On Node Stability and Organization in Peer-to-Peer Video Streaming Systems

Feng Wang, *Student Member, IEEE,* Jiangchuan Liu, *Senior Member, IEEE,* and Yongqiang Xiong, *Member, IEEE*

***Abstract*—Recently, peer-to-peer video streaming technology has been widely suggested to support broadcasting videos over the Internet. Given that the application-layer nodes (peers) are autonomous and may join or leave at will. Dealing with node dynamics (known as *churn*) under the bandwidth and timing constraints for video streaming thus becomes a necessary yet challenging task. One conventional argument is that the stable nodes are too small a group to be effectively utilized. While we agree this argument from a whole session's point-of-view, in this paper, we argue that the significantly longer lifespans of the stable nodes allow each of them to appear in much more snapshots of the overlay than transient nodes and, as a result, they constitute a significant portion in every snapshot of the overlay, substantially affecting the performance of the overall system. Such observation has been verified by our trace analysis on PPLive, a large-scale peer-to-peer live streaming system, as well as analytical modeling based on node behaviors. Motivated by this, we further propose a tiered overlay design, with stable nodes being organized into a tier-1 backbone for serving tier-2 nodes. It offers a highly cost-effective and deployable alternative to proxy-assisted designs. We develop a comprehensive set of algorithms for stable node identification and organization. Specifically, we present a novel structure, *labeled tree* (or LBTree in short), for the tier-1 overlay, which, leveraging stable peers, simultaneously achieves low overhead and high transmission reliability. Our tiered framework flexibly accommodates diverse existing overlay structures in the second tier. Through extensive simulations, we demonstrate that the customized optimization using selected stable nodes can greatly boost the streaming quality and also effectively reduce the control overhead.**

***Index Terms*—Labeled tree, overlay network, peer-to-peer video streaming, stable node.**

I. Introduction

RECENTLY, peer-to-peer video streaming technology has been widely suggested to support broadcasting videos over the Internet. There are two key factors continuously making the approach attractive. First, such an application-layer technology does not require support from Internet routers and network infrastructure, and, consequently, is cost-effective and easy to deploy. Second, in a peer-to-peer overlay, a participant that tunes into a multicast/broadcast channel is not only downloading a video stream but also uploading it to other participants watching the video. It thus has the potential to scale with group size, as greater demand also generates more resources. Peer-to-peer technologies have been applied to a wide range of applications, in particular, file downloading. However, video streaming applications pose very different demands, i.e., stringent real-time performance requirements in terms of bandwidth and latency. The problem is further complicated given that the application-layer nodes are autonomous and may join or leave at will or crash without notification. Dealing with the node dynamics (known as *churn*) under bandwidth and timing constraints thus becomes a necessary yet challenging task.

Previous works on peer-to-peer video streaming can be largely categorized into approaches using tree (or multitree) structure and those using mesh structure. The tree structure, stemmed from internet protocol (IP) multicast, is probably the most natural and efficient structure in terms of bandwidth and delay optimization [4], [8], [26]. However, the approaches using a tree structure are often known to be severely suffered from node churn, where the leave or failure of a node, especially one close to the source, may cause data outage in all its descendants. The overlay thus has to repair the tree structure frequently, which in turn brings extra costs and renders the structure to be suboptimal during the repair processes.

One possible enhancement to such a single tree structure is to employ multiple disjoint trees [7], [22], [25], with each of the trees delivering a substream of the video. While the multitree can remarkably improve the resilience, it is clearly more complex to be constructed and maintained. Optimizing the multiple trees as a whole without violating the disjoint property can also be quite difficult in the presence of autonomous end-hosts [6], [20].

Recently, data-driven overlays have been proposed as an alternative resilient solution [1], [13], [16], [23], [32]. A mesh is constructed out of the end-hosts in such overlays, where each node independently selects some other nodes as neighbors and exchanges data with them. One significant difference between the tree and the mesh overlays lies in their data delivery strategies. In a tree or multitree, a video stream is basically pushed along the well-defined routes, i.e.,

Manuscript received December 14, 2010; revised March 9, 2011; accepted May 12, 2011. Date of publication September 15, 2011; date of current version November 23, 2011. The work of F. Wang and J. Liu was supported in part by a Canada NSERC Strategic Grant, in part by a NSERC Discovery Grant, in part by a NSERC DAS Grant, and in part by a MITACS Project Grant. A preliminary version of this paper appeared in IEEE INFOCOM'08.

F. Wang and J. Liu are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: fwa1@cs.sfu.ca; jcliu@cs.sfu.ca).

Y. Xiong is with the Wireless and Networking Group, Microsoft Research Asia, Beijing 100190, China (e-mail: yqx@microsoft.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2011.2165189

from parents to their children. In a mesh, given that multiple dynamic neighbors may have video data available to send, a node has to pull data to avoid significant redundancies. A mesh-based system is more robust, but experiences longer delays and higher control overhead. More explicitly, there is an efficiency-delay tradeoff [18], [27], [31]; if the mesh nodes choose to send notifications for every data block arrival, then the overhead will be excessive. Periodical notifications containing buffer maps within a sliding window, as suggested in [23] and [32], reduce the overhead, but increase the delays.

The tradeoffs in these proposals largely come from a general belief that every node in the overlay is subject to churn. While strategically deployed proxies could alleviate this problem, their applicability is limited due to the excessive deployment cost. In this paper, we take a different cost-effective approach to explore the potentials of existing stable nodes. The conventional argument is that the stable nodes are too small a group to be effectively utilized. While we agree this argument from a whole session's point-of-view, we find that the significantly longer lifespans of the stable nodes allow each of them to appear in much more snapshots of the overlay than transient nodes and, as a result, they constitute a significant portion (on average >70%) in every snapshot of the overlay. Such observation has been verified by our trace analysis on PPLive, a large-scale peer-to-peer live streaming system, as well as analytical modeling based on node behaviors. Our further investigation shows that, in the practical system, up to 80% of the data were delivered through 20% of the connections. A closer look reveals that the nodes associated to these connections are generally stable. In short, the stable nodes reach a critical mass and play an important role in peer-to-peer overlay streaming.

Inspired by this observation, we proposed a tiered overlay design that conceptually separates stable nodes and transient nodes into two levels. The tier-1 overlay, consisting mainly of stable nodes, works as an amplifier of the source to the transient nodes in tier-2. In other words, each or several tier-1 nodes can act as a "proxy" for a cluster of tier-2 nodes, and the latter can be organized through diverse existing overlay structures with minimal modification. Though the tier-1 nodes are not persistent like a dedicated proxy, their higher quantity compensates this, and more importantly, they do not suffer from the excessive deployment cost.

Within this framework, we develop a comprehensive set of algorithms for stable node identification and organization. Specifically, we present an effective predictor to identify potential stable nodes, as well as a randomized and distributed algorithm for promoting the nodes into the tier-1 overlay. We further develop a novel structure, *labeled tree* (or LBTree in short), for the tier-1 overlay, which, leveraging the stable nodes, simultaneously achieves low overhead and high transmission reliability.

Our tiered framework is evaluated through extensive simulations, which demonstrate that the customized optimization for stable nodes noticeably boosts the streaming quality and effectively reduces the control overhead. The tiered system also better accommodates flash crowd, i.e., many nodes join the overlay in a short time [17].

The remainder of this paper is organized as follows. Section II introduces the related work. Section III validates the existence and importance of stable nodes through both trace analysis and mathematical modeling. In Section IV, we devise effective online algorithms for identifying stable nodes. The LBTree that organizes stable nodes into a tier-1 overlay is presented in Section V. We then evaluate the performance of our proposal in Section VI. Finally, Section VII concludes this paper and discusses potential future directions.

II. Background and Related Work

Numerous peer-to-peer multicast protocols have been developed for live video streaming, which can be broadly classified into two categories according to their overlay structures [18], namely, *tree-based* and *mesh-based*. The former, like IP multicast, uses a tree rooted at the source as the data delivering structure [4], [8], [26]. However, unlike IP multicast with dedicated routers, the nodes in an application-layer overlay are autonomous end-hosts, which may join or leave at will or crash without notification. Later studies rely on multiple disjoint trees to mitigate the impact of such overlay churns [7], [22], [25]. Another popular alternative is data-driven mesh overlay, inspired by file swarming systems like BitTorrent. Each node maintains a number of partners, and the partner relationships among all nodes consist of a mesh structure. The partners periodically exchange data availability information, and accordingly issue requests to fetch expected data segments [1], [13], [16], [23], [32].

Both tree/multitree and mesh solutions have seen their success in practical deployment, and yet neither completely overcomes the challenges from the dynamic peer-to-peer environment. The selling point for the data-driven mesh overlays is their robustness, but the lack of a well-ordered parent/children relation implies that data have to be pulled from neighbors, which suffers an efficiency-latency tradeoff. The push delivery in a tree is efficient, but has to face data outage in descendants when an internal node fails. The predefined flow direction also prevents the overlay from fully utilizing the bandwidth between node pairs, e.g., that between two leaf nodes.

There have been efforts to reconcile tree and mesh to form a hybrid overlay, e.g., Chunky-Spread [27], PRIME [20], and mTreebone [29], or use pull and push simultaneously to improve the efficiency of data delivery, e.g., GridMedia [31], CoolStreaming+ [14], and the work proposed in [15]. We, however, consider a tiered design that conceptually separates stable nodes and others, and we do not restrict the system to specific overlay structures, particularly for tier-2. Some recent studies have also differentiated nodes and implicitly considered node stability, e.g., using supernode (in Skype voice streaming) [21] or superior peers [19], or dedicated proxies to assist other peers [5], or giving preference to nodes with longer durations for overlay construction [6]. Our work differs from them in that we present a systematic study on the existence and importance of stable nodes using both real traces and analytical models. Leveraging the promising results from this paper, we present customized algorithms to effectively identify stable nodes, and then organize them into a separated overlay to maximize their contributions.

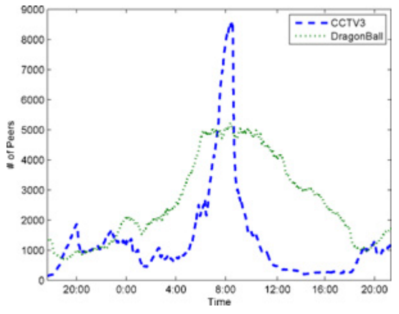

Fig. 1. Community size over time.

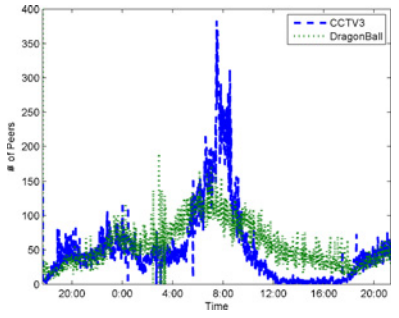

Fig. 2. Node arrival rate over time.

III. Understand Stable Nodes in Peer-to-Peer Video Streaming Systems

We first try to answer the following fundamental questions.

1) Do stable nodes exist in real peer-to-peer streaming systems?
2) And if so, will they reach a critical mass that substantially impacts the overlay performance?

A. Trace Analysis and Modeling

We start from a trace-driven study on a representative practical system, PPLive [1]. It is one of the largest commercial peer-to-peer live streaming systems to date, attracting over 100 000 online users during peak times, and is also one of the mostly examined systems in academia [11], [12]. Our investigation is based on traces of two popular PPLive channels, namely, CCTV3 and DragonBall. These traces are gathered by an online crawler that continuously collects information from each channel. Figs. 1 and 2 show the community size, i.e., the node population and the node arrival rate of each channel. It is clear that there are *time-of-day effects* in a 24-h period [24]. To mitigate the impact of such effects, we focus our study on two representative periods for each channel: 6:00–10:00 (CCTV3 Trace1/DragonBall Trace1) and 18:00–22:00 (CCTV3 Trace2/DragonBall Trace2), respectively. We have found that our conclusions generally apply to other periods, and are also consistent with our recent traces collected from PPLive video accelerator [2], the latest PPLive-based streaming video accelerator.

In Fig. 3, we plot the complementary cumulative distribution functions (CCDFs) of nodes' lifetimes in the channels. Intuitively, the longer a node resides in a channel, the more stable it is. Let us assume that a node is stable if its lifetime

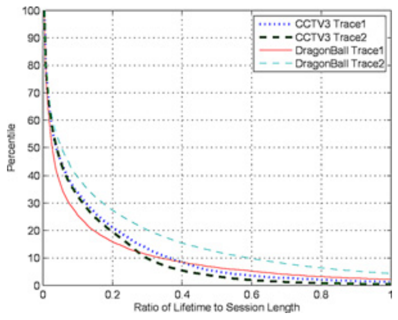

Fig. 3. CCDF of lifetime distribution in different traces.

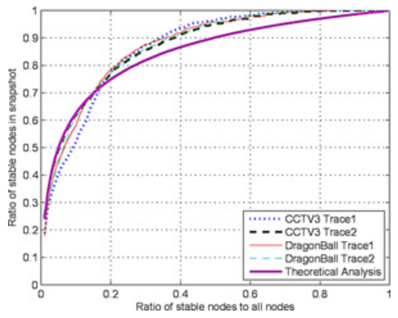

Fig. 4. Ratios of stable nodes (different minimum lifetimes are used to define stable nodes).

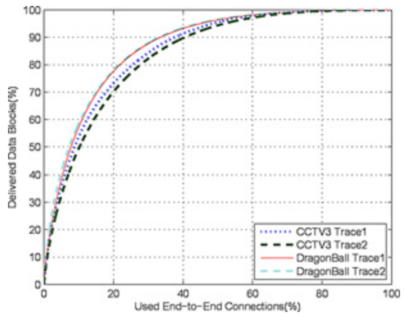

Fig. 5. CDF of data blocks delivered by different node-pair connections.

exceeds 40% of the observed period (as will be seen later, this is a pretty conservative definition), we can see that there does exist a set of stable nodes, though insignificant (from 5.5% to 15.4% in different traces). While this observation is consistent with the conventional argument that transient nodes dominate the overlay, we note that the significantly longer life spans of the stable nodes allow each of them to appear in much more snapshots of the overlay and, as a result, they constitute a significant portion (on average >70%) in every snapshot of the overlay (see Table I).

To better understand how the small portion of the stable peers become significant in a per-snapshot view, we next show a simple analysis. Assume T is the current time, $X(t)$ is the number of nodes arrived at time t, and $F(x)$ is the cumulative distribution function (CDF) of node lifetime. The total number of nodes at T is then

$$\sum_{i=0}^{T-1} X(i) \cdot (1 - F(T - i)).$$

Let l be the minimum lifetime for a node to be considered stable, the total number of stable nodes at time T is

$$\begin{aligned}
&\sum_{i=0}^{T-1}\{X(i)\cdot(1-F(T-i))\cdot I_{[T-i\geq l]} \\
&\quad +X(i)\cdot(1-F(l))\cdot I_{[T-i<l]}\} \\
&= \sum_{i=0}^{T-l} X(i)\cdot(1-F(T-i)) \\
&\quad + \sum_{i=T-l+1}^{T-1} X(i)\cdot(1-F(l))
\end{aligned}$$

where I is the indicator function.

It is known that, in a peer-to-peer system, the node lifetime can be approximated by a Pareto distribution normalized by the session length [6], [24]. This is also verified by our PPLive traces and we find that given a session length $L = 4\,\text{h}$, the typical parameters are $k = 1$ and $x_m = 1\,\text{min}$. Assume the node arrival rate is μ, we can calculate the expected ratio of stable nodes in the overlay snapshot at time T as

$$\begin{aligned}
&\left\{\sum_{i=0}^{T-l} E(X(i))\cdot(1-F(T-i))\right. \\
&\quad \left. + \sum_{i=T-l+1}^{T-1} E(X(i))\cdot(1-F(l))\right\} \\
&\quad \Big/ \left\{\sum_{i=0}^{T-1} E(X(i))\cdot(1-F(T-i))\right\} \\
&= \frac{\sum_{i=0}^{T-l}\mu\frac{\frac{x_m^k}{(T-i)^k}-\frac{x_m^k}{L^k}}{1-\frac{x_m^k}{L^k}} + \sum_{i=T-l+1}^{T-1}\mu\frac{\frac{x_m^k}{l^k}-\frac{x_m^k}{L^k}}{1-\frac{x_m^k}{L^k}}}{\sum_{i=0}^{T-x_m}\mu\frac{\frac{x_m^k}{(T-i)^k}-\frac{x_m^k}{L^k}}{1-\frac{x_m^k}{L^k}} + \mu(x_m-1)} \\
&= \frac{\sum_{i=0}^{T-l}\left[\frac{1}{(T-i)^k}-\frac{1}{L^k}\right] + \sum_{i=T-l+1}^{T-1}\left[\frac{1}{l^k}-\frac{1}{L^k}\right]}{\sum_{i=0}^{T-x_m}\left[\frac{1}{(T-i)^k}-\frac{1}{L^k}\right] + \frac{x_m-1}{x_m^k}(1-\frac{x_m^k}{L^k})} \\
&= \frac{\sum_{i=l}^{T}\frac{1}{i^k}+\frac{l-1}{l^k}-\frac{T}{L^k}}{\sum_{i=x_m}^{T}\frac{1}{i^k}+\frac{x_m-1}{x_m^k}-\frac{T}{L^k}}.
\end{aligned}$$

Using different minimum lifetimes to define stable nodes, Fig. 4 shows the ratio of stable nodes from both session views and snapshot views. It is clear to see that, for a session time of 4 h, when the ratio of stable nodes in the whole session is about 20%, the expected ratio of stable nodes per snap shot is 74.9%. This is very close to the results of our trace studies and thus reaffirms our observations.

B. Contribution of Stable Nodes

We further examine the role of these stable nodes in overlay data delivery. To this end, we emulate the PPLive using the collected traces. Fig. 5 shows the CDF of the data blocks delivered over all active connections throughout the observed period, where a connection (between two nodes) is active if at least one block passes through it. We can see that, over 50% of the data are delivered by 10% of the connections only, and this ratio increases up to 80% for 20% of the connections. A closer look reveals that the nodes associated to these connections are generally stable (using 40% session time as the threshold in this example).

Since mesh protocols including PPLive have built-in mechanisms to avoid loops in data delivering, the delivery paths of each single block simply form a tree, which we call *per-block-tree*. For each PPLive trace, we then extract and compare all the per-block-trees. There are two important findings in this comparison: 1) the internal nodes of the per-block-trees are generally stable, and 2) we can find a small set ($\ll 1\%$) of representative trees. The representativity, defined as the ratio of common internal links between a per-block-tree tree and its best-matching representative, is on average close to 80% (see Table I). In other words, there is an implicit backbone mainly consisting of stable nodes in the PPLive mesh, which delivers a majority of the data blocks.

The evolution of the overlay, together with the multineighbor scheduling, also explains the existence of multiple representative trees in PPLive. Unfortunately, in PPLive, the formation of such trees are implicit, which is not well organized to explore the potential of the stable nodes. Moreover, the naive pull operation is still used, which significantly reduces the efficiency and responsiveness of the mesh overlay.

IV. Online Prediction of Stable Nodes

The existence and the importance of the stable nodes inspires us to envision a two-tier overlay design. The first tier mainly consists of stable nodes, while others are in the second tier. Given the low churn, the tier-1 overlay can be explicitly optimized with minimized maintenance and transmission overhead. This semistable backbone will then serves as an amplifier of the source to tier-2. As such, our framework flexibly accommodates diverse existing overlay organizations with minimal modification for the tier-2 overlay.

Before discussing the details about the organization of the stable nodes in tier-1, we have to first address the critical issue on how the potential stable nodes can be identified online.

Our previous trace analysis and modeling both assume that the node lifetimes are known, which however cannot be determined in advance for autonomous nodes. To address this problem, we introduce a stability index (s-Index) to characterize a node's degree of stability: an s-Index close to 0 means the node is highly transient and close to 1 means it is likely stable. We evaluate s-Index of a node as follows:

$$SI = \begin{cases} \frac{2s}{L-t}, & \text{if } s \leq \frac{L-t}{2} \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

where SI is the value of the s-Index, s is the node duration in the session so far, t is its arrival time, and L is the session length. This index follows the well-known observation that the longer a node stays in the overlay, the longer it would

TABLE I
STATISTICS OF STABLE NODES AND REPRESENTATIVE PER-BLOCK-TREES (SHORTENED AS RP TREES)

Data Set	Average Ratio of Stable Nodes Per Snapshot (%)	Ratio of Stable Nodes Per Trace (%)	Ratio of RP Trees to All Per-Block-Trees (%)	Average Representativity (%)
CCTV3 Trace1	77.7	8.4	0.17	82.3
CCTV3 Trace2	73.8	5.5	0.13	80.9
DragonBall Trace1	85.0	8.5	0.22	79.1
DragonBall Trace2	84.6	15.4	0.28	80.5

stay in the future [6]. Specifically, if a node has already stayed half of the residual session time, it is rational to expect that the node will stay in the next half, and its s-Index thus becomes 1. We have validated the effectiveness of this simple predictor in our simulations. Our design is also flexible to incorporate other known information into the s-Index, e.g., the s-Index of the server or a dedicated proxy can be directly set to 1. As such, our 2-tier framework inherently covers a broad spectrum of systems ranging from pure peer-to-peer to hybrid with proxy assistance.

Given the s-Index of a node, a straightforward way to predict stable nodes is to set a threshold H; for a tier-2 node, if its s-Index is greater than H, it will be predicted stable and promoted to tier-1. Using a firm threshold, however, suffers from two drawbacks: 1) at the beginning of the session, no node but the source is considered stable, and it takes a long time to build up the tiered overlay for efficient data delivering, and 2) a flash crowd will have a double impact, i.e., when a large number of fresh nodes join simultaneously and when these nodes are identified stable simultaneously.

We solve these problems by a randomized identification algorithm. The algorithm strikes to achieve a probability SI/H for a node to be predicted stable (when $SI \leq H$). To realize the linearly increasing probability SI/H, each tier-2 node independently checks its status per unit time; for its sth check (i.e., at time $t+s$), it will be promoted with probability $2/((L-t)\cdot(H-SI)+2)$ (and 0 for $s=0$).

Theorem 1: In the above promotion algorithm, the probability that an active node of duration s so far is predicted stable and promoted to tier-1 is SI/H.

Proof: With discrete time, the probability is given by

$$1-\prod_{k=1}^{s}\left(1-\frac{2}{(L-t)\cdot(H-SI)+2}\right).$$

Since $SI < H \leq 1$, we have

$$\begin{aligned}
&1-\prod_{k=1}^{s}\left(1-\frac{2}{(L-t)\cdot(H-SI)+2}\right)\\
=\;&1-\prod_{k=1}^{s}\left(1-\frac{2}{(L-t)\cdot(H-2k/(L-t))+2}\right)\\
=\;&1-\prod_{k=1}^{s}\frac{(L-t)\cdot H-2k}{(L-t)\cdot H-2k+2}\\
=\;&1-\frac{(L-t)\cdot H-2}{(L-t)\cdot H}\cdot\frac{(L-t)\cdot H-4}{(L-t)\cdot H-2}\\
&\cdots\frac{(L-t)\cdot H-2s}{(L-t)\cdot H-2s+2}\\
=\;&1-\frac{(L-t)\cdot H-2s}{(L-t)\cdot H}=\frac{2s}{(L-t)\cdot H}=\frac{SI}{H}.
\end{aligned}$$

■

In our simulations, we will evaluate the impact of different thresholds H and provide general selection guidelines. Note that our algorithm is fully distributed with no extra message exchanged among the overlay nodes. In addition, as suggested by observations from [10], the built-in randomness will also help to improve the stability of the identified nodes.

V. ORGANIZATION OF STABLE NODES: LBTREE WITH SIDE LINKS

After predicting stable nodes, the immediate question is how they will be organized into the tier-1 overlay to feed themselves as well as to serve the tier-2 overlay. The better stability of tier-1 nodes potentially enables smarter optimization with much lower control costs. Specifically, we suggest a *tree* organization with data *push* for the backbone. This is well-recognized as the most efficient structure for multicast, though in the past its application has been hindered by the high churn rate in a flat overlay.

However, since a tier-1 node acts as a logical proxy for multiple tier-2 nodes, it has more stringent demand on the streaming quality. It must promptly recover data losses, and *loss multiplications*, i.e., avalanche-like losses where one loss in an ancestor leads to losses in all its descendants, have to be minimized. The latter is known as a critical problem of tree structures. Moreover, tier-1 nodes are not absolutely persistent like routers or proxies, and the stability prediction can be inaccurate; thus the churn of nodes still has to be dealt with.

To this end, we introduce LBTree, a novel structure for the tier-1 overlay, which, leveraging the stable nodes, efficiently recovers missing data and overcomes the inherent problems in previous tree designs.

A. Design of LBTree

In an LBTree, a label is used to indicate the position of each tier-1 node. The label is defined as $(R,(a,b))$, where R denotes which level the node resides, and (a,b) is a range recursively calculated as follows. Given a node's label and its max number of children N, its range will be equally divided into N nonoverlapped subranges and each is assigned to a child. For example, for a node of label $(R,(a,b))$ with two children, the children's labels will be $(R+1,(a,(a+b)/2))$ and $(R+1,((a+b)/2,b))$, respectively.

Since the size of a label is quite small (it is straightforward to show that the size is of $O(\log N)$ bits for an LBTree of

TABLE II
EXAMPLES OF RELATIONSHIPS BETWEEN TWO LBTREE NODES x AND y WITH LABELS $(R_x, (a_x, b_x))$ AND $(R_y, (a_y, b_y))$, RESPECTIVELY

Label	$(a_y, b_y) \subset (a_x, b_x)$	$(a_y, b_y) \cap (a_x, b_x) = \emptyset$
$R_x < R_y$	x is y's ancestor	x is closer to source
$R_x = R_y$	N/A	Same distance to source
$R_x > R_y$	N/A	y is closer to source

N nodes), it can be piggybacked with other control information exchanged between nodes, e.g., through the light-weight gossip protocols used in existing systems [9], [28]. Using the labels, it is easy to identify the relationship between two nodes in the tree. Table II gives three basic relationships that facilitate tree construction. We will illustrate more complex relationships for tree maintenance and loss recovery later. Note that all these comparisons can be done in a distributed manner.

In the very beginning, all nodes but the original source are in tier-2 overlay. The LBTree (i.e., the tier-1 overlay), initially only including the source, expands with new stable nodes being promoted. By comparing the labels of existing nodes with sufficient available bandwidth, each newly promoted node can easily locate one closest to the source as its parent. If a tier-1 node leaves, which is unusual, particularly for internal nodes of high s-Indices, similar operations will be invoked for its direct children.

The above operations enable that the expected s-Index is generally higher for nodes closer to the source, and, since the s-Index of an active node is increasing over time, this desirable relation persists over time. In addition, even if a node can only maintain a partial list of the LBTree nodes and initially find a parent that is only locally closest to the source, it may locate other closer ones with information continuously exchanged. The LBTree thus will gradually evolve to a more balanced tree.

B. Side Links for Loss Recovery

In the tier-1 overlay, not only the nodes but also their connections are more stable than others. Hence, it suffers less data loss as well. Our trace analysis and simulation results show that the loss rate in the LBTree can be an order of magnitude lower than that in a traditional mesh overlay, e.g., PPLive. Nevertheless, given the significance of tier-1 nodes and the known loss multiplication problem in the tree structure, once a data loss happens, more proactive and effective recovery operations are expected.

Our solution is to establish *side links* between eligible node pairs in the LBTree for loss recovery. The side links make effective use of the *bandwidth remainders*, i.e., the residual bandwidth that cannot support a full-rate streaming, for transmitting recovery request and response. Bandwidth remainders widely exist in a tree overlay because each child is expecting a full-rate streaming from its parent. For example, for the upload bandwidth of 1 Mb/s and a streaming with the rate 384 kb/s, there remains 256 kb/s bandwidth that cannot be used to fully support a child. Such bandwidth has been largely ignored and thus wasted in existing tree overlays.

Each node can maintain multiple side links from it, and pick up one side link to issue a request for fetching each lost block.

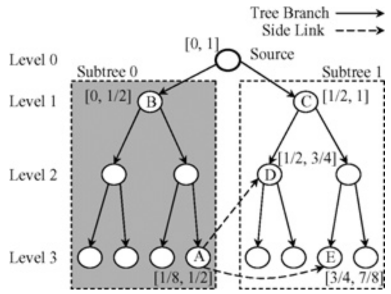

Fig. 6. LBTree with side links.

To effectively recover the lost data, the side links are placed according the following criteria (referred to Fig. 6).

1) A side link from a node should be connected to another node that is closer to the source or at least the same. For example, in Fig. 6, there can be a side link from node A to node E or from E to A, but the side link between nodes A and D must start from A. This guarantees that node A can recovery missing data blocks on time. Note that either A or D can initiate the link construction, because the direction can be easily identified from a label comparison.
2) The two ends of a side link should not share any ancestor except for the source, which we refer to as *orthogonality*. As an example, in Fig. 6, node A should not have a side link with any node in the subtree rooted at node B. Clearly, this constraint will minimize the impact of loss multiplication. Assume that the source's label is $(R_s, (a_s, b_s))$ and its max children number is N_s, for two nodes x and y, the orthogonal relation can be identified using a local label comparison as follows: x and y are orthogonal, if and only if $\forall i \in \{0 \ldots (N_s - 1)\}$, $((a_x, b_x) \cup (a_y, b_y)) \not\subseteq (a_s + \frac{i(b_s - a_s)}{N_s}, a_s + \frac{(i+1)(b_s - a_s)}{N_s})$.

C. Further Discussion

At a first glance, the request-response by side links may look similar to the data pull method used in a mesh overlay. There are indeed two fundamental differences.

1) In our side link approach, the data availability information is no longer needed. It is known that the broadcast of data availability between neighbors introduces remarkable overhead and poses the efficiency-latency tradeoff in mesh-based overlays.
2) In a mesh overlay, a node can pull a data block only if the block is available at one of the neighbors and its availability information has been received.

The end-to-end delay is thus significantly increased, particularly in the presence of loss [27]. In side link request, a node that receives a recovery request must either have the requested data, which is very likely because it is in a different subtree, or have already sent out a recovery request because it is no farther away from the source. In other words, parallel recovery requests are forwarded along the side link path, which makes the recovery process much more efficient and timely.

To further understand the effectiveness of side links and to quantify the expected number of links for each node, we now offer a simple analysis. Consider a node with certain

TABLE III
AVERAGE VALUES OF DATA LOSS RATE, STARTUP LATENCY, AND PLAYBACK DELAY (STATIC SCENARIO)

Approach	Data Loss Rate	Startup Latency	Playback Delay
PPLive	0.0225	54.8087	142.8602
ChunkySpread	0.0004	34.4858	81.9568
PPLive + LBTree	0.0002	20.5929	51.9568
ChunkySpread + LBTree	0.0004	34.6559	78.1457

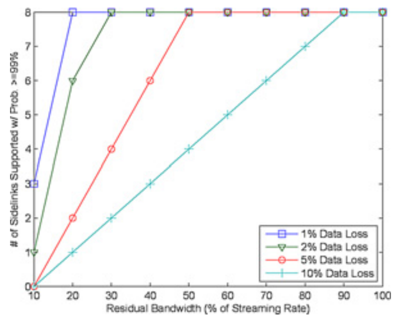

Fig. 7. Number of side links supportable by different residual bandwidth with the loss recovery probability ≥ 99%.

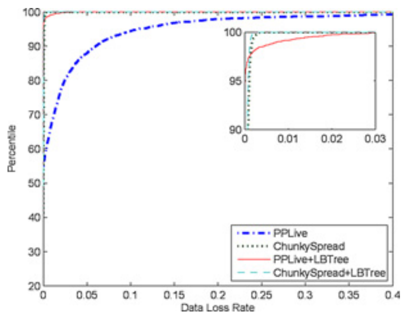

Fig. 8. CDF of data loss rate (static scenario).

bandwidth remainder as a "server" in a queuing system, and the recovery requests as "clients." Let P be the segment loss rate at a node; the "client" arrival thus follows $Pr(X = k) = \sum_{i=1}^{N} \binom{k}{N} (P^k)(1 - P)^{N-k}$. Given the bandwidth remainder and streaming rate, as well as the expected probability for a node to recover missing data before the deadlines, we can then calculate the expected number of side links N through a $G/D/1$ queuing model.

Fig. 7 shows the numerical result for a recovery probability of 99%, where each recovered segment is to be received before its playback deadline (we assume the playback deadline is 30 s after the recovery request is sent). It is clear to see that even the residual bandwidth of a node is only 20% of the streaming rate, it still can support more than eight side links serving nodes with 1% data loss, or one side link serving a node with up to 10% data loss.

VI. PERFORMANCE EVALUATION

We have conducted extensive simulations to validate our analysis and evaluate the proposed algorithms for identifying and organizing stable nodes. For the sake of comparison, we have also implemented two known systems, namely, ChunkySpread [27] and a PPLive-like system. ChunkySpread is a multitree system, which also adopts an auxiliary neighboring graph to facilitate tree construction. PPLive is a typical mesh-based system, which is known as the largest commercial peer-to-peer streaming site to date. Since its code is not open, our implementation is based on the trace data as well as its protocol analysis from [11] and various other sources.

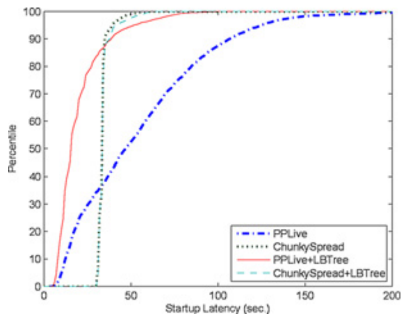

Fig. 9. CDF of startup latency (static scenario).

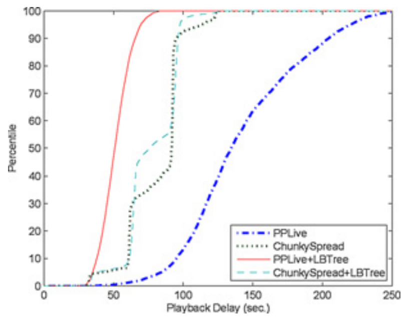

Fig. 10. CDF of playback delay (static scenario).

In our evaluation and comparison, we use the following three typical metrics, which together reflect the quality of service (QoS) experienced by end users.

1) Data loss rate, which is defined as the fraction of the data blocks missing their playback deadlines, i.e., either lost during transmission or experienced excessive delays.
2) Startup latency, which is the time taken by a peer between its requesting to join the session and receiving enough data blocks (30 continuous blocks in our evaluation setting) to start playback.
3) Playback delay, which is the time for a data block from being sent out by the source to being played at a peer.

We also examine the *control overhead*, i.e., the messages used to construct and maintain the overlay as well as the messages to exchange data availability and to request blocks (in mesh only). Such messages are generally of small sizes, but their huge number could still impose heavy load to networks. Hence, we use the average number of control messages per node per second as the metric to measure their impacts.

Unless otherwise specified, the following default parameters are used in our simulation, most of which follow the typical values reported in [3], [11], and [30]. The session length L is set to 3600 s and each data block is of 1-s video; there are 5000 overlay peers. The maximum end-to-end delay is 1000 ms between two peers, and the packet loss rate is set to 2%; the maximum upload bandwidth is uniformly distributed from 1 to 12 times of the bandwidth required for a full streaming. For comparison, we set the number of data delivering neighbors

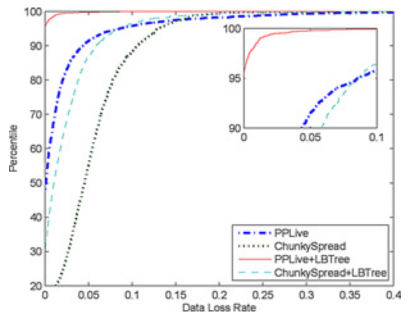

Fig. 11. CDF of data loss rate (dynamic scenario).

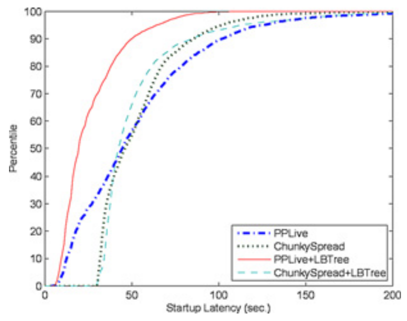

Fig. 12. CDF of startup latency (dynamic scenario).

in PPLive between 4 and 6. The number of substreams in ChunkySpread is set to 16, and receiving any 13 out of the 16 substreams is enough to recover the original streaming. The details about these parameters and their setting guidelines can be found in [11] and [27], respectively.

A. Static Scenario

In the static scenario, we let 200 nodes join the session at the beginning and the rest arrive following a Poisson process (with the average arrival rate being set to 6). The nodes then remain in the session until the session finishes. Since all nodes are stable in this scenario, we set the threshold for s-Index to the default value of 0.2 and defer the discussion of its impacts to later section. The results are shown in Tables III and IV and Figs. 8–10.

Fig. 8 shows the CDF of the data loss rates of the pure PPLive/ChunkySpread and our tiered system (PPLive + LBTree/ChunkySpread + LBTree). It is clear to see that, with the support of LBTree, the data loss rates of both PPLive and ChunkySpread become very low. The pure ChunkySpread also achieves low data loss rate. This is because in the static scenario, no nodes leave during the session time, which eliminates the data interruptions caused by internal nodes leaving. Besides, the redundancy introduced in ChunkySpread also helps to alleviate the data loss during packet delivery. On the other hand, the pure PPLive has a relatively higher data loss rate due to suffering from the in-network packet loss (note that the in-network packet loss rate is set to 2%). As both the exchanged data block information and pull request packets may also be lost in the network, more delays may be introduced to deliver a data packet from the source to a node, making more packets miss their playback deadlines and be counted as data loss.

Fig. 9 compares the startup latencies of the different approaches. The startup latency of PPLive has a great variance and is also the largest one on average (see Table III). This is because in the static scenario, the peers that join early will quickly establish stable neighbor relations with each other and make their neighbor slots nearly saturated (note that a peer can only have a limited number of neighbors). As such, the peers that join later will have to take more time to find an old peer with available neighbor slots to pull streaming data, which will increase the startup latency. On the other hand, by incorporating LBTree, PPLive enjoys noticeable improvement as LBTree can effectively amplify the source capacity. For ChunkySpread, since it is a tree-based system with no latency due to data pull from neighbors, its startup latencies with and without LBTree have no significant difference.

TABLE IV
CONTROL OVERHEAD (STATIC SCENARIO)

Approach	Message Per Node Per Second
PPLive	5.28
ChunkySpread	3.24
PPLive + LBTree	0.94
ChunkySpread + LBTree	1.00

Fig. 10 gives the results on the playback delay. Again, LBTree significantly reduces the playback delay of PPLive by using the tree structure as a source amplifier. With LBTree, the playback delay of ChunkySpread is also improved. This is because in ChunkySpread, substream data from the source are delivered along different paths in different subtrees, and the playback delay is determined by the longest path among these paths. With LBTree as a source amplifier, the variance of these paths can be effectively reduced and as a result, the length of the longest path among them is also shortened.

We also investigate the control overhead with and without incorporating the LBTree. Table IV gives the results. We can see that the original PPLive has the highest overhead, simply due to the exchange of the data availability notifications and the requests for pulling data. When combined with LBTree, since the tier-1 overlay has eliminated pulling and the tier-2 overlay has a reduced scale, the total control overhead is significantly reduced. For ChunkySpread, the control overhead has also been reduced after incorporating the LBTree. We believe that the gain comes from the reduced costs to maintain multiple trees, since in this static scenario, most of nodes will eventually be promoted into the tier-1 overlay.

B. Dynamic Scenario

Similar to the static scenario, we let 200 nodes join the session at the beginning and the rest arrive following a Poisson process (with the average arrival rate being set to 6). To emulate node dynamics, a Pareto distribution is then used to model the nodes' durations in the session, where the default distribution parameters ($k = 1$ and $x_m = 1$ min) are adopted from the recently modeling work for peer-to-peer streaming in [6] as well as the PPLive traces. The simulation results are shown in Table V and Figs. 11–15.

Fig. 11 shows the results of the data loss rates of different systems. The default threshold for s-Index is set to 0.2 (the rationale of this setting will be explained later). By incorporating LBTree, the data loss rates of both PPLive and

TABLE V
AVERAGE VALUES OF DATA LOSS RATE, STARTUP LATENCY, AND PLAYBACK DELAY (DYNAMIC SCENARIO)

Approach	Data Loss Rate	Startup Latency	Playback Delay
PPLive	0.0182	53.0206	137.7132
ChunkySpread	0.0505	54.7610	89.3708
PPLive + LBTree	0.0005	26.1336	57.1269
ChunkySpread + LBTree	0.0243	52.3060	85.3945

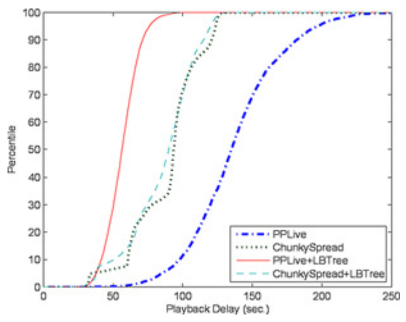

Fig. 13. CDF of playback delay (dynamic scenario).

ChunkySpread are significantly reduced. This indicates that the tier-1 LBTree successfully amplifies the data availability of the source and thus facilitates peers to quickly find available data before playback deadlines in the presence of the peer churns. The comparison of startup latency is given in Fig. 12. Again, significant improvements can be observed when PPLive works with LBTree. Even for the multitree-based ChunkySpread, adding LBTree can further reduce the startup latencies for most of nodes. Similar trends can be observed on the playback delay, as shown in Fig. 13.

Comparing with the results in the static scenario, the performance of the original ChunkySpread in this dynamic scenario has degraded noticeably, largely because of the high-cost of multiple tree repairing. Fortunately, adding LBTree can noticeably mitigate the impact. For PPLive, its relatively stable performance verifies its reliability and robustness against node churns, and by incorporating LBTree, its performance can be greatly improved, too.

We next evaluate the control overhead for different thresholds of the s-Index. The results are shown in Fig. 14. Note that the original PPLive and ChunkySpread do not identify stable nodes and hence their overheads are constant with the s-Index threshold. Similar to the static scenario, the original PPLive has the highest overhead due to using pull operations for data delivery. LBTree in the tier-1 overlay eliminates the overhead caused by pull operations and thus greatly reduce the overall control overhead. For ChunkySpread, the savings however are insignificant. This is because ChunkySpread and LBTree both use tree structures, and there is also transition overhead for a peer to be promoted to tier-1 overlay when incorporating the LBTree. Nevertheless, the overhead of ChunkySpread after incorporating LBTree does decrease for a larger s-Index threshold, e.g., 0.4 to 1.

Comparing with the static scenario, the control overhead of PPLive + LBTree and ChunkySpread + LBTree is relatively higher than that in the static scenario. This is because in

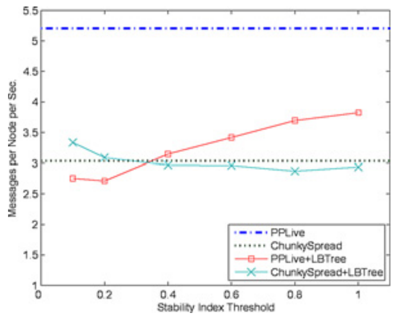

Fig. 14. Control overhead as a function of s-index threshold (dynamic scenario).

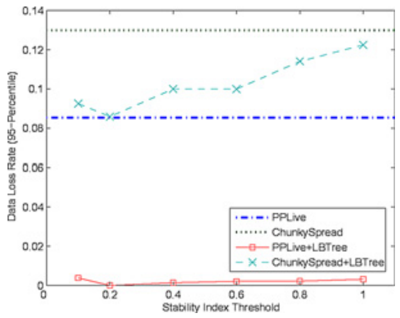

Fig. 15. Data loss rate as a function of s-index threshold (dynamic scenario).

the dynamic scenario, much more nodes stay in the tier-2 overlay, where the control overhead is in general higher and thus increases the average level.

In addition to the control overhead, we also evaluate the data loss rates for different thresholds of the s-Index. The results are shown in Fig. 15. From both Figs. 14 and 15, it is easy to see that the optimal threshold of s-Index seems to be 0.2. It suggests that this low value of s-Index is sufficient to filter out most transient nodes, and thus well predict a node's future stability. If we use too high an s-Index, e.g., close to 1, as the threshold, then we might identify and promote a potentially stable node too late, and hence greatly shorten its service time in the tier-1 overlay. The threshold of 0.2, together with our randomized promotion algorithm, achieves the best balance between the stability and scale for tier-1 overlay. As shown previously in our other simulations, it has served as a reasonable default setting.

VII. CONCLUSION

In this paper, we presented a systematic in-depth study on the existence, importance, and application of stable nodes in peer-to-peer live video streaming. We found that their long lifespans make them constitute a significant portion in a per-snapshot view of streaming overlays. As a consequence, the stable nodes have played an important role in overlay data distribution, which challenges the traditional belief that they are too small a group to be utilized. Our observations were validated through both trace analysis on a large-scale real system and mathematical modeling. It inspired a tiered overlay design, with stable nodes being organized into a tier-1 backbone for serving more transient tier-2 nodes. We presented an effective prediction algorithm for identifying potentially stable nodes. We further developed a novel structure, LBTree, for tier-1 overlay. It simultaneously achieves low overhead and high transmission reliability by leveraging the stable nodes, and works well with diverse overlay structures in tier-2. We extensively evaluated the performance of our approach by incorporating it to PPLive, a mesh-based overlay, and ChunkySpread, a multitree-based overlay. Our simulation results demonstrated that the customized optimization for stable nodes noticeably boosts the overlay performance, in terms of data loss rate, startup latency, and playback delay, and also effectively reduces the control overhead.

Future Work: Next, we would like to explore other factors that affect the performance of our tiered overlay, e.g., outbound bandwidth and peer location. These factors can also be integrated into the s-Index to further improve the performance. We are also interested in the case with assistance from dedicated proxies or servers from cloud services (i.e., s-Index = 1), where dynamically deployed or leased proxies/servers can further improve the overall performance such as QoS of video streaming, traffic locality, and maintenance costs.

Acknowledgment

The authors would like to thank Dr. X. Hei from Polytechnic University, Brooklyn, NY, for providing PPLive traces. The authors would also like to thank the administrator and technicians of PPLive for providing the PPLive video accelerator traces.

References

[1] *PPLive* [Online]. Available: http://www.pplive.com

[2] *PPLive Video Accelerator* [Online]. Available: http://www.ppacc.com

[3] K. C. Almeroth and M. H. Ammar, "Collecting and modeling the join/leave behavior of multicast group members in the MBone," in *Proc. IEEE HPDC*, Aug. 1996, pp. 209–216.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, 2002, pp. 205–220.

[5] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in *Proc. IEEE INFOCOM*, vol. 2. Mar.–Apr. 2003, pp. 1521–1531.

[6] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering priority in overlay multicast protocols under heterogeneous environments," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–13.

[7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proc. ACM SOSP*, 2003, pp. 298–313.

[8] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS*, 2000, pp. 1–12.

[9] A. J. Ganesh, A. M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Trans. Comput.*, vol. 52, no. 2, pp. 139–149, Feb. 2003.

[10] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proc. ACM SIGCOMM*, 2006, pp. 147–158.

[11] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[12] 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 *Proc. ACM SIGCOMM*, 2008, pp. 375–388.

[13] D. Kostic, R. Braud, C. Killian, E. Vandekieft, J. W. Anderson, A. C. Snoeren, and A. Vahdat, "Maintaining high bandwidth under dynamic network conditions," in *Proc. USENIX Annu. Tech. Conf.*, 2005, pp. 193–208.

[14] B. Li, S. Xie, Y. Qu, G. Y. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 1031–1039.

[15] Z. Li, D. H. Tsang, and W.-C. Lee, "Understanding sub-stream scheduling in P2P hybrid live streaming systems," in *Proc. IEEE INFOCOM Mini-Conf.*, Mar. 2010, pp. 1–5.

[16] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-peer live streaming," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–10.

[17] F. Liu, B. Li, L. Zhong, B. Li, and D. Niu, "How P2P streaming systems scale over time under a flash crowd?" in *Proc. USENIX IPTPS*, Apr. 2009, p. 5.

[18] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proc. IEEE*, vol. 96, no. 1, pp. 11–24, Jan. 2008.

[19] Z. Liu, C. Wu, B. Li, and S. Zhao, "Distilling superior peers in large-scale P2P streaming systems," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 82–90.

[20] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver-driven mesh-based streaming," in *Proc. IEEE INFOCOM*, May 2007, pp. 1415–1423.

[21] B. Mitra, A. K. Dubey, S. Ghose, and N. Ganguly, "How do superpeer networks emerge?" in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

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

[23] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proc. IPTPS*, 2005, pp. 127–140.

[24] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proc. ACM IMC*, 2004, pp. 41–54.

[25] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, and W. Zhu, "Robust and efficient path diversity in application-layer multicast for video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 8, pp. 961–972, Aug. 2005.

[26] D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: An efficient peer-to-peer scheme for media streaming," in *Proc. IEEE INFOCOM*, vol. 2. Mar.–Apr. 2003, pp. 1283–1292.

[27] V. Venkataraman, K. Yoshida, and P. Francis, "ChunkySpread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *Proc. IEEE ICNP*, Nov. 2006, pp. 2–11.

[28] V. Vishnumurthy and P. Francis, "On heterogeneous overlay construction and random node selection in unstructured p2p networks," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–12.

[29] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *Proc. IEEE ICDCS*, Jun. 2007, p. 49.

[30] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-upload decoupling: A redesign of multi-channel P2P video systems," in *Proc. IEEE INFOCOM Mini-Conf.*, Apr. 2009, pp. 2726–2730.

[31] M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang, "A peer-to-peer network for live media streaming: Using a push-pull approach," in *Proc. ACM Multimedia*, 2005, pp. 287–290.

[32] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *Proc. IEEE INFOCOM*, vol. 3. Mar. 2005, pp. 2102–2111.

Feng Wang (S'07) received the Bachelors and Masters degrees in computer science and technology from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently pursuing the Ph.D. degree from the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, conducting research on wireless sensor networks and peer-to-peer networks under the supervision of Prof. J. Liu.

In 2006, he was an Intern with the Wireless and Networking Group, Microsoft Research Asia, Beijing, where he conducted research on peer-to-peer live video streaming. He was as a Visiting Ph.D. Student with the Department of Computing at the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 2008 and 2009, where his research was on data collections in wireless sensor networks. In the spring of 2010, he visited the Wireless Ad Hoc Networks Laboratory, Institute of Software at Chinese Academy of Sciences, Beijing, where his research was on the testbed and information system design for wireless sensor networks. His current research interests include wireless sensor networks, peer-to-peer networks, and distributed computing.

Mr. Wang was awarded the Tsinghua University Scholarship for Excellent Student in 1998, 2000, and 2001. At Simon Fraser University, he was awarded the SFU-CS Graduate Entrance Scholarship in 2005, the Computing Science Fellowship in 2007, the Graduate Fellowship in 2008 and 2009, and the President's Ph.D. Research Stipend Award in 2011. He was also awarded the Chinese Government Scholarship for Outstanding Self-Financed Students Studying Abroad in 2009–2010.

Jiangchuan Liu (S'01–M'03–SM'08) received the B.E. degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2003, both in computer science.

He is currently an Associate Professor with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. From 2003 to 2004, he was an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong. His current research interests include multimedia systems and networks, wireless *ad hoc* and sensor networks, and peer-to-peer and overlay networks.

Dr. Liu was a recipient of the Microsoft Research Fellowship in 2000, the Hong Kong Young Scientist Award in 2003, and the Canada NSERC DAS Award in 2009. He was a co-recipient of the Best Student Paper Award of IWQoS in 2008, the Best Paper Award of IEEE ComSoc Multimedia Communications Technical Committee in 2009, and the Canada BCNet Broadband Challenge Award in 2009. He is a member of Sigma Xi. He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, an Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and an Area Editor of *Computer Communications*. He is the Technical Program Committee Vice Chair for Information Systems of IEEE INFOCOM'2011.

Yongqiang Xiong (M'04) received the B.S., M.S., and Ph.D. degrees from Tsinghua University, Beijing, China, in 1996, 1998, and 2001, respectively, all in computer science.

He is currently a Researcher with the Wireless and Networking Group, Microsoft Research Asia, Beijing. He has published over 40 papers, and served as a technical program committee member or reviewer for the international key conferences and leading journals in the areas of wireless and networking. His current research interests include peer-to-peer networking, routing protocols for both mobile *ad hoc* networks and overlay networks, and network security.