiations among peers with different contributions. We provided a performance analysis on the server cost bound and system behavior in steady state. Through detailed packet-level simulations, we showed that iPASS can efficiently offload server and achieve the desired balance between the system-wide QoE and service differentiations among heterogeneous peers.

References

[1] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, “I Tube, You Tube, Everybody Tubes: Analyzing the World’s Largest User Generated Content Video System,” in *Proceedings of Internet Measurement Conference*, 2007.

[2] “BitTorrent Homepage,” http://www.bittorrent.com/.

[3] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, “A measurement study of a large-scale P2P IPTV system,” *IEEE Transactions on Multimedia*, 2007.

[4] A. Sharma, A. Bestavros, and I. Matta, “dPAM: A distributed prefetching protocol for scalable asynchronous multicast in p2p systems,” in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2005.

[5] Y. Guo, K. Suh, J. Kurose, and D. Towsley, “P2Cast: Peer-to-peer Patching Scheme for VoD Service,” in *World Wide Web Conference*, 2003.

[6] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, “Challenges, Design and Analysis of a Large-scale P2P-VoD System,” in *Proceedings of ACM SIGCOMM*, 2008.

[7] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, “Push-to-peer Video on Demand system: design and evaluation,” in *IEEE Journal on Selected Areas in Communications*, 2008.

[8] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and D. Gunawardena, “Is High Quality VoD Feasible using P2P Swarming?” in *Proceedings of International World Wide Web Conference*, 2007.

[9] Y. Guo, S. Mathur, K. Ramaswamy, S. Yu, and B. Patel, “PONDER: Performance Aware P2P Video-on-Demand Service,” in *Proceedings of GLOBECOM*, 2007.

[10] A. Vlavianos, M. Iliofotou, and M. Faloutsos, “BiToS: Enhancing BitTorrent for Supporting Streaming Applications,” in *IEEE Global Internet Symposium*, 2006.

[11] Y.-W. Sung, M. Bishop, and S. Rao, “Enabling Contribution Awareness in an Overlay Broadcasting System,” in *Proceedings of ACM SIGCOMM*, 2006.

[12] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, “Substream Trading: Towards an Open P2P Live Streaming System,” in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2008.

[13] C. Huang, J. Li, and K. W. Ross, “Can Internet Video-on-Demand be Profitable?” in *Proceedings of ACM SIGCOMM*, 2007.

[14] B. Cheng, L. Stein, H. Jin, and Z. Zhang, “Towards Cinematic Internet Video-on-Demand,” in *Eurosys*, 2008.

[15] D. Wang and J. Liu, “A Dynamic Skip List-Based Overlay for On-Demand Media Streaming with VCR Interactions,” in *IEEE Transactions on Parallel and Distributed Systems*, 2008.

[16] X. Qiu, C. Wu, X. Lin, and F. C. Lau, “InstantLeap: Fast Neighbor Discovery in P2P VoD Streaming,” in *NOSSDAV*, 2009.

[17] Y. Cui, B. Li, and K. Nahrstedt, “oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks,” in *IEEE Journal on Selected Areas in Communications*, 2004.

[18] C. Dana, D. Li, D. Harrison, and C.-N. Chuah, “BASS: BitTorrent Assisted Streaming System for Video-on-Demand,” in *International workshop on multimedia signal processing (MMSP)*, 2005.

[19] M. Feldman and J. Chuang, “Overcoming free-riding behavior in peer-to-peer systems,” *SIGecom Exch.*, vol. 5, no. 4, pp. 41–50, 2005.

[20] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, “Free-riding and whitewashing in peer-to-peer systems,” in *PINS ’04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, 2004.

[21] Y. Chu, J. Chuang, and H. Zhang, “A case for taxation in peer-to-peer streaming broadcast,” in *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, 2004.

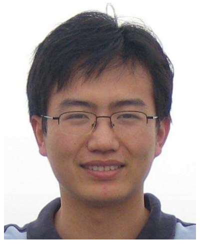
[22] A. Srinivasan, J. Teitelbaum, and J. Wu, “DRBTS: Distributed Reputation-based Beacon Trust System,” in *The 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC’ 06)*, 2006.

[23] S. D. Kamvar, M. T. Schlosser, and H. GarciaMolina, “The EigenTrust Algorithm for Reputation Management in P2P Networks,” in *Proceedings of International World Wide Web Conference*, 2003.

[24] D. Qiu and R. Srikant, “Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks,” in *Proceedings of ACM SIGCOMM*, 2004.

[25] M. Zhang, Q. Zhang, L. Sun, and S. Yang, “Understanding the power of pull-based streaming protocol: can we do better?” *IEEE Journal on Selected Areas in Communications*, 2007.

[26] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, “Analyzing and Improving a BitTorrent Networks Performance Mechanisms,” in *Proceedings of IEEE INFOCOM*, 2006.



Chao Liang (S’07) received his B.Engr. and M.Engr. degrees from Department of Electronic and Information Engineering, Huazhong University of Science & Technology (HUST), China, in 2000 and 2002, respectively. He is currently a Ph.D. candidate at the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, New York. His research interests include peer-to-peer networks, network optimization and content distribution, design and evaluation of algorithms and protocols.

Zhenghua Fu (S’00-M’04) is an assistant vice president in Automatic Volatility Trading group of Barclays Capital. He received Ph.D in Computer Science from UCLA in 2004. He was with IBM Research during 2006 to 2009 working on composition, sharing and collaboration of digital media. His current research interests include high speed networking and distributed systems, parallel computing, algorithms design and implementation.

Yong Liu (M’01) has been an assistant professor at the Electrical and Computer Engineering department of the Polytechnic Institute of NYU since March, 2005. He received his Ph.D. degree from Electrical and Computer Engineering department at the University of Massachusetts, Amherst, in May 2002. He received his master and bachelor degrees in the field of automatic control from the University of Science and Technology of China, in July 1997 and 1994 respectively. His general research interests lie in modeling, design and analysis of communication networks. His current research directions include robust network routing, Peer-to-Peer IPTV systems, overlay networks and network measurement. He is the winner of the IEEE INFOCOM Best Paper Award in 2009, and the IEEE Communications Society Best Paper Award in Multimedia Communications in 2008. He is a member of IEEE and ACM.

Chai Wah Wu (S’88-M’96-SM’00-F’01) is currently manager of the Distributed Digital Media Infrastructure group at IBM T. J. Watson Research Center in Hawthorne, NY. He received a B.A. degree in Cognitive Science from Lehigh University and a M.A degree in Mathematics and a Ph.D. degree in Electrical Engineering from the University of California at Berkeley and has worked on various research topics including image halftoning, multimedia security and synchronization and consensus in networks of coupled systems. He has authored over 100 papers and over 50 issued patents. He is author of the books ”Synchronization in coupled chaotic circuits and systems” and ”Synchronization in complex networks of nonlinear dynamical systems” (World Scientific, Singapore) and has served as associate editor of IEEE TCAS for 4 terms. He was elected Fellow of the IEEE in 2001.