This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

Stable Peers: Existence, Importance, and Application in Peer-to-Peer Live Video Streaming

Feng Wang, Jiangchuan Liu[†], Yongqiang Xiong[‡]

[†]School of Computing Science, Simon Fraser University, British Columbia, Canada.
Email: {fwa1, jcliu}@cs.sfu.ca

[‡]Wireless and Networking Group, Microsoft Research Asia, Beijing, China.
Email: yqx@microsoft.com

***Abstract*—This paper presents a systematic in-depth study on the existence, importance, and application of stable nodes in peer-to-peer live video streaming. Using traces from a real large-scale system as well as analytical models, we show that, while the number of stable nodes is small throughout a whole session, their longer lifespans make them constitute a significant portion in a per-snapshot view of a peer-to-peer overlay. As a result, they have substantially affected the performance of the overall system. Inspired by this, we 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*, 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. Our extensive simulation results demonstrated that the customized optimization using selected stable nodes boosts the streaming quality and also effectively reduces the control overhead. This is further validated through prototype experiments over the PlanetLab network.

I. Introduction

Recently, there has been significant interest in the use of peer-to-peer technologies for live video multicast 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 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 download. 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. Dealing with the node dynamics (known as *churn*) under bandwidth and timing constraints thus becomes a necessary yet challenging task.

It is known that a tree structure, stemmed from IP multicast, severely suffers from node churn. Multi-tree and mesh overlays accommodate node dynamics better. However, multi-tree incurs higher construction and maintenance overhead, and mesh suffers from the delay and message overhead due to its per-block pull operation [21] [14]. The trade-offs 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.

This paper takes 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

This research is supported by a Canadian NSERC Discovery Grant and an NSERC Strategic Project Grant.

compensates this, and more importantly, they do not suffer from the 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,* 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 well accommodates flash crowd, that is, many nodes join the overlay in a short time. Such results are further validated by our prototype experiments over the PlanetLab [1].

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 labeled tree 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 the 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 [14], 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][20]. 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][16][19]. 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 [2][12][13][17][24].

Both tree/multi-tree 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 pre-defined 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., ChunkySpread [21], PRIME [15] and mTreebone [23]. 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) 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 study, we further present customized algorithms to effectively identify stable nodes, and then organize them into a separated overlay to maximize their contributions.

III. Existence and Importance of Stable Nodes

In this section, we try to answer the following fundamental questions: Do stable nodes exist in real peer-to-peer streaming systems? 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 mesh-based system, PPLive [11]. PPLive is the largest commercial peer-to-peer live streaming system to date, which attracts over 100,000 online users during peak times. Our investigation is based on traces of two popular PPLive channels, namely, CCTV3 and DragonBall, from Wednesday Nov 22 17:40 2006 to Thursday Nov 23 21:30 2006. These traces are gathered by an online crawler that continuously collects information from each channel. To mitigate the *time-of-day effects* [18], 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.

In Fig. 1, we plot the complementary cumulative distribution functions (CCDFs) of node 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 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 Tab. 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

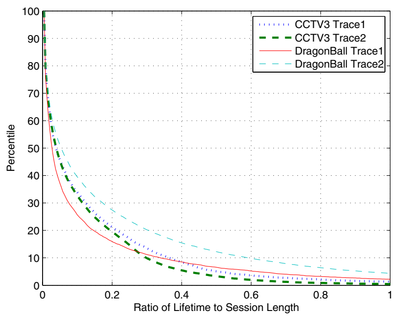

Fig. 1. CCDF of life time distribution in different traces.

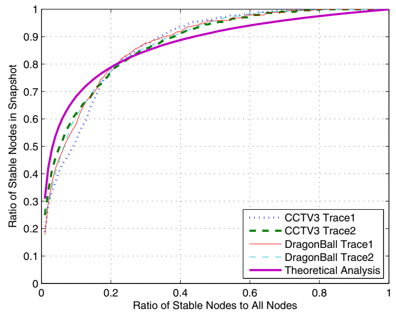

Fig. 2. Ratios of stable nodes (different minimum lifetimes are used to define stable nodes).

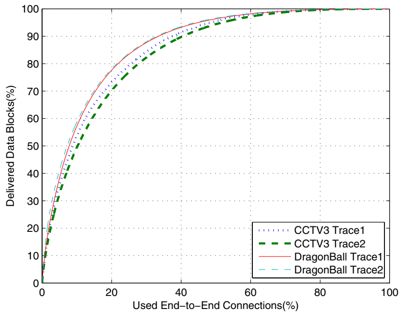

Fig. 3. CDF of data blocks delivered by different node-pair connections.

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

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

where I is the identity function.

It is known that, in a peer-to-peer system, the node lifetime can be approximated by a Pareto distribution. This is also verified by the PPLive traces and we find that the typical parameters are $k = 1$ and $x_m = 1$ minute. Assume node arrival rate is μ, we can calculate the expected ratio of stable nodes in the overlay snapshot at time T as

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

Using different minimum lifetimes to define stable nodes, Fig. 2 show the ratio of stable nodes from both session views and snapshot views. It is clear to see that, for a session time of 4 hours, when the ratio of stable nodes in the whole session is about 20%, the expected ratio of stable nodes is 78.8%. 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. 3 shows the cumulative distribution function (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 (<<1%) of representative trees. The representativity, defined as the ratio of common internal links between a per-block-tree tree and its

Data Set	Avg. Ratio of Stable Nodes per Snapshot	Ratio of Stable Nodes per Trace	Ratio of Representative Trees to All Per-Block-Trees	Avg. 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%

TABLE I
STATISTICS OF STABLE NODES AND REPRESENTATIVE PER-BLOCK-TREES.

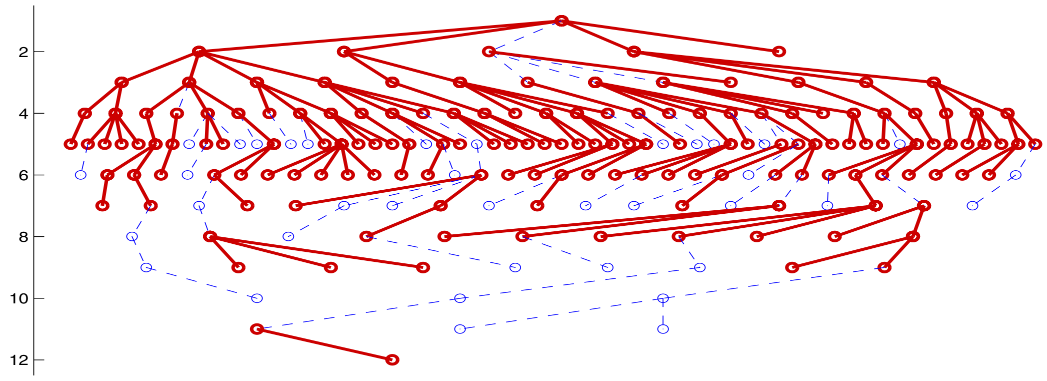

Fig. 4. A snapshot of a representative per-block-tree from CCTV3 Trace-1. The thick circles and lines correspond to stable nodes and their connections.

best-matching representative, is on average close to 80% (see Tab. 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 multi-neighbor 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 (as can be seen from Fig. 4, a highly unbalanced tree). 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 2-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 semi-stable 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 as well as their interactions with the tier-2 nodes, 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,

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

where SI is the value of s-Index, s is 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 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 experiments; yet, other known information could also be incorporated 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 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. To realize the linearly increasing probability SI/H, each tier-2 node independently checks its status per unit time; for the s-th 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 4.1: In the above promotion algorithm, at (discrete) time s (with $SI < H$), the probability that a node currently of duration s is predicted stable and promoted to tier-1 is SI/H.

Proof: The probability is given by

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

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

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

■

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

V. Organization of Stable Nodes: The Labeled Tree 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 eliminated. 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.

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

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.

To this end, we introduce *Labeled Tree* (or LBTree in short), 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. Overview of Labeled Tree

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 non-overlapped 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 a LBTree of 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][22]. Using the labels, it is easy to identify the relationship of between two nodes in the tree. Tab. II gives three basic relationships between labels, which facilitates 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 except 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 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 stability index is generally higher for nodes closer to the source, and, since s-Index 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 locally closest to the source only, it may locate other closer

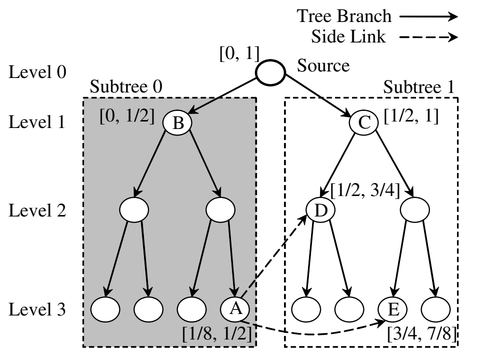

Fig. 5. A LBTree (labeled tree) with side links.

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 experimental 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 recovery operations are expected beyond the mesh or multi-tree approaches.

Our solution is to establish *Side Link* between eligible node pairs in the LBTree for loss recovery. The side links make effective use of the *bandwidth remainder*, 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 1Mbps and a streaming with the rate 384Kbps, there remains 256Kbps bandwidth which can not 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. To effectively recover the lost data, the side links are placed according the following criteria (referred to Fig. 5):

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. 5, 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 block on time. Note that, either A or D can initiate the link construction, because the direction can be easily identified from label comparison.

2) The two ends of a side link should not share any ancestor except for the source, which we refer to as *orthogonal*. As an example, in Fig. 5, node A should not have 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 local label comparison as follows:

x and y are orthogonal, if and only if $\forall i \in \{0 \dots (N_s - 1)\}$, $((a_x, b_x) \cup (a_y, b_y)) \nsubseteq (a_s + \frac{i(b_s - a_s)}{N_s}, a_s + \frac{(i+1)(b_s - a_s)}{N_s})$.

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 however two fundamental differences: 1) In the 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 [21]. 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 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. And surprisingly, we find that with a probability of at least 99% for a missing segment to be recovered before its playback deadline, even the residual bandwidth of a node is only 20% of the streaming rate, it still can support more than 8 side links serving nodes that suffer 1% data loss or 1 side link serving a node that suffers up to 10% data loss.

VI. Performance Evaluation

To validate our analysis and evaluate the proposed algorithms for identifying and organizing stable nodes, we have conducted extensive simulations as well as experiments using a PlanetLab-based [1] prototype. For the sake of comparison, we have also implemented two state-of-the-art systems, namely, ChunkySpread [21] and a PPLive-like system. ChunkySpread is a multi-tree 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 it is not open-source, our implementation is based on the trace data as well as its protocol analysis from [11] and various other sources.

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

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;

Startup latency, which is the time taken by a peer between its requesting to join the session and receiving enough data blocks to start playback;

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*, that is, 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 excessive total 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.

A. Simulation Results

Unless otherwise specified, the following default parameters are used in our simulation, most of which follow the typical values reported in [3][11]. The session length L is set to 3600 seconds and each data block is of 1-second 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 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][21], respectively.

To emulate node dynamics, we let 200 nodes join the session at the beginning and the rest arrive following a Poisson process. A Pareto distribution is used to model the nodes' durations in a session [3][18]. The default parameters of the distributions are adopted from the recently modeling work for peer-to-peer streaming in [6] as well as PPLive traces.

Fig. 6 shows the cumulative distribution function (CDF) of the data loss rates of the pure PPLive/ChunkySpread and our tiered system (PPLive+LBTree/ChunkySpread+LBTree). The default threshold for s-Index is set to 0.2 (the rationale of this setting will be explained later). It is clear that, with the support of LBTree, the data loss rates of both PPLive and ChunkySpread can be significantly reduced. This is because LBTree successfully amplify the data availability of the source and thus facilitates peers to quickly find available data before playback deadline despite the churn of other peers. The comparison of startup latency is given in Fig. 7. Again, PPLive enjoys noticeable improvement by incorporating the LBTree. For Chunkyspread, since it is a tree-based system with no latency due to data pull, its startup latencies with and without LBTree have no significant difference, though the case with LBTree is slightly better. Similar trends can be observed on the playback delay, as shown in Fig. 8.

We next evaluate the control overhead for different thresholds of the s-Index. The results are shown in Fig. 9. Note that the original PPLive and ChunkySpread do not identify stable nodes and hence their overheads are constant with the s-Index threshold. We can see that 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 eliminatd pulling and the tier-2 overlay has a reduced scale, the total control overhead is significantly reduced. For ChunkySpread, the gain in terms of overhead is 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.

B. PlanetLab Experimental Results

To further investigate the effectiveness of LBTree, we have implemented a prototype that integrates LBTree (tier-1) with PPLive (tier-2), and have conducted a series of experiments over the PlanetLab. A total of 200 PlanetLab nodes participated in our experiments. The session length is set to 1800 seconds. Each node joins the session at the beginning and its duration follows the same distribution as in the simulations. After a node leaves, it will sleep for a random period and rejoin the session as a new node, so as to emulate a larger user base with churn. The streaming rate is set to 400Kbps, which is a typical upper bound in most of PPLive channels [11].

Figs. 11-13 show data loss rate, startup latency and playback delay of PlanetLab experiments. It is clear that, for all the three metrics, the use of LBTree has substantially improved the performance of PPLive. We also vary the the s-Index threshold to examine its impact, and the results are shown in Figs. 14 and 15. These results are consistent with that of our simulation, and thus reaffirm the effectiveness of using LBTree to boost the performance of existing peer-to-peer overlays.

Note that the optimal threshold of s-Index seems to be 0.2 for both simulations and experiments (Figs. 9-10 and 14-15). It suggests that this low value of stability index is sufficient to filter out most transient nodes, and thus well predicts 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. It has served as a reasonable default setting in our other simulations and experiments, as shown before.

Another interesting observation is that, except for data loss, the performance measures of PPLive are generally better in

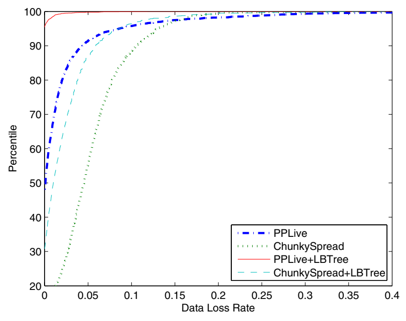

Fig. 6. CDF of data loss rate (simulation).

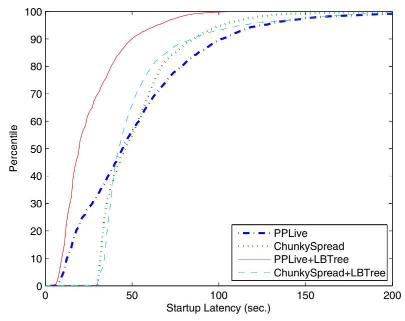

Fig. 7. CDF of startup latency (simulation).

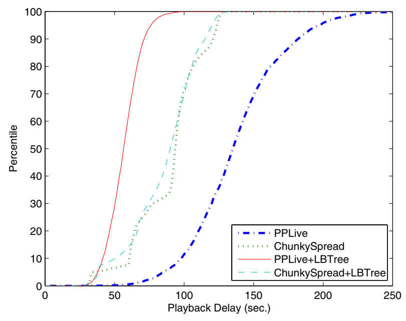

Fig. 8. CDF of playback delay (simulation).

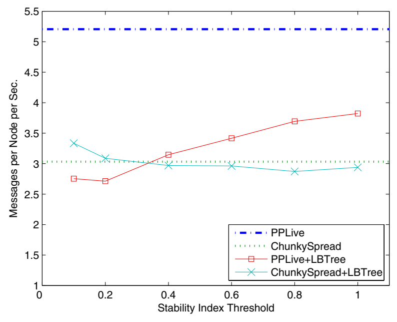

Fig. 9. Control overhead as a function of *s*-index threshold (simulation).

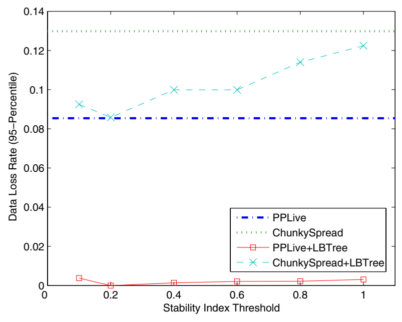

Fig. 10. Data loss rate as a function of *s*-index threshold (simulation).

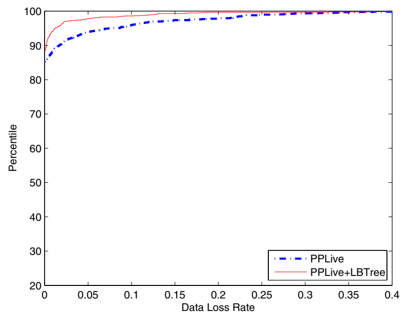

Fig. 11. CDF of data loss rate (PlanetLab).

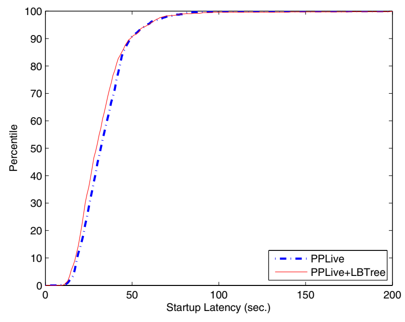

Fig. 12. CDF of startup latency (PlanetLab).

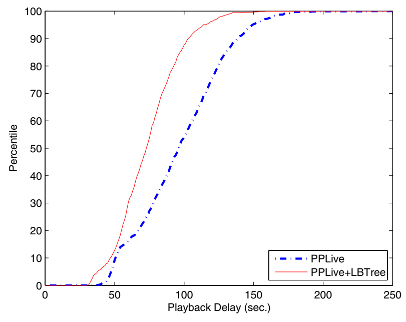

Fig. 13. CDF of playback delay (PlanetLab).

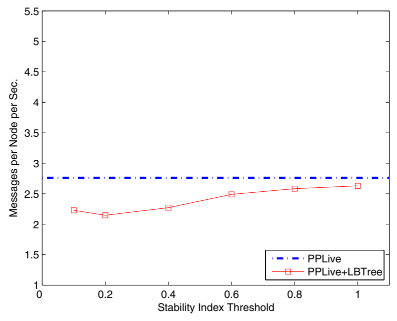

Fig. 14. Control overhead as a function of *s*-index threshold (PlanetLab).

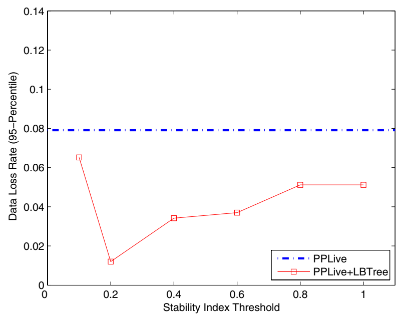

Fig. 15. Data loss rate as a function of *s*-index threshold (PlanetLab).

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

the experiments than in the simulation. We have taken a close investigation and found that there are two reasons:

1) In the simulation, we let 200 nodes join the session exactly at the beginning of the session, which serves as a flash crowd; in the experiments, though we still intended to introduce a flash crowd, we notice that its impact is largely reduced due to the asynchronicity among the PlanetLab nodes and the delay for command dispatching. After a flash crowd, newly joined nodes often become neighbors to each other and thus form an island. PPLive has implemented mechanisms to unlock the island, but this takes time and introduces extra overhead. Our tiered overlay however accommodates flash crowed much better.

2) The scale of the experiments is relatively smaller. In PPLive, there is a constraint on the number of neighbors that a peer may have, plus that once a connection between two stable peers is established, it is very likely to be kept until one of the peers leaves the session. Hence, when the overlay grows over time, it takes longer time for a new node to establish neighbor relations from its partial list of overlay nodes. Although the scale of our experiments is constrained by available PlanetLab nodes, we have conducted experiments with longer sessions, which show the startup latency, playback delay and overhead of PPLive do increase with the session length.

VII. Conclusion and Future Work

In this paper, we systematically investigated the existence, importance, and application of stable nodes in peer-to-peer live video streaming. We found that, the long lifespan make them constitute a significant portion in a per-snapshot view of the overlay. As a consequence, the stable nodes have played an important role in overlay data distribution, which is contrast to traditional belief that they are too small a group to be utilized. Such observations were validated through both trace analysis of a large scale real system and through mathematical modeling. It inspired a tiered overlay design, with stable nodes being organized into a tier-1 backbone for serving tier-2 nodes. We presented an effective prediction algorithm for identifying stable nodes. We further developed a novel structure, *labeled tree*, 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 multi-tree-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. This was further verified by our prototype experiments (combination with PPLive) over the PlanetLab network.

We are currently examining the performance of our tiered overlay in larger scale networks, possibly with assistance from dedicated proxies (i.e., s-Index=1). We are also interested in developing better predictors, as well as incorporating other factors into the stability index, e.g., outbound bandwidth and peer location, to further improve the overlay performance.

Acknowledgements

The authors thank Dr. Xiaojun Hei from Polytechnic University, Brooklyn, USA, for providing PPLive traces.

References

[1] PlanetLab. [Online]. Available: http://www.planet-lab.org/

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

[3] K. C. Almeroth and M. H. Ammar, "Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the MBone," in *IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 1996.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *ACM SIGCOMM*, 2002.

[5] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications," in *IEEE INFOCOM*, 2003.

[6] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments," in *IEEE INFOCOM*, 2006.

[7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *ACM SOSP*, 2003.

[8] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in *ACM SIGMETRICS*, 2000.

[9] A. J. Ganesh, A. M. Kermarrec, and L. Massoulie, "Peer-to-Peer Membership Management for Gossip-based protocols," *IEEE Transactions on Computers*, no. 2, pp. 139–149, February 2003.

[10] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing Churn in Distributed Systems," in *ACM SIGCOMM*, 2006.

[11] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "Insights into PPLive: A Measurement Study of a Large-scale P2P IPTV System," in *Workshop on Internet Protocol TV (IPTV) Services over World Wide Web*, 2006.

[12] 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 *USENIX Anual Technical Conference*, 2005.

[13] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming," in *IEEE INFOCOM*, 2006.

[14] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," *to appear in Proceedings of the IEEE*, 2007.

[15] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming," in *IEEE INFOCOM*, 2007.

[16] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanid-kulchai, "Distributed Streaming Media Content Using Cooperative Networking," in *ACM NOSSDAV*, 2002.

[17] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast," in *IPTPS*, 2005.

[18] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet," in *Internet Measurement Conference*, 2004.

[19] 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 Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, pp. 961–972, August 2005.

[20] D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," in *IEEE INFOCOM*, 2003.

[21] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast," in *IEEE ICNP*, 2006.

[22] V. Vishnumurthy and P. Francis, "On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks," in *IEEE INFOCOM*, 2006.

[23] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast," in *IEEE ICDCS*, 2007.

[24] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming," in *IEEE INFOCOM*, 2005.