Time-Shifted Streaming in a Tree-Based Peer-to-Peer System

Jeonghun Noh
ASSIA inc., Redwood City, CA, USA
Email: jnoh@assia-inc.com

Bernd Girod
Information Systems Laboratory, Department of Electrical Engineering
Stanford University, CA, USA
Email: bgirod@stanford.edu

***Abstract*— We propose a peer-to-peer (P2P) streaming system that multicasts live video as well as provides the live video as Video-on-Demand (VoD) during the live session. The proposed system allows users to individually pause and resume a live video stream, and to view video that was streamed before the particular user joined the session. Since live video content naturally results in many users watching it online at the same time, the P2P concept is leveraged to reduce server load.**

We extend our Stanford Peer-to-Peer Multicast (SPPM) protocol, originally designed for live video multicast, to support playback control. To achieve this in a P2P fashion, peers store received video packets in their local buffer and forward them at a later time when requested by other peers, which is called *time-shifted streaming*. To understand interaction between live and time-shifted traffic, we analyze video availability of the proposed system, which refers to how many peers possess video contents of a particular position. Motivated by the analysis, we propose *fast prefetching* and a parent selection scheme suitable for fast prefetching in order to further reduce server load by disseminating video quickly. Fast prefetching is possible because the relaxed transfer deadline of time-shifted streams and peers' extra uplink bandwidths are exploited. With simulations and mathematical analysis, it is shown that fast prefetching not only alleviates the requirement of server bandwidth by expediting video dissemination, but also mitigates video disruption due to peer churn.

***Index Terms*— P2P streaming, video-on-demand, time-shifted streaming, playback control**

I. INTRODUCTION

Two primary forms of video streaming are live streaming and video-on-demand (VoD). Live streaming often implies multicast communications as many users concurrently watch video. Video-on-demand, on the other hand, has been achieved by unicast communications because the temporal correlation among video segments users watch tends to be low. In this paper, we propose a P2P streaming system that allows viewers to individually pause a live video stream, rewind (even to contents prior to their time of joining), and jump forward (to contents beyond the current point of playback), which allows for the VoD of a live stream. As viewers watch the same content at a different time in a rather short time scale, the proposed P2P system exploits high temporal correlation among users' viewing points in time. The potential benefits from high temporal correlation among user arrival times for P2P VoD systems are well studied in [1]. In the proposed system, peers store received video in their local buffer. If a peer rewinds or jumps forward to content not present in its local buffer, the required video segment is retrieved from other peers. The video, buffered at some peer(s), is then streamed to the requesting peer asynchronously with respect to the live stream emanating from the server. To keep track of the video contents at peers, the peer database at the source peer is augmented with additional information of peers. The protocol is changed to be able to handle additional fields, such as a video position or a time stamp. This is called *time-shifted streaming*.

As mentioned earlier, the concept of time-shifting in a live session is closely related to asynchronous video streaming in VoD systems. VoD systems allow users to select and watch video content on demand. With playback control, also known as *trick mode*, users can also pause a video stream, rewind to an arbitrary position, and fast forward. Unlike live video, video files requested on-demand are pre-encoded and stored at video servers. If the available uplink bandwidth at the sender exceeds the video bitrate, the user can receive video faster than the play-back speed [2]. Traditionally VoD services were provided by servers. For instance, Chaining [3] and Patching [4] employ media servers and IP multicast, where IP multicast is used to reduce server workload. The authors of [5] study trade-off between the system profit and user experiences in a near-VoD system. In [6], patching and batching of server-based VoD service are augmented by taking into consideration heterogeneity in receivers.

As in server-based live streaming, servers deployed for VoD services may fail in the presence of a large population of users. Thus, many P2P-based VoD systems have been proposed to achieve scalable services. In order to achieve low workload on servers, the resources in the P2P network are leveraged for efficiently taking

Jeonghun Noh was with Stanford University, Stanford, CA 94305 USA. Manuscript received February 15, 2011; revised July 15, 2011; accepted September 30, 2011.

doi:10.4304/jcm.7.3.202-212

advantage of uplink bandwidth and storage contributed by peers [7]–[10]. For VCR-like operation (temporal random access), either distribution overlay is combined with look-up overlay [7], [8], or a separate look-up overlay is built to avoid a central look-up service [11]–[13]. In [9], [14], [15], viewers are able to watch a pre-recorded video from the beginning of the content. Dynamic Skip List [8] and oStream [7] provide users random temporal access to video content. The mesh-based P2P VoD systems are studied in [16]–[18]. The authors of [15] exploit the extra uplink bandwidths and peer storages by downloading video faster than playback speed. In [19], a user-log based prefetching algorithm is proposed. PPB, Peer-to-Peer Batching, combines peer-to-peer and IP multicast by using IP multicast for batching while a peer-to-peer network is used to provide the initial portion of video to new peers [20]. In [21], the authors propose a tree-based system that constructs dynamic multicast trees, called J-tree. The J-tree structure explicitly distinguishes the live multicast tree and playback trees that are implicitly built in our system. DRPSS, presented in [22], also supports live and time-shifted streaming in a single system based on the Distributed Hash Table overlay. P2TSS [23] supports time-shifted streaming in a live broadcast. In P2TSS, peers have two separate buffers; a buffer for local video playback and a shared buffer for serving time-shifted streams to other peers. The contents of the shared buffer do not change once the buffer becomes full of the received video. Unlike P2TSS, our system requires a single buffer to store the incoming video stream regardless of its shift in time (e.g., live or time-shifted stream).

The remainder of the paper is structured as follows. Section II introduces the Stanford Peer-to-Peer Multicast (SPPM) protocol [24] . In Section III, SPPM is extended to support time-shifted streaming. Section IV presents the analysis of video availability, which represents how many peers possess video contents of a particular position. In Section V, we describe fast prefetching to improve video availability. Fast prefetching allows peers to disseminate video faster. To facilitate video dissemination with fast prefetching, a new parent selection algorithm, called *maximum throughput*, is proposed and its effects are demonstrated with extensive simulations in Section VI.

II. Stanford Peer-to-Peer Multicast

Stanford Peer-to-Peer Multicast (SPPM) is a P2P system that achieves low-latency and robustness in live video multicast. Like in [25], [26], SPPM adopts the push approach for low-latency data dissemination. The push approach requires a persistent route among peers. This route is built by connecting a pair of peers using unicast connections, such as TCP or UDP. Since unicast connections are established on top of the IP layer, the collection of the unicast connections is often called an *overlay*. Each data path, a chain of peers in the overlay, starts from the server. On the data path, all intermediate peers act as both parents and children; peers that relay video packets are the *parents* of the peers that receive the packets. Peers that receive video packets are the *children* of the peers that relay the packets. When a parent peer can relay packets to more than one child peer, then the path naturally evolves from a chain to a tree structure, with the server being the root and peers being intermediate nodes and bottom nodes. Note that the desired distribution structure becomes a spanning tree because no packet needs to visit the same peer more than once.

Fig. 1 illustrates an example overlay in which two complementary trees are constructed among a small group of peers. Typically, we use 4 to 8 trees, and peer groups might be much larger [27]. As peers join the system, the trees are incrementally constructed in a distributed manner. When a new peer contacts the video source, the video source replies with session information, such as the number of multicast trees and the video bit rate. It also sends a list of candidate parents randomly chosen from the table of participating peers it maintains. The new peer then probes each candidate parent to know about their current status. After receiving probe replies, the best candidate parent is selected and contacted for each tree by minimizing the height of the distribution tree. Once the selected candidate parent accepts the attachment request, a data connection is established between the parent and the new peer. After data transmission starts, each child peer periodically sends *hello* messages to their parents. When a peer leaves the system ungracefully, its parents detect it by observing consecutive missing *hello* messages, and stop forwarding video to the child. The departing peer's children notice that neither video packets nor a response to *hello* messages arrive. Each abandoned child then initiates procedure to connect to a different parent node in the tree.

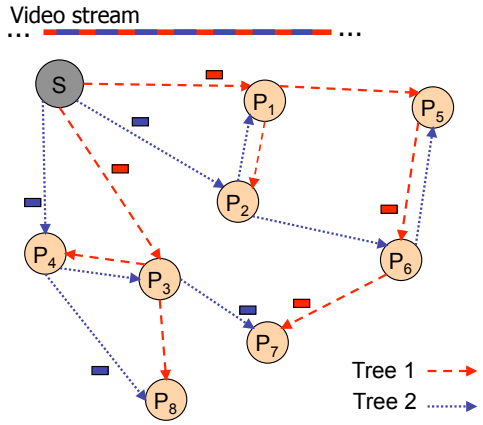

Figure 1. A peer-to-peer overlay consisting of two complementary distribution trees. The encoded video is packetized and split into disjoint substreams, one for each distribution tree.

To provide the best video quality despite congestion and peer churn, SPPM employs sender-driven packet scheduling [28] in conjunction with receiver-driven retransmission requests [29]. Acting as a relay, each peer schedules the next packet to forward by comparing the importance of packets in its output queue. As a receiver,

each peer evaluates the importance of missing packets by computing the expected video quality improvement if the corresponding frame is received before its playout deadline. Retransmission requests of selected missing packets are then sent to alternative parents. This NACK-based retransmission is advantageous as packet error rate is often not stationary over time nor uniform among peers. Since packet losses typically occur in rare bursts, such feedback error control is generally superior to forward error control. Feedback implosion, prohibiting feedback error control in network-layer multicast, is not an issue for a P2P overlay because retransmissions are handled locally between a parent and a child peer.

III. Time-Shifted Streaming

In this section, SPPM is extended to allow users to control their playback in the time domain [30]. With playback control, also known as *trick mode* in video-on-demand systems, peers can individually pause a live video stream, rewind to an arbitrary position (even to contents prior to their time of joining), and fast forward until the latest position of the live video. In the extended SPPM, peers cache the received video in their local buffer. For pause/resume, the locally cached video can be played back at a later time. When a peer rewinds or jumps forward to content not present in its local buffer, the requested video segment, buffered at some peer(s), is streamed to the peer asynchronously unlike the live stream being multicast to peers. This is called *time-shifted streaming*. As we will see later, SPPM supports both the live stream and time-shifted streams while keeping the system's scalability.

While extending the SPPM system for time-shifted streaming, the fundamental architecture of the system is kept intact so that live video multicast is seamlessly supported. In this section, we describe only the system extensions added to the existing architecture.

A. Preliminaries

As in live stream multicast, the video server (source peer) starts to stream a live video starting at time 0. Let $V(x)$ denote a video frame associated with position x in time, $x \geq 0$. The live video emanating from the server at time t is thus represented by $V(t)$. A time-shifted stream is represented by $V(t-d)$ with delay d, $0 < d \leq t$, at time t. Suppose that a new peer or an existing peer requests video $V(x)$ at time t. If the peer requests the live video $V(t)$, the peer is called *LS* peer. If the peer requests a time-shifted stream $V(t-d)$, $0 < d \leq t$, then the peer is called *TS* peer. For LS peers, the live video is immediately relayed from peer to peer. For TS peers, the live stream must be delayed by time d somewhere in the system, either at the server or at the peers. Delayed streaming, or *time-shifted streaming*, is achieved by storing past video packets in peers' buffers while they watch the video[1] and transmitting them when requested.

The time-video plot in Fig. 2 illustrates the trajectories of the live stream, Peer 1 (LS peer), and Peer 2 (TS peer). The trajectories indicate the video contents stored at the peers over time. The vertical axis represents the time stamp of video frames (a time stamp is the time when a video frame is encoded at the server). In Fig. 2, Peer 1 requests the live video $V(t_1)$ at time t_1. Its video trajectory overlaps the live stream trajectory. At time t_2, Peer 2 requests the video starting from $V(x_2)$. Since the requested video is stored at Peer 1, Peer 2 can obtain the video delayed by $t_2 - x_2$ from Peer 1.

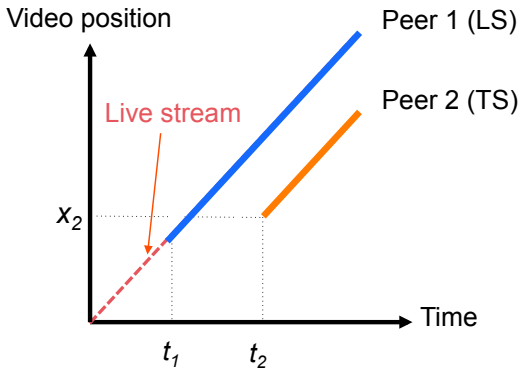

Figure 2. Video trajectories of live video, Peer 1, and Peer 2. Peer 1, an LS peer, requests the live video. Peer 2, a TS peer, requests the video delayed by $t_2 - x_2$.

B. Peer Database

In the system, peers collectively form a distributed video storage. To facilitate the use of the videos stored by peers, locating video among peers is essential. To locate the video of a particular position, the video server maintains a database comprising each peer's identifier, the time the video is requested, the time stamp of the initial video frame, and the time stamp of the latest video stored in the peer's buffer. The latest time stamp is estimated by the server based on each peer's initial video position and request time. Estimation by the server reduces the amount of control traffic between the server and peers by truncating the number of buffer state updates. To prevent the estimation from deviating too far from the actual buffer states, peers may report their buffer state intermittently (see Section V-A).

C. Video Request and Connection Set-up

When a new peer joins or an existing peer seeks a video of a particular position, it sends a query requesting a certain video position to the server. The server refers to the peer database it maintains. It returns a list of randomly chosen parent candidates whose video buffer may contain the requested (possibly time-shifted) video. The peer then probes each parent candidate to learn about the candidates' current status, including time stamps of the stored video and available bandwidth. After the peer receives probe replies, it selects and requests for each

[1] A peer shares its storage only when it participates in a session. For a typical two hour-long video stream encoded at 600 kbps, about 540 MB of storage is required. This is a moderate space requirement considering today's typical personal computers.

multicast tree the best parent candidate that will accept it as a child. We discuss parent selection criteria in detail in Section V. Once the parent candidates accept the video request, data connections are established between the parents and the child.

D. Asynchronous Video Transmission

Connections established for time-shifted streaming are different from connections that constitute the live video multicast trees. In the latter case, packets are immediately relayed to the next hop in the multicast tree, whereas in the former, packets are asynchronously relayed over the time-shifted streaming connection. Multiple complementary trees are the basis of data dissemination in SPPM. As LS peers, TS peers receive packets from multiple parents. Parents asynchronously relay video packets to their TS peers by ensuring that a packet travels along the tree designated to itself.

A parent peer dedicates a streaming window for each TS child peer. Fig. 3 shows a streaming window of a TS child that requests video from position x_1 at time t_1. The lower end of the window, denoted by W_L, is the oldest video packet the parent can transmit. Since packets that are past their display time (called the play-out deadline) are not decoded at the child, the value of W_L is set to the minimum of the initial video position in time and of the time stamp of the video frame displayed at the child. The upper end of the window, denoted by W_U, corresponds to the time stamp of the latest video packet the parent can send. The value of W_U is selected in such a way that the child's downlink is not congested by the parent, and the child does not suffer from buffer underflow. The parent regularly searches for packets whose time stamps are between W_L and W_U in its video buffer. In addition, only packets that belong to the parent's tree ID are selected to construct the corresponding substream.

To avoid duplicate transmission, the history of transmitted packets is maintained for a finite period of time. Selected packets are marked with their play-out deadline and put into the network output queue. The output queue sorts video packets according to their play-out deadline. Therefore, urgent packets, such as retransmitted packets or live video packets, are sent earlier than packets with a relaxed deadline, such as packets in time-shifted streams.

At a child peer, packets that belong to different substreams may arrive out-of-order. Once received, packets are ordered according to their positions in the video stream before being passed to the video decoder.

E. Recovery from Disconnect

After video transmission starts, each child peer periodically sends *hello* messages to its parents. When a peer leaves the system ungracefully, its parents detect this by observing consecutive missing *hello* messages and stop forwarding video to the child. The departing peer's children notice that neither video packets nor responses to *hello* messages arrive. Each abandoned child then initiates a recovery procedure to connect to a different parent node in the tree. During the recovery procedure, LS peers request the latest portion of the video in order to keep end-to-end transmission delay as low as possible. TS peers, on the other hand, request the next position of the video they received if the position was ahead of their playback position. Otherwise, TS children request the playback position of the video.

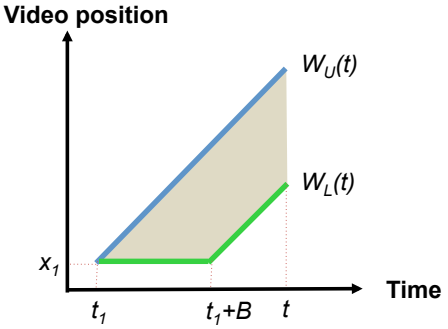

Figure 3. A streaming window for a TS child that requests video from position x_1 at time t_1. W_L is its lower end and W_U is its upper end. B is the pre-roll delay of play-out at the child. Packets whose time stamps fit in the streaming window are transmitted unless they have been sent already.

IV. Analysis of Video Availability

This section presents the analysis of the availability of time-shifted video segments among peers, which is closely associated with system scalability. High video availability lessens the need for making a connection to the server, thereby reducing server load. The analysis also sheds light on the interaction between live and time-shifted streams. Suppose that peers join the system according to a Poisson process $N(t)$ with rate λ [31]. Their lifetime follows an exponential distribution[2] with parameter μ ($1/\mu$ is the average peer lifetime). We assume that the probability that the next peer is an LS peer follows a Bernoulli process with parameter α. When a TS peer requests a video, its video access pattern is assumed to be uniform. This assumption allows us to be free from any particular video content and it provides the worst-case study because overlaps between peers' buffered contents are minimized, which results in minimizing the video availability. Based on this assumption, the position x is selected, at time t, uniformly between 0 and t. We further assume that a peer always receives the video it requests, by connecting either to another peer or to the video server.

Let $M(t, x; \lambda, \mu, \alpha)$ denote the number of peers possessing the video at position x at time t. To analyze the

[2] A peer lifetime is known to follow a long-tail distribution, such as the Zipf distribution [27], [32], [33]. However, the exponential distribution is still a popular choice for both analysis and simulation study because it reasonably approximates the actual distribution. More importantly, this assumption allows the system analysis to be tractable, and the system behavior can be predicted by observing only a few primary system parameters.

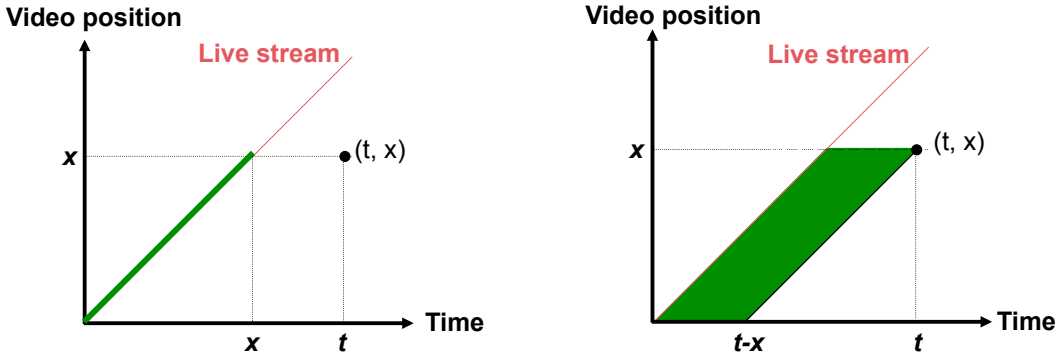

Figure 4. Initial video positions and request times for LS peers (Left) and TS peers (Right) that possess video at position x at time t. Left: An LS peer whose arrival time is earlier than x has video at position x at time t. Right: A TS peer whose request time and initial video position are within the area surrounded by $(0,0)$, $(t-x,0)$, (t,x), and (x,x) on the time-video plot has video at position x at time t.

availability of video contents among peers, we compute the expected value of $M(t,x;\lambda,\mu,\alpha)$, given by

$$\begin{aligned} &E\{M(t,x;\lambda,\mu,\alpha)\} \\ &= \sum_{k=1}^{\infty} k \Pr\{M(t,x;\lambda,\mu,\alpha)=k\} \\ &= \sum_{k=1}^{\infty} k \sum_{i=k}^{\infty} \Pr\{M(t,x;\cdot)=k|N(t)=i\} \Pr\{N(t)=i\}, \end{aligned} \tag{1}$$

where $N(t)$ represents the number of peers that have arrived at the system between 0 and time t. Given $N(t)=i$, since $N(t)$ is a Poisson process, peers' arrival times $S_1, \ldots, S_i$, considered unordered random variables, are distributed independently and uniformly in the interval $[0,t]$ [34]. This allows us to study each peer's behavior independently regardless of peers' join sequence. Consider a peer that joins the system between 0 and t. Let p denote the probability that the peer is still present and possesses video at position x at time t. Then, $E\{M(t,x;\lambda,\mu,\alpha)\}$ in (1) can be rewritten as

$$\begin{aligned} &E\{M(t,x;\lambda,\mu,\alpha)\} \\ &= \sum_{k=1}^{\infty} k \sum_{i=k}^{\infty} \binom{i}{k} p^k (1-p)^{i-k} e^{-\lambda t} \frac{(\lambda t)^i}{i!} \\ &= \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \left(\frac{p}{1-p}\right)^k e^{-\lambda t} \sum_{i=k}^{\infty} \frac{((1-p)\lambda t)^i}{(i-k)!} \\ &= \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \left(\frac{p}{1-p}\right)^k e^{-\lambda t} e^{(1-p)\lambda t} ((1-p)\lambda t)^k \\ &= e^{-p\lambda t} \sum_{k=1}^{\infty} \frac{(p\lambda t)^{k-1}}{(k-1)!} p\lambda t \\ &= p\lambda t. \end{aligned} \tag{2}$$

We compute p by considering the cases of TS peer arrivals and LS peer arrivals separately. Suppose that a peer arrives at time s, $s \le t$. If it is a TS peer, then it contains video at position x at time t only when its arrival (or request) time and its requested video position are inside the area surrounded by $(0,0)$, $(t-x,0)$, (t,x), and (x,x) on the time-video plot in Fig. 4. One can easily check this by drawing a video trajectory (in parallel with the live stream trajectory) from any point inside the area until time t and examine whether the trajectory passes or touches the horizontal line drawn from $(0,x)$ to (t,x). Note that the video is not available above the live stream trajectory. Let $g(s;t,x)$ denote the probability that a TS peer joining at time s contains the video x at time t. The value of $g(s;t,x)$ is 1 when $0 \le s \le (t-x)$, $(t-x)/s$ when $(t-x) < s \le x$, and $(t-s)/s$ when $x < s < t$, respectively. Based on the exponential distribution of peer lifetime, the probability that the peer is still present at time t is $e^{-\mu(t-s)}$. Then, the probability that a TS peer that contains video at x at time t, $l(t,x;\mu)$, is

$$\begin{aligned} l(t,x;\mu) &= \int_0^t e^{-\mu(t-s)} g(s;t,x) \frac{1}{t} ds \\ &= \frac{e^{-\mu t}}{t} \left\{ \int_0^{t-x} e^{\mu s} ds + (t-x) \int_{t-x}^{x} \frac{e^{\mu s}}{s} ds \right. \\ &\quad \left. + \int_x^t e^{\mu s} \frac{t-s}{s} ds \right\}. \end{aligned} \tag{3}$$

When an LS peer arrives at time s, it possesses video at position x at time t only when $s \le x$. The set of the pairs of the live video position and arrival time is the line segment from $(0,0)$ to (x,x) shown on the left side of Fig. 4. Then, by incorporating the likelihood of a peer's departure before time t, the probability that an LS peer that contains video at x at time t is expressed as

$$\int_0^x e^{-\mu(t-s)} \frac{1}{t} ds = \frac{e^{-\mu t}(e^{\mu x}-1)}{\mu t}. \tag{4}$$

Finally, p can be written as

$$\begin{aligned} p &= \Pr\{L^c\} \Pr\{x \text{ is possessed at time } t | L^c\} + \\ &\quad \Pr\{L\} \Pr\{x \text{ is possessed at time } t | L\} \\ &= (1-\alpha) l(t,x;\mu) + \frac{\alpha e^{-\mu t}(e^{\mu x}-1)}{\mu t}, \end{aligned} \tag{5}$$

where L is the event that an LS peer joins the system. By

inserting (5) into (2), we rewrite (2) as

$$E\{M(t,x;\lambda,\mu,\alpha)\} = (1-\alpha)\lambda tl(t,x;\mu) + \alpha\frac{\lambda}{\mu}e^{-\mu t}(e^{\mu x}-1), \quad (6)$$

where $(1-\alpha)\lambda tl(t,x;\mu)$ corresponds to the expected number of TS peers that possess video at position x at time t and $\alpha\frac{\lambda}{\mu}e^{-\mu t}(e^{\mu x}-1)$ corresponds to the expected number of LS peers that possess x at time t.

In (6), the video availability is predicted to increase linearly as the peer's join rate λ increases. This result demonstrates that the system is self-scalable. The availability of the early portion of the video continues to decrease as $t \to \infty$, which indicates that time-shifted streams have to be provided by the video server at a later time. In Fig. 5, the video availability is predicted by evaluating (6) with λ = 1.125 joins per second, $1/\mu$ = 120 seconds, and α = 0.5. In the figure, the video near the live stream is available at many peers, the majority of which are LS peers. As time progresses, more video frames become available and the overall video availability decreases.

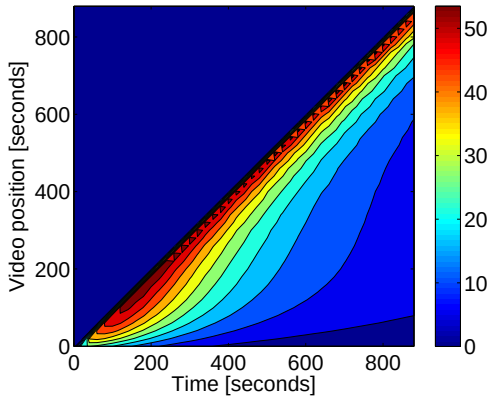

Figure 5. Video availability on the time-video plot. The value at (t,x), predicted by the model, represents the expected number of peers available for video at position x at time t (λ = 1.125 joins per second, $1/\mu$ = 120 seconds, and α = 0.5).

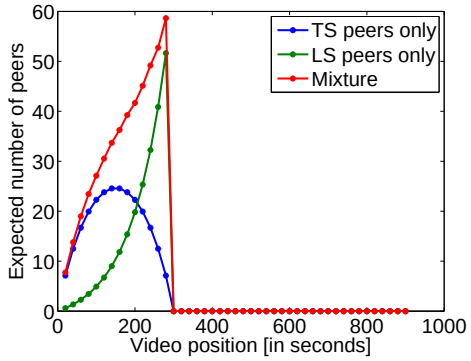

Figure 6. Model: video availability at 300 seconds. (λ = 1.125 joins per second, $1/\mu$ = 120 seconds, and α = 0.5)

Figs. 6 and 7 illustrate the expected video availability at 300 and 900 seconds, respectively. In these figures, the contribution from LS peers and TS peers is separately shown. The expected number of TS peers having video at position x at time t peaks when $x = t/2$, whereas the expected number of LS peers at time t peaks at the live video position of $x = t$ and its value is $\lim_{t\to x}\alpha\frac{\lambda}{\mu}e^{-\mu t}(e^{\mu x}-1) \approx \alpha\lambda/\mu$. Although the videos stored at either LS or TS peer have high overlaps at time 300s, the overlap dramatically decreases at time 900s. This result implies that at a later time of the session more TS peers have to connect to the server, which was already observed in Fig. 5.

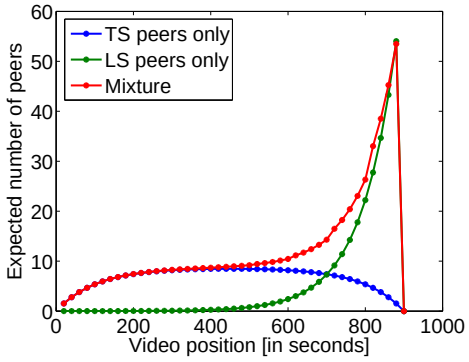

Figure 7. Model: video availability at 900 seconds. (λ = 1.125 joins per second, $1/\mu$ = 120 seconds, and α = 0.5)

V. Fast Prefetching

In the previous section, we have observed that the video availability decreases over time. Server load is highly affected by video availability because higher video availability allows more peers to obtain video from other peers, thus reducing the number of connections made to the server. In this section we propose *fast prefetching*, which allows peers to download video more rapidly. Fast prefetching exploits the relaxed transfer deadline of time-shifted streams when compared to the live stream. As an input device such as a video camera generates a video signal, the live video is immediately encoded and transmitted to peers. On the other hand, time-shifted streams are the past portion of the video, already encoded and stored somewhere in the system. Fast prefetching also exploits the extra uplink bandwidth of peers.

A. Improving Video Availability

Suppose a TS child receives a time-shifted stream from its parent. When its parent has sufficient uplink bandwidth, the stream can be pushed to the child faster than playback speed. Since fast prefetching facilitates video dissemination among peers, video availability can be improved accordingly. Fast prefetching also improves the video playback quality of TS peers. With fast prefetching, peer buffers grow faster than playback speed, which reduces the urgency for acquiring later packets. When a peer is disconnected due to failure of its parent, it reconnects to the overlay. Since the peer has more video frames to play back until it resumes to receive later video,

it experiences statistically less video disruption due to buffer underflow.

Since fast prefetching is based on the relaxed transfer deadline of time-shifted streams, it does not benefit LS peers. When LS peers are disconnected from the overlay, they often experience buffer underflow because the play-out deadline of a video packet is kept small in live streaming. To compensate for a tight play-out deadline, LS peers connect to peers that can minimize the number of logical hops to the source [35]. Using this criterion, LS peers attempt to reduce disruptions in receiving the video when their parent fails or leaves unexpectedly.

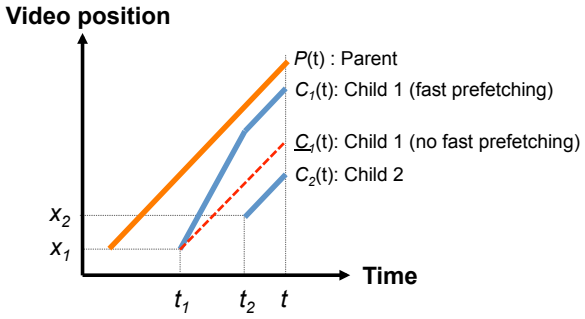

Figure 8. An example of fast prefetching for TS peers. The parent peer can serve two children concurrently. At time t, the excess download at Child 1 due to fast prefetching is $C_1(t) - \underline{C}_1(t)$.

The effects of fast prefetching are illustrated in Fig. 8. In this illustration, the uplink of the parent is assumed to be twice the video bitrate. $P(t)$ is the video trajectory of the parent. After Child 1 connects to its parent, it prefetches the video faster than the playback speed until Child 2 connects to the same parent. The trajectories of the downloaded video for Child 1 and 2 are depicted as $C_1(t)$ and $C_2(t)$, respectively. If fast prefetching is not employed, Child 1 receives video at the playback speed, which is marked by $\underline{C}_1(t)$. Since fast prefetching enables peers to download video faster than at the playback speed, the peer database's estimation on the peers' buffer state may deviate from the actual buffers. To correct mismatches incurred at the database, peers intermittently report their buffer state to the video server.

B. Selecting Parents that Maximize Prefetching

LS peers choose their parents according to the number of hops to the server, which is certainly not optimal for TS peers because the video disruption is less likely due to fast prefetching. To maximize prefetching, we propose a new parent selection algorithm for TS peers, which estimates video download for each parent candidate. When Parent Candidate i is probed by a TS peer that requests the video from position x, the parent candidate reports (1) the download rate q_i from its current parent, (2) the number of bits cached at Parent Candidate i and requested by the TS peer (e.g., a part of the requested substream and beyond the position x), denoted by l_i, and (3) the maximum upload rate it can offer to the TS peer, denoted by r_i. In Fig. 9, q_i, r_i, and l_i reported by Parent Candidate i are depicted over time. The Y-axis shows the video position in bits. Note that the rates and the buffer amount were adjusted to reflect the number of trees. As each multicast tree delivers a $1/k$ fraction of the video bitstream, q_i, r_i, and l_i were multiplied by k for the illustration (k is the number of trees).

After probing the candidates, the TS peer computes the amount of prefetched video that it expects to receive time D after connecting to each parent candidate: if the peer's download catches up with its parent's download before time D has elapsed, the expected download is $D \cdot q_i + l_i$. Otherwise, the expected download is $D \cdot r_i$ (see Fig. 9). The value of D is chosen to be sufficiently small, as q_i and r_i may change over time. The best candidate k is chosen according to the rule

$$k = \arg\max_i \min\left(D \cdot q_i + l_i, D \cdot r_i\right). \tag{7}$$

The video server is selected as a parent only when there is no peer possessing the requested video or when no peer that possesses the requested video has available bandwidth. It should also be noted that the server does not participate in fast prefetching in order to minimize its uplink bandwidth usage.

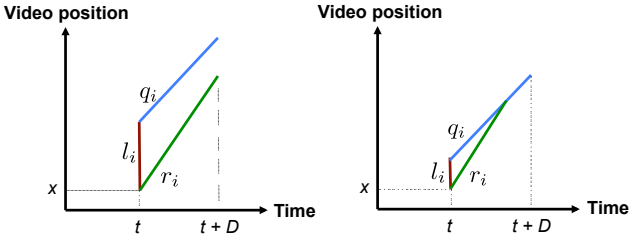

Figure 9. Expected download at a child peer after time D when a child peer joining at time t connects to parent candidate i. Left: child buffer lags behind parent buffer at time $t + D$. Right: child buffer catches up with parent buffer. (q_i: download rate of Parent Candidate i from its parent, l_i: buffered video in bits at Parent Candidate i, r_i: upload rate from Parent Candidate i to the child peer.)

C. Uplink Bandwidth Partitioning

To support fast prefetching, peers partition their available uplink bandwidth to children as follows:

(Step 1) Allocate the tree bitrate $r(= R/k)$ to every LS child. R is the video bitrate and k is the number of trees.

(Step 2) Allocate r to every TS child who completed fast prefetching (the child's latest video time stamp is the same as the parent's latest video time stamp).

(Step 3) Compute $\tilde{r} = U/n$, $\tilde{r} \geq r$. U is the remaining uplink bandwidth after Step 2, and n is the number of TS children who have not been assigned bandwidth in the previous steps. If some TS children have a smaller downlink capacity than $\tilde{r}$, the bandwidth allocated to them is curtailed in order not to overwhelm them. The remaining bandwidth is evenly allocated to the remaining TS children.

The algorithm is illustrated in Fig. 10. Note that the server performs no bandwidth partitioning because it supports no fast prefetching.

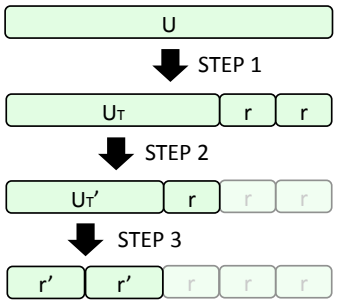

Figure 10. Uplink bandwidth partitioning. In Step 1, LS children are assigned the tree bitrate. In Step 2, TS children that have completed fast prefetching are assigned the tree bitrate. In Step 3, TS children that can perform fast prefetching are assigned the remaining uplink bandwidth evenly. Every peer except the server regularly performs bandwidth partitioning.

VI. Experimental Results

In the experiments in the ns-2 simulator [36], 150 peers join and leave repeatedly with an average lifetime of 120 seconds. We used the *Mother and Daughter* sequence, encoded at $R = 420$ kbps, using H.264/AVC with an intraframe period of 15 frames. Two B frames were encoded between anchor frames. The video server's uplink bandwidth was set to $20R$. $2R$ was allocated to LS peers, and the rest of the server's uplink bandwidth was allocated to TS peers. The peer uplink bandwidth was set to $2R$ and the download bandwidth was set to $4R$. On joining at time t, half of the peers watched the live stream. The rest of them watched the stream shifted by a random delay ranging from 0 to t.

First, we examine the system without fast prefetching. Fig. 11 is the plot of the experimental results for video availability among peers. The value associated with the pair $(t,\ x)$ represents the average number of peers that possess video at position x at time t. The result closely matches the video availability predicted by the model in Section IV and shown in Fig. 5. Fig. 12 shows the video availability for the model and the experiments in terms of the number of TS peers that possess video at position x $(0 \leq x \leq 900)$ at time 900s (no LS peers were present). It is notable that video availability peaks at 450s, the half point of the video available at 900s, under the assumption of random video access. As with Fig. 5 and 11, the video availability predicted by the model closely matches the averaged video availability of the experiments. As the session proceeds, only a small neighborhood of the video near the live stream is cached by the majority of the peers. The extent of the neighborhood is determined by several factors, one of which is the average peer lifetime. The expected number of peers that cache a past portion continues to decrease over time, as expected from the analysis.

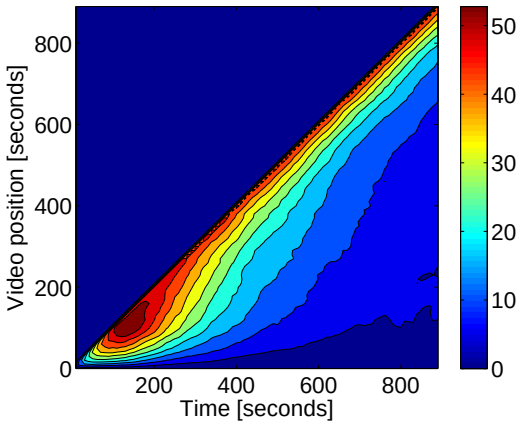

Figure 11. Video availability averaged over 100 simulation runs. (λ = 1.125 joins per second, $1/\mu$ = 120 seconds, and α = 0.5 are used.)

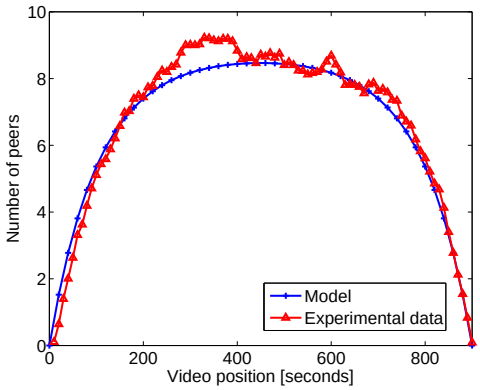

Figure 12. Video availability at 900 seconds. Averaged over 100 simulation runs. Only TS peers (α = 0) joined the system at the rate of λ = 1.125 joins per second. $1/\mu$ = 120 seconds.

Next, we study the system behavior when fast prefetching is employed. In Fig. 13, the average number of direct children of the server is plotted for three different cases:

- No fast prefetching is performed.
- Fast prefetching with parent selection *minimum hops*. For both TS and LS peers, parents that minimize the number of logical hops to the server are selected.
- Fast prefetching with parent selection *maximum throughput*. The parent selection algorithm described in Section V-B is used (D is set to 10 seconds).

The graphs show that with fast prefetching, fewer connections directly to the server are established because peers find their requested video among other peers more often than without fast prefetching. The proposed parent selection algorithm further reduces the server load, compared to the case where parent selection *minimum hops* was used.

As the session progresses, more TS peers connect to the server. This causes peers to experience less peer churn, leading to higher video quality at the cost of server

bandwidth. Since only few TS peers need to connect to the video server at the early stage of a session, we evaluate video quality for the first 900 seconds. When there are missing or late packets that do not arrive in time, the last fully decoded frame is displayed until the next I frame arrives. Table I summarizes average video quality and server load. The average PSNR is computed at the end of the simulation by taking the average of the PSNR value of the video frames displayed at all peers. For missing frames, copy error concealment (display of the previously decoded frame) is used. The table indicates that fast prefetching achieves higher or similar video quality with reduced server load. We observe that fast prefetching with parent selection *maximum throughput* outperforms no fast prefetching by 2 dB for TS peers with about 40% sever load reduction. In Fig. 13, the server load remains low at the early stage of the session because TS peers can easily find peers to connect to. Toward the end of the session, in contrast, more TS peers need to receive video from the video server as fewer peers are found to cache video.

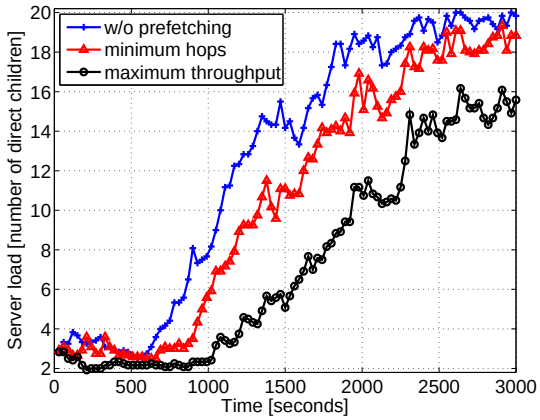

Figure 13. Comparison of server load over time (averaged over 10 simulation runs).

TABLE I.
EFFECTS OF FAST PREFETCHING (COMPUTED OVER THE FIRST 900 SECONDS). VIDEO QUALITY IS REPRESENTED BY THE AVERAGE PSNR OF PEERS. SERVER LOAD IS THE AVERAGE NUMBER OF DIRECT CHILDREN OF THE SERVER. *Hops* AND *throughput* ARE PARENT SELECTION CRITERIA.

	Parent selection criterion	Average PSNR LS peers [dB]	Average PSNR TS peers [dB]	Server load [num. of children]
Without prefetching	hops	40.6	40.4	3.7
With prefetching	hops	41.0	42.2	2.9
	throughput	41.5	42.3	2.2

Finally, we investigate the sensitivity of fast prefetching against D. Recall that D denotes a *look-ahead time*, the time a TS peer looks ahead to compute the expected download. In the experiment, we had 75 TS peers with uplink speed of $2R$, with 60 seconds of average lifetime and 6 seconds of average off-time. Simulations ran for 900 seconds and the average download speed at which a peer received video was averaged over 20 simulations. Fig. 14 shows the average download rate, which is normalized by the video bitrate. We observe that the average download peaks near 0 and decreases as D increases, indicating that large D may overestimate expected downloads because selected parents may depart the system. However, the download speed is not very sensitive to the value of D, showing only a 3% drop from the peak value for $1 \leq D \leq 20$. The download rate becomes nearly flat as D approaches the average peer lifetime. It is worth noting that when $D \approx 0$, the maximum-throughput algorithm in Eq. 7 can be viewed as a maximum-download-rate algorithm; when $D \approx 0$ and l_i is moderate, $\min(D \cdot q_i + l_i, D \cdot r_i)$ approximates to $D \cdot r_i$, thereby $\arg\max_i \min(D \cdot q_i + l_i, D \cdot r_i) \approx \arg\max_i r_i$.

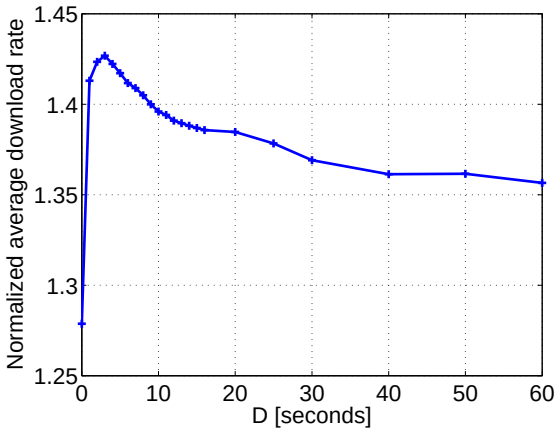

Figure 14. Normalized average download rate with respect to D, look-ahead time.

VII. CONCLUSIONS

In this paper, a tree-based P2P live streaming system like SPPM is shown to support time-shifted streaming without any fundamental changes in the system architecture. To understand the distribution of video segments among peers, we modeled the average number of peers that possess the required video segments cached in their buffers, defined as video availability. The model and the experiments revealed that few peers request time-shifted streams from the server at the early stage of the session because they can connect to peers who have been caching the live stream, and that the server load increases as the session progresses since a smaller number of peers are likely to have the requested time-shifted streams. The video availability improves more significantly when fast prefetching is combined with the maximum throughput parent selection. This shows that the overlay may look very different depending on what objectives are pursued in the overlay construction.

In this work, we considered the continuous bitstream consisting of past video packets stored in peers' local storage. One can devise more sophisticated caching algorithms, which may allow multiple chunks of video that are not necessarily continuous in bitstream. Server bandwidth allocation is also an important research problem. We have found that time-shifted streaming often requires more server bandwidth than live streaming because peers that request past portion of video need to connect to the server if the requested video chunk is not available from other peers. Thus, the server needs to divide its finite uplink bandwidth between live streaming and time-shifted streaming dynamically in order to maximize the overall video quality of 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 the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 1–14.

[2] Y. Shen, Z. Shen, S. Panwar, K. Ross, and Y. Wang, "On the design of prefetching strategies in a peer-driven video on-demand system," *IEEE Intern. Conf. Multimedia and Expo*, pp. 817 – 820, Jul. 2006.

[3] S. Sheu, K. Hua, and W. Tavanapong, "Chaining: a generalized batching technique for video-on-demand systems," in *IEEE Proc. Intern. Conf. Multimedia Computing and Systems*, Ottawa, Canada, Jun. 1997, pp. 110 – 117.

[4] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. ACM Intern. Conf. Multimedia*, University of Bristol, United Kingdom, 1998, pp. 191–200.

[5] F. A. T. S.-H. G. Chan, "Tradeoff between system profit and user delay/loss in providing near video-on-demand service," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 8, pp. 916–927, Aug. 2001.

[6] B. Qudah and N. Sarhan, "Towards scalable delivery of video streams to heterogeneous receivers," in *Proc. ACM Intern. Conf. Multimedia*, Oct. 2006, pp. 347 – 356.

[7] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE Journ. on Selected Areas on Comm.*, vol. 22, no. 1, pp. 91 – 106, 2004.

[8] D. Wang and J. Liu, "Peer-to-peer asynchronous video streaming using Skip List," *Proc. of IEEE International Conference on Multimedia and Expo*, 2006.

[9] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service," in *Proc. Intern. Conf. World Wide Web*. ACM Press New York, NY, USA, May 2003, pp. 301–309.

[10] T. Do, K. Hua, and M. Tantaoui, "Robust video-on-demand streaming in peer-to-peer environments," *Computer Communications*, vol. 31, no. 3, pp. 506–519, Feb. 2008.

[11] H. Chi, Q. Zhang, and S. Shen, "Efficient search and scheduling in P2P-based media-on-demand streaming service," *IEEE Journ. on Selected Areas on Comm.*, vol. 25, no. 1, pp. 119–130, Jan. 2007.

[12] W. Yiu, X. Jin, and S. Chan, "VMesh: distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journ. on Selected Areas on Comm.*, vol. 25, no. 9, pp. 1717–1731, 2007.

[13] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "Directstream: A directory-based peer-to-peer video streaming service," *Computer Communications*, Jan. 2008.

[14] T. Do, K. A. Hua, and M. Tantaoui, "P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment," vol. 3, June 2004, pp. 1467—1472.

[15] C. Huang, J. Li, and K. W. Ross, "Peer-assisted vod: Making internet video distribution cheap," Feb. 2007.

[16] X. Xu, Y. Wang, S. Panwar, and K. W. Ross, "A peer-to-peer video-on-demand system using multiple description coding and server diversity," *Intern. Conf. Image Processing*, vol. 3, pp. 1759 – 1762, Oct. 2004.

[17] Z. Liu, Y. Shen, S. Panwar, K. W. Ross, and Y. Wang, "Efficient substream encoding and transmission for P2P video on demand," in *IEEE Proc. Intern. Packet Video Workshop*, Nov. 2007, pp. 143 – 152.

[18] S. Annapureddy, S. Guha, C. Gkantsidis, and D. Gunawardena, "Exploring VoD in P2P swarming systems," in *IEEE Proc. Intern. Conf. Computer Comm.*, Anchorage, AK, Jan. 2007, pp. 2571 – 2575.

[19] C. Zheng, G. Shen, and S. Li, "Distributed prefetching scheme for random seek support in peer-to-peer streaming applications," in *Proc. ACM Intern. Conf. on Multimedia, P2PMMS Workshop*, Singapore, Nov. 2005, pp. 29 – 38.

[20] K. M. Ho, W. F. Poon, and K. T. Lo, "Video-on-demand systems with cooperative clients in multicast environment," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 19, no. 3, pp. 361 – 373, Mar. 2009.

[21] Weihui, J. Xu, and C. Qing, "Dynamic multicast tree to support time-shifting in real-time streaming," in *International Conference on Intelligent Computing and Cognitive Informatics*, Los Alamitos, CA, USA, 2010, pp. 251–254.

[22] X. Hou and R. Ding, "Research of providing live and time-shifting function for structured P2P streaming system," in *Proceedings of the 2010 International Conference on Machine Vision and Human-machine Interface*, Washington, DC, USA, 2010, pp. 239–242.

[23] S. Deshpande and J. Noh, "P2TSS: Time-shifted and live streaming of video in peer-to-peer systems," in *Proc. of IEEE International Conference on Multimedia and Expo*, 2008, pp. 649–652.

[24] J. Noh, P. Baccichet, F. Hartung, A. Mavlankar, and B. Girod, "Stanford Peer-to-Peer Multicast (SPPM) – overview and recent extensions," *Proc. of International Picture Coding Symposium (PCS), Chicago, Illinois, USA, invited paper,*, May 2009.

[25] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. of ACM NOSSDAV, Miami Beach, FL*, Miami, FL, May 2002, pp. 177 – 186.

[26] Y. Chu, S. Gao, and H. Zhang, "A case for end system multicast," in *Proc. ACM Sigmetrics*, Santa Clara, CA, 2000, pp. 1 – 12.

[27] J. Noh, P. Baccichet, and B. Girod, "Experiences with a large-scale deployment of Stanford Peer-to-Peer Multicast." Seattle, WA: IEEE Proc. Intern. Packet Video Workshop, May 2009, pp. 1 – 9.

[28] E. Setton, J. Noh, and B. Girod, "Congestion-distortion optimized peer-to-peer video streaming," *International Conference on Image Processing (ICIP), Atlanta, Georgia*, pp. 721 – 724, Oct. 2006.

[29] ——, "Low-latency video streaming over peer-to-peer networks," *IEEE Intern. Conf. Multimedia and Expo*, pp. 569 – 572, Jul. 2006.

[30] J. Noh, A. Mavlankar, P. Baccichet, and B. Girod, "Time-shifted streaming in a peer-to-peer video multicast system," in *Proc. of IEEE Global Communications Conference (GLOBECOM 2009)*, Honolulu, Hawaii, USA, Nov./Dec 2009.

[31] B. Chang, L. Dai, Y. Cui, and Y. Xue, "On Feasibility of P2P on-demand streaming via empirical VoD user behavior analysis," in *Proc. Intern. Conf. Distributed Computing Systems Workshops*, Beijing, China, 2008, pp. 7–11.

[32] Y. Chu, A. Ganjam, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early experience with an Internet broadcast system based on overlay multicast," in *USENIX Annual Technical Conference*, June 2004, pp. 347 – 356.
[33] B. Li, S. Xie, G. Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang, "An empirical study of the Coolstreaming+ system," *IEEE Journ. on Selected Areas on Comm.*, vol. 25, no. 9, pp. 1627–1639, Dec. 2007.
[34] S. Ross, *Stochastic processes*, 2nd ed., 1996.
[35] P. Baccichet, J. Noh, E. Setton, and B. Girod, "Content-aware P2P video streaming with low latency," in *IEEE Proc. Intern. Conf. Multimedia and Expo*, Beijing, China, 2007, pp. 400 – 403.
[36] "The Network Simulator - NS-2," http://www.isi.edu/nsnam/ns, seen on June 10, 2010.

Jeonghun Noh received an M.S. degree in Electrical Engineering from Seoul National University, Korea, in 1999, and a Ph.D. in Electrical Engineering from Stanford University, CA, in 2010, respectively.

He is currently a senior systems engineer at ASSIA inc., Redwood City, CA, USA. His research interests include low-latency data distribution, peer-to-peer streaming, mobile video streaming, and broadband access networks. He has authored or co-authored over 17 conference papers and one journal paper, and holds two US patents granted and seven US patents under review. He worked as a summer research intern at Sharp Laboratories of America, Camas, WA, in 2006 and 2007. He worked as a consultant at Dyyno Inc., Palo Alto, CA, in 2008. He collaborated with Deutsche Telekom Laboratories in Los Altos, CA, and in Berlin, Germany, in 2008-2010. He is a recipient of the ACM Mobimedia Best Student Paper Award in 2009 and the IEEE CCNC Best Presentation Award in 2008. Dr. Noh is a member of IEEE, IEEE ComSoc, and IEEE MMTC.

Bernd Girod is Professor of Electrical Engineering and (by courtesy) Computer Science in the Information Systems Laboratory of Stanford University, California, since 1999. Previously, he was a Professor in the Electrical Engineering Department of the University of Erlangen-Nuremberg. His current research interests are in the areas of video compression, networked media systems, and image-based retrieval. He has published over 450 conference and journal papers, as well as 5 books, receiving the EURASIP Signal Processing Best Paper Award in 2002, the IEEE Multimedia Communication Best Paper Award in 2007, the EURASIP Image Communication Best Paper Award in 2008, as well as the the EURASIP Technical Achievement Award in 2004. As an entrepreneur, Professor Girod has been involved with several startup ventures, among them Polycom, Vivo Software, 8x8, and RealNetworks.

He received an Engineering Doctorate from University of Hannover, Germany, and an M.S. Degree from Georgia Institute of Technology. Prof. Girod is a Fellow of the IEEE, a EURASIP Fellow, and a member of the German National Academy of Sciences (Leopoldina).