Path Selection in Streaming Video Over Multioverlay Application Layer Multicast

Yang Y. Lin and Jack Y. B. Lee, *Senior Member, IEEE*

***Abstract*—Application-layer multicast (ALM) has grown tremendously in recent years, making the distribution of bulk data such as streaming video economically feasible for small companies and even individuals. The efficiency of an ALM network depends on its data distribution overlay, which is constructed based on metrics such as round-trip time (RTT) measurement between peers. However Internet measurement experiments revealed that RTT is far from an accurate estimator of bandwidth availability and as such, may lead to sub-optimal performance in the constructed ALM overlays. This paper tackles this problem by developing a new in-band bandwidth probing tool which can estimate the amount of achievable bandwidth available in the target network path so that excess data traffic can be diverted from the congested path without causing new congestion in the target path. Moreover, the probing tool does not incur any bandwidth overhead as it piggybacks on the existing data flow. Simulation results show that multioverlay ALM networks constructed based on achievable bandwidth consistently out-performs RTT-based and residual bandwidth-based approaches in terms of data delivery ratio and video playback continuity. Moreover, the proposed bandwidth probing tool can be implemented entirely within the application and thus can be readily incorporated into existing ALM protocols.**

***Index Terms*—Application layer multicast, bandwidth estimation, overlay network, video streaming.**

I. Introduction

APPLICATION-LAYER multicast (ALM) has grown tremendously in recent years, making the distribution of bulk data such as multimedia data economically feasible for small companies and even individuals. More recently, ALM has been further applied to bandwidth-demanding applications such as video streaming to take advantage of its bandwidth efficiency [1]–[6].

The principle of ALM is to organize participating peers into one or more virtual networks, or called overlays, on top of the physical network, and then distribute data along the logical paths in the overlays. Naturally, construction of the overlay topology is critical to its performance and, therefore, much research has been done in this area [7].

Manuscript received August 26, 2009; revised January 1, 2010 and March 9, 2010. Date of publication May 27, 2010; date of current version July 16, 2010. This work was supported in part by the Shun Hing Institute of Advanced Engineering, Chinese University of Hong Kong, Shatin, Hong Kong, under Project MMT 9/07. This paper was recommended by Associate Editor P. Frossard.

Y. Y. Lin is with Harmonic, Inc., Kowloon, Hong Kong (e-mail: leon.lyy@gmail.com).

J. Y. B. Lee is with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: jacklee@computer.org).

Digital Object Identifier 10.1109/TCSVT.2010.2051280

Common among many of the existing works is the use of round-trip time (RTT) between peers as the metric in selecting paths for overlay construction [8]–[10]. As peers farther apart geographically tend to have longer RTT between them, by favoring short RTT, the system can exploit the geographic locality of peers to reduce the number of links that the data have to traverse. Moreover, nearby peers are more likely to share high-speed network links which improves performance further. Finally, RTT can also be used to indirectly detect network congestion as queuing delay during congestion will cause the RTT to increase.

Given the wide-spread adoption of the RTT metric in overlay construction, it is therefore important to investigate its actual performance in path selection. In this paper, we first report experimental results obtained from measurements conducted in the Internet to quantify the performance of using the RTT metric in path selection. Contrary to common believes, RTT may not always provide accurate estimation of bandwidth availability when used in certain configurations. For example, in one of our experiments, if RTT is used to select between two paths then it will correctly identify the higher-bandwidth path only 67.3% of the time, i.e., slightly better than random (see Section III).

In addition to the RTT metric, researchers have also employed residual bandwidth in path selection [11], [12]. Residual bandwidth is defined as the minimum unused capacity of the links along a path and it can be estimated from sending probing packets [13], [14] to the next peer in the overlay topology. Nevertheless, due to protocol interactions, residual bandwidth is not the same as bandwidth usable by an application and our results show that while the residual bandwidth metric performs better than RTT, it is still far from optimal.

More recently, an increasing number of ALM protocols began to employ not one, but multiple overlays for data distribution. Multioverlay ALM protocols can exploit path diversities in the network to de-correlate packet loss [15], to explore more available network bandwidth [16], and to increase resilience to local network failures as well as peer churn [17].

In a multioverlay ALM protocol, the source first splits the original data stream into multiple, say N, sub-streams and then distributes them over the N overlays. Each peer establishes up to N connections to other parent peers according to the overlay topologies to receive and then also forward the sub-streams to its downstream peers along the overlay networks.

Thus, each peer is continuously exchanging data with at least N peers. Apart from the data transported, these N connections also provide indirect information of the paths' bandwidth availabilities. This motivated us to consider an alternative metric for path selection—the actual throughput achieved in a path, referred to as *achievable bandwidth*.

We develop a new in-band bandwidth probing tool which can estimate achievable bandwidth, i.e., the data throughput that can be realized between two peers over the transport protocol employed [e.g., transmission control protocol (TCP)]. Unlike the RTT and residual bandwidth metrics this new tool can determine the amount of extra bandwidth available in the target network path so that excess data traffic can be diverted from congested path without causing new congestion in the target path. Moreover, the probing tool does not incur any bandwidth overhead as it obtains its measurements as a by-product of transporting actual data (as opposed to probing packets).

To analyze and compare the performance of the RTT and achievable bandwidth metrics in path selection, we developed a multioverlay ALM protocol to evaluate the two metrics under the same simulation settings. Our results show that: 1) packet loss across the overlay networks is not entirely due to network congestions, but also due to topology changes as well; 2) the RTT metric results in significantly more topology changes due to inherent variations in the measured RTT and the fact that topology change itself can also affect the RTT of a path; and 3) only the achievable bandwidth metric can result in converged overlay topologies. These results strongly suggest that the use of achievable bandwidth metric can offer substantially better performance than the RTT and residual bandwidth metrics in multioverlay ALM.

The rest of the paper is organized as follows. Section II reviews the previous related work. Section III reports experimental results to evaluate the accuracy of using RTT to finding network paths with higher bandwidth. Section IV presents the proposed in-band bandwidth probing mechanism and analyzes its delay tradeoffs. Section V presents the reference ALM protocol we developed to quantitatively compare the two path selection metrics in a controlled environment. Section VI reports simulation results to compare the two path selection metrics' performance in the reference ALM protocol. Section VII concludes the paper and outlines some future work.

II. Related Work

We review below two categories of related work in overlay networks, namely latency-based and bandwidth-based approaches. The focus is in the metrics being used in the construction and adaptation of the overlay topology, and the way such metrics are estimated.

A. Latency-Based Approaches

Latency, typically measured in the form of RTT, has been widely used as the metric for overlay construction. Narada [8] is an early study to investigate the feasibility of implementing multicast capability in end hosts. Since Narada was designed for delay-sensitive video conferencing applications, the latency of overlay links was used as the primary routing metric to minimize end-to-end delay.

The NICE protocol [9] was designed to support real-time data applications with large receiver sets. To keep the control overhead for an average peer constant regardless of system population, the protocol clusters peers into a hierarchy. Peers are clustered according to the distance metric derived from round-trip latency estimations. The data delivery tree is then constructed from the hierarchy formed.

Topology-aware hierarchical arrangement graph (THAG) [5] is a scheme targeting live streaming applications. In THAG, the adjacent hosts are organized into a group called an arrangement graph (AG), and hosts serve each other within the same group. The AGs are then organized into a hierarchical architecture. To reduce propagation delay for live streams, hosts closer (latency-wise) to the source will be assigned to higher level AGs. Multiple overlay trees are embedded in each AG for data delivery with the trees constructed in a way similar to SplitStream [15].

There are numerous other overlay protocols [18], [19] which employ latency as the metric to construct and maintain their overlay topologies. Due to space limitation the reader is referred to the survey by Hosseini *et al*. [20] for more comparisons.

B. Bandwidth-Based Approaches

For clarity we define three types of bandwidth: 1) link bandwidth—this refers to the maximum bandwidth capacity of the bottleneck link along a network path; 2) residual bandwidth—this refers to the unused bandwidth along a network path; and 3) achievable bandwidth—this refers to the data throughput achievable by a given congestion-aware transport protocol (e.g., TCP [21], TCP friendly rate control (TFRC) [22], etc.) along a network path.

Most existing work employed residual bandwidth as the metric for overlay construction. For example, Overcast [11] is an early single-tree ALM protocol designed to maximize bandwidth between receiving hosts and the source at the root of the tree. It employs explicit bandwidth probing to determine the initial location to insert new hosts into the existing tree overlay and also reevaluates the bandwidth availability periodically using probing to adapt to changes in the network.

LION [23] is a more sophisticated ALM protocol which employs multiple overlays for the delivery of multilayer-encoded data. The mesh overlays are constructed based on bandwidth information measured using active probing tools.

BARON [24] is a bandwidth-aware routing scheme for overlay networks that target at bandwidth-sensitive applications. When a route between two end hosts is experiencing congestion, BARON finds candidate alternate paths based on link bandwidth and from that selects the best one according to residual bandwidth.

In another work, Jain and Dovrolis [25] proposed to use residual bandwidth as the metric in a link-state overlay routing protocol for video streaming. They found that residual bandwidth can result in better video quality compare to other metrics such as loss ratio and jitter. Their residual bandwidth

measurement was also in-band using data traffic, but they have only considered overlay networks built by content providers with up to two hops.

C. Other Approaches

Besides latency and bandwidth metrics, researchers have also developed ALM protocols based on other metrics. For example, Chunkyspread [17] constructs multitree overlay based on data delivery delays. TAG [26] exploits knowledge of the physical network topology in constructing its logical overlay tree. The principle is to align the physical and logical topologies so that data will traverse the same path as defined by the routing protocol in the underlying network.

D. Comparisons and Contributions

This paper tackles the path selection problem using a new metric—achievable bandwidth, which has three desirable properties. First, compare to latency-based approaches achievable bandwidth can offer better accuracy in selecting paths with higher bandwidth as confirmed by experiments in two different network environments (see Section III).

Second, using link capacity to construct an overlay ignores the effect of competing traffics. It is easy to see that a congested high-capacity link may in fact be a poorer choice than an idle low-capacity link. At the other extreme, residual bandwidth is very conservative and thus will likely limit the overlay's performance as it can only utilize the bandwidth leftover by other competing traffics. By contrast, achievable bandwidth can more accurately reflect the actual throughout that can be realized, with the impact of competing traffic and protocol interactions all accounted for.

Third, the achievable bandwidth metric allows the use of congestion-aware transport protocols such as TCP and TFRC for data delivery. This promotes fair sharing of bandwidth between the ALM protocol and other competing traffics, and ensures that the ALM protocol will react to alleviate network congestion in the same way as other Internet applications.

The use of achievable bandwidth in constructing ALM networks does present a new challenge. Specifically, existing bandwidth measurement tools either estimate the link capacity or the link's residual bandwidth. As these probing tools are not designed to measure achievable bandwidth they will not take into account the behavior of the transport protocol such as TCP. To illustrate this point consider a hypothetical link of capacity 2 Mb/s which has one existing TCP flow consuming 1.5 Mb/s of the bandwidth. An ideal probing tool will measure the link capacity and residual bandwidth to be 2 Mb/s and 0.5 Mb/s, respectively. Now, if an ALM network routes one of its data flow transported over TCP to this link then the new TCP flow will share bandwidth with the existing one, splitting the link capacity equally due to TCPs fair bandwidth sharing property.

Thus, the *actual* bandwidth that can be achieved by routing traffic to this link will be approximately 1 Mb/s (ignoring protocol overheads), which is clearly different from the bandwidth estimated using the existing probing tools that measure link capacity and residual bandwidth. Our results shown that the more accurate measurements obtained using the in-band bandwidth probing algorithm enable the ALM protocol to more efficiently utilize bandwidth in the network.

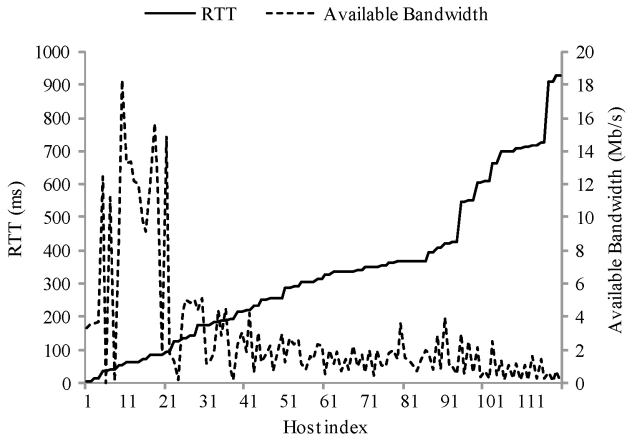

Fig. 1. Comparison of RTT and achievable bandwidth for 120 hosts in SpeedTest [28].

III. RTT-Based Path Selection Revisited

In this section, we report experimental results to evaluate the effectiveness of using RTT in ALM overlay construction and maintenance. We conducted two sets of experiments, one using hosts in the PlanetLab [27] and a second one using SpeedTest [28]—a network of servers for network performance measurements. In the PlanetLab experiment, we developed and deployed our own measurement software in more than 500 PlanetLab hosts. In each experiment run, one node pings its peer node several times to measure the average RTT. After the ping results are collected, it sends a 5 MB file to its peer node using TCP and measures the average throughput achieved.

Fig. 1 plots the measured RTT and the achievable bandwidth for 120 peers in the SpeedTest experiment. Upon inspection, there appears to be correlations between RTT and achievable bandwidth. To evaluate it quantitatively, we consider the following general path-selection problem: given the RTT measurements for X peers, select the Y peers with the shortest RTT as the new paths for data delivery.

Let r_i and b_i, $i = 0, 1, \ldots, (X-1)$, be the measured RTT and the achievable bandwidth for peer i, respectively. Without loss of generality, we sort them according to ascending order of RTT, i.e., $r_i \leq r_j$ for all $i < j$. Then, the combined bandwidth of the first Y peers as selected based on RTT is given by

$$B_{\mathrm{RTT}} = \sum_{i=0}^{Y-1} b_i. \tag{1}$$

Next, we re-sort b_i's into c_i's according to descending order of bandwidth, i.e., $c_i \geq c_j$ for all $i < j$. Then, we can obtain the maximum combined bandwidth of Y peers from

$$B_{\max} = \sum_{i=0}^{Y-1} c_i \tag{2}$$

TABLE I
COMPARISONS OF PATH SELECTION ACCURACIES α

SpeedTest	$X = 2$	$X = 3$	$X = 4$	$X = 5$	$X = 6$	$X = 7$
$Y = 1$	0.673	0.557	0.506	0.463	0.456	0.442
$Y = 2$	–	0.532	0.371	0.302	0.263	0.230
$Y = 3$	–	–	0.451	0.268	0.196	0.148
$Y = 4$	–	–	–	0.401	0.209	0.137
PlanetLab	$X = 2$	$X = 3$	$X = 4$	$X = 5$	$X = 6$	$X = 7$
$Y = 1$	0.794	0.700	0.626	0.591	0.566	0.543
$Y = 2$	–	0.691	0.555	0.470	0.398	0.347
$Y = 3$	–	–	0.649	0.506	0.405	0.312
$Y = 4$	–	–	–	0.585	0.450	0.357

TABLE II
COMPARISONS OF PATH SELECTION EFFICIENCY β

SpeedTest	$X = 2$	$X = 3$	$X = 4$	$X = 5$	$X = 6$	$X = 7$
$Y = 1$	0.833	0.762	0.726	0.692	0.673	0.661
$Y = 2$	–	0.870	0.813	0.784	0.761	0.745
$Y = 3$	–	–	0.902	0.846	0.822	0.797
PlanetLab	$X = 2$	$X = 3$	$X = 4$	$X = 5$	$X = 6$	$X = 7$
$Y = 1$	0.886	0.821	0.788	0.763	0.757	0.780
$Y = 2$	–	0.924	0.872	0.836	0.805	0.789
$Y = 3$	–	–	0.948	0.906	0.870	0.842

which is also the upper bound for B_{RTT}. Hence, how close B_{RTT} approaches $B_{\max}$ will give a quantitative measure of the goodness of using RTT for path selection.

Specifically, we define two performance indices for comparisons. The first one, denoted by α, is defined as the probability that RTT-based selection will result in the maximum combined bandwidth, that is

$$\alpha = \Pr\{B_{\mathrm{RTT}} \equiv B_{\max}\}. \tag{3}$$

The second one, denoted by β, is defined as the fraction of combined bandwidth as selected based on RTT to the maximum combined bandwidth, that is

$$\beta = \frac{B_{\mathrm{RTT}}}{B_{\max}}. \tag{4}$$

Tables I and II summarize these two performance indices for the two sets of experiments conducted in PlanetLab and SpeedTest. We first consider path selection accuracy a in Table I. For the simplest case of selecting one path ($Y = 1$) from two choices ($X = 2$), RTT correctly identified the higher-bandwidth path only 67.3% of the time, i.e., slightly better than random. The accuracy generally decreases as the number of choices increases (i.e., larger X) as the random probability of selecting the highest bandwidth path decreases. Depending on the experimental dataset and the particular $\{X, Y\}$ combination the accuracy can range from 0.137 to 0.794. This shows that RTT cannot consistently select the paths with the most combined bandwidth accurately.

Next, we consider bandwidth efficiency b in Table II. The results suggest that RTT-based path selection works better for smaller X (i.e., fewer choices) and larger Y (i.e., select more paths). However existing ALM protocols typically employ large X and small Y, thereby the constructed overlay could be far from optimal. For example, NICE employed $\{X = 3$ to 8; $Y = 1\}$ and Narada employed $\{X = 3$ to 6; $Y = 1\}$. Assuming $X = 5$ then a and b will become 0.463 and 0.692, respectively, which has substantial room for improvement.

IV. PATH BANDWIDTH MEASUREMENT

A peer in an overlay network is constantly exchanging data with multiple peers, so the actual throughput achieved already provides information on the path bandwidth available.

However, unlike file transfer applications such as FTP, video streaming applications typically transfer data at a prescribed data rate rather than as fast as possible. Thus, the actual throughput achieved between two peers can only indicate the minimum bandwidth available rather than the maximum bandwidth achievable (unless the throughput is lower than the prescribed video data rate).

For example, suppose the maximum bandwidth achievable between two peers is 1 Mb/s, while video data are transferred between the two peers at a prescribed data rate of 1.5 Mb/s. In this case, there is clearly not sufficient bandwidth to carry the video stream at the video data rate and so, depending on the implementation of the overlay/transport protocols, either substantial amount of data will be discarded or data delivery will be significantly delayed. Nonetheless, the receiving peer can still measure the throughput of the incoming data, e.g., at about 1 Mb/s, to estimate that the path bandwidth is in fact lower than the required video data rate.

On the other hand, if the path bandwidth is higher than the video data rate, e.g., at 3 Mb/s vs. 1.5 Mb/s, the receiving peer will still only measure a throughput of 1.5 Mb/s as the sending peer transmit data at the prescribed video data rate. This presents a problem as it means that unused achievable bandwidth in excess of the video data rate is not known to the peers.

To tackle this problem, we propose in this section an in-band bandwidth probing tool designed for video streaming applications. This probing tool has three desirable characteristics: 1) it does not require the transmission of additional probing packets (as in active bandwidth measurement tools); 2) it can be implemented at the application layer without modification to the transport protocol; and 3) it can probe for unused bandwidth in excess of the prescribed video data rate.

A. In-Band Bandwidth Probing

The principle of in-band bandwidth probing is to modulate the transmission timings of data packets. In particular, by appropriately delaying the transmission of data packets, the sending peer can compress the transmission duration of a set of consecutive packets and hence raise the transmission data rate above the video data rate, temporary for a period of time, to probe for additional bandwidth. Note that this does not affect the long-term average data rate, which will still be equal to the prescribed video data rate.

Fig. 2 illustrates the in-band bandwidth probing mechanism. Probing is performed independently and periodically by each peer once every *probing cycle*. A probing cycle begins with a *probing window* of K consecutive video packets, follows by normal video data transmission for a duration of $(n-1)$ times

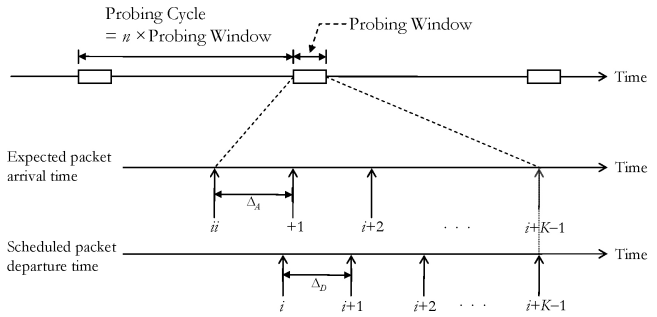

Fig. 2. Illustration of the in-band bandwidth probing mechanism.

the duration of the probing window. In a probing window, the peer will transmit video packets at a data rate higher than the prescribed video data rate to the receiving peer downstream. These packets can be marked (e.g., by a bit in the header field) so that the receiving peer will measure the incoming data rate during this probing window. If the measured data rate exceeds the prescribed video data rate then it implies that the path between the sending peer and the receiving peer possesses unused achievable bandwidth that can be used for traffic diversion (see Section V-C). Depending on the design of the ALM protocol, this bandwidth information can either be used directly by the receiving peer or it can be distributed to other peers or a rendezvous peer to initiate traffic diversion. See Section V-C for an example.

Let R_v be the prescribed video data rate and assume video data are divided into fixed-size packets of L bytes. Then, the expected incoming video packet inter-arrival time, denoted by Δ_A, is given by

$$\Delta_A = L/R_v. \tag{5}$$

To raise the outgoing data rate by a *probing factor* of f ($f > 1$), we need to shorten the video packet inter-departure time, denoted by Δ_D, to

$$\Delta_D = L/(fR_v). \tag{6}$$

To achieve this, the sending peer will delay the forwarding of incoming video packets so that the K packets in the probing window can be transmitted in a shorter time window and hence higher outgoing data rate. Specifically, let a_i and a'_i be the actual and expected time for data packet i to arrive at the sending peer and let d_i be the scheduled departure time for transmitting packet i to the downstream receiving peer. Assume the probing window consists of data packets i to $i+K-1$. Then, we first schedule the departure time of packet $(i+K-1)$ to the packet's expected arrival time

$$d_{i+K-1} = a'_{i+K-1}. \tag{7}$$

Next, we work backward to compute the transmission times of the remaining packets (i.e., i to $i+K-2$) in the probing window using the shortened inter-departure time Δ_D

$$d_{i+K-j} = a'_{i+K-1} - (j-1)\Delta_D. \tag{8}$$

In other words, the expected scheduling delay experienced by packets in the probing window, denoted by $\{\delta_j | j = i, i+1, \ldots, i+K-1\}$ is given by

$$\begin{aligned}\delta_j &= d_j - a'_j \\ &= \left(a'_{i+K-1} - ((i+K-1)-j)\Delta_D\right) - a'_j \\ &= \left(a'_{i+K-1} - a'_j\right) - ((i+K-1)-j)\Delta_D \\ &= ((i+K-1)-j)\Delta_A - ((i+K-1)-j)\Delta_D \\ &= ((i+K-1)-j)(\Delta_A - \Delta_D). \end{aligned} \tag{9}$$

Substituting (5) and (6) into (9) we have

$$\delta_j = \frac{L\left((i+K-1)-j\right)\left(1-f^{-1}\right)}{R_v}. \tag{10}$$

Assuming packets arrive at their expected arrival time. Then, we can compute the maximum scheduling delay from

$$\begin{aligned}\delta_{\max} &= \max_{j=i,\ldots,(i+K-1)} \left\{\delta_j\right\} \\ &= \max_{j=i,\ldots,(i+K-1)} \left\{ \frac{L\left((i+K-1)-j\right)\left(1-f^{-1}\right)}{R_v} \right\} \\ &= \frac{L\left(K-1\right)\left(1-f^{-1}\right)}{R_v}, \text{ when } j = i. \end{aligned} \tag{11}$$

Thus, the first packet (i.e., packet i) in a probing window experiences the longest scheduling delay.

In practice, the actual packet arrival time a_j may deviate from the expected arrival time a'_j. A peer simply substitute a'_j with a_j in (9) to compute the schedule delay accordingly. In case the packet arrives so late such that $\delta_j < 0$, then it will be transmitted immediately. In this case, the probing data rate may be affected (see Section IV-C).

Scheduling delay is extra delay introduced by the bandwidth probing mechanism to the end-to-end data delivery delay. From (11) we can see that it is proportional to the probing window size K and the inverse of the probing factor f. Configuration of these two parameters enables the designer to tradeoff between probing accuracy, probing bandwidth, and scheduling delay.

Specifically, increasing K will lead to longer probing window and thus provides more accurate measurement of the achievable bandwidth, at the expense of longer scheduling delay and vice versa. The probing factor f on the other hand, determines the maximum achievable bandwidth that can be measured. Thus, larger value of f will allow more bandwidth to be discovered, again at the expense of longer scheduling delay.

The in-band bandwidth probing tool is independent of the ALM protocol and thus these two parameters enables the probing tool to be optimized for the specific ALM protocol. As a rule of thumb, we will first select K to ensure robust bandwidth measurement accuracy and then determine the probing factor f either statically (e.g., subject to delay and buffer constraints—to be discussed in the next section) or adaptively (e.g., based on bandwidth availability and demand).

B. Scheduling Constraints

In practice, the scheduler is subject to delay and buffer constraints set by the application. Specifically, let B_p be the size of the pre-fetch buffer allocated to absorb the scheduling

delay. The pre-fetch buffer will be filled with video data before playback begins and thus can absorb the extra scheduling delay introduced by the bandwidth probing mechanism.

To prevent playback starvation, there must be sufficient video data in the pre-fetch buffer to sustain playback during bandwidth probing. Let H be the maximum depth of the overlay network, i.e., a packet will be forwarded by at most $(H-1)$ peers (including the source) before reaching the receiver, then we have the following constraint on the scheduling delay:

$$\delta_{\max}(H-1) \leq B_p/R_v. \tag{12}$$

Substituting (11) into (12) and rearranging terms we have

$$f \leq \left(1 - \frac{B_P}{L(K-1)(H-1)}\right)^{-1}. \tag{13}$$

Thus, given the delay constraint and the probing window size K we can determine the maximum probing factor f that can be used without causing video playback interruptions.

C. Cascaded Bandwidth Probing

The depth of the overlay network H is proportional to the size of the ALM population. Thus, for very large ALM networks, the accumulated scheduling delay as derived in (12) could lead to the need for large pre-fetch buffer and consequently long startup delay. For example, assume video packet size $L = 10$ kB, video data rate $R_v = 800$ kb/s, probing factor $f = 2$, and probing cycle $K = 30$ packets, then from (11) the computed maximum scheduling delay $d_{\max}$ will be equal to 1.45 s. For a large ALM network with a depth of $H = 6$, the worst-case delay will reach 7.25 s. Coupled with the buffer needed to absorb normal packet delay variations, the total pre-fetch buffer needed will exceed 7.25 s.

Nevertheless, the above scheduling delay is based on the worst-case scenario only. Consider an ALM network of depth H. Assume bandwidth probing is performed periodically with a cycle n times the duration of the probing window. Then, the probability for a data packet to arrive within the probing window is equal to $1/n$. Thus, the probability of a data packet to join the probing window in H-1 consecutive hops, denoted by P_{H-1}, is given by

$$P_{H-1} = \frac{1}{n^{H-1}}. \tag{14}$$

For example, with $n = 30$ and $H = 6$, the probability is merely 0.00000004 and so is not significant in practice.

A second, more subtle problem with cascaded probing is that it may negatively affect probing accuracy, leading to underestimated bandwidth. To see why, recall that the delay to be added to a probing packet δ_i is computed based on the expected packet arrival time [see (9)]. If a packet arrives so late such that the computed $\delta_i < 0$, then the resultant probing data rate may become lower than that specified by the probing factor f. This can occur whenever a probing packet was previously delayed by another probing window upstream.

To estimate the significance of this problem, we compute below the probability of a packet participating in *more than one* probing windows along the delivery path from the source to the destination. Assuming the overlay tree is a binary and *balanced* tree with depth H. Then, the number of peers at level l is 2^l. Recall that the probability for a data packet to arrive within the probing window is equal to $1/n$. Then, when a packet arrives at a peer at tree level l, the probability of it having participated in more than one probing window is given by

$$P_{>1,l} = \sum_{m=2}^{l} \binom{m}{l} \left(\frac{1}{n}\right)^m \left(1 - \frac{1}{n}\right)^{l-m}. \tag{15}$$

In a balanced binary overlay tree with H levels, the proportion of peers at level l, denoted by r_l, is given by

$$\rho_l = \frac{2^l}{2^H - 2}. \tag{16}$$

Here, $2^H - 2$ is the total number of peers, excluding the source peer, in the balanced overlay tree. Thus, the average probability of cascaded probing across all peers in the overlay tree can then be computed from

$$P_{>1} = E\left[P_{>1,l}|\rho_l\right] = \sum_{l=2}^{H-1} P_{>1,l}\rho_l. \tag{17}$$

For example, with $n = 30$ and $H = 6$, this expected probability is equal to 0.0074.

Another side effect of probing is increased delivery delay. Specifically, incoming packets participating in a probing window are delayed according to (10) in order to raise the outgoing data rate. Assuming it is equally probable for a packet to arrive at any time during a probing window. Then, the scheduling delay, denoted by the random variable d, for a randomly arriving packet will be uniformly distributed between 0 and $LR_v^{-1}(K-1)(1-f^{-1})$.

If a packet participates in m probing cycles end-to-end, then its accumulated scheduling delay, denoted by $\delta^{(m)}$, can be computed from the m-times auto-convolution of δ

$$\delta^{(m)} = \underbrace{\delta * \cdots * \delta}_{m}. \tag{18}$$

For a packet destined to a peer at tree level l, the probability for it to participate in m probing windows is

$$P_{m,l} = \binom{m}{l} \left(\frac{1}{n}\right)^m \left(1 - \frac{1}{n}\right)^{l-m}. \tag{19}$$

Assuming a balanced binary tree, then the scheduling delay distribution can be computed from taking expectation over peers at all levels of the overlay tree, that is

$$P = \sum_{m=1}^{H-1} \left[\sum_{l=m}^{H-1} \left(\frac{2^l}{2^H - 2}\right) P_{m,l}\right] \delta^m. \tag{20}$$

D. Model Verification

The mathematical model in Section IV-C employed two assumptions, namely the overlay tree is balanced and the individual scheduling delay is uniformly distributed. We investigate below the impact of these two assumptions by relaxing them in a discrete-event simulator.

Fig. 3 compares the probability distribution of cascaded probing vs. n—the ratio between duration of probing cycle and

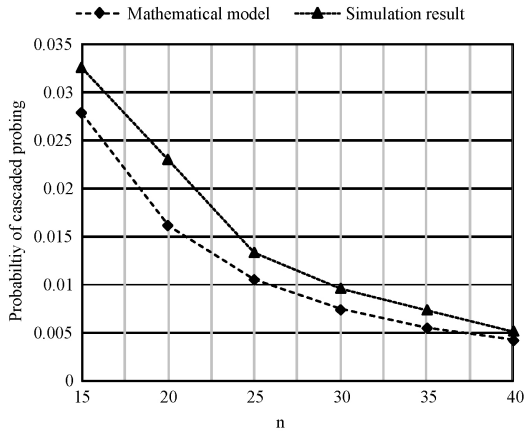

Fig. 3. Probability of cascaded probing vs. probing window to probing cycle ratio n.

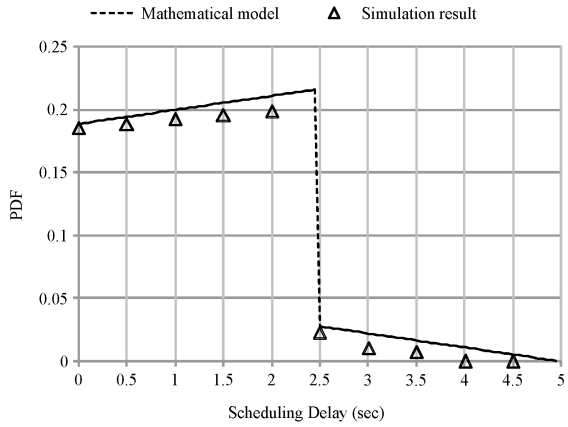

Fig. 4. Probability distribution of scheduling delay incurred in bandwidth probing.

duration of a probing window. As expected, cascaded probing can be reduced by increasing n as the probing window will be spaced temporally farther apart. The tradeoff is slower reaction to path bandwidth variations. For example, with $n = 40$ and $K = 30$, a probe will be initiated every 120 s and in this case the probability of cascaded probing is 0.0042.

Compare to the numerical results computed from the mathematical model, the simulated cascaded probing probability follows the same trend but at slightly higher values. This is because in simulation the constructed overlays are not necessarily balanced tree—this increases the probability of cascaded probing as the average overlay tree depth will be larger, resulting in more peers with larger depths.

Next, we simulated the actual scheduling delay with $n = 30$, $H = 6$, $K = 30$. The results in Fig. 4 confirm that the mathematical model closely approximate the simulation results. Due to the very small probability of cascaded probing, the scheduling delay is nearly uniformly distributed from 0 to 2.5 s, beyond which the probability is insignificant.

V. Adaptive Multioverlay ALM

The in-band bandwidth probing tool described in Section IV can generally be incorporated into any ALM protocols that employ multiple overlays for data distribution. To facilitate evaluation and comparison of path selection using achievable bandwidth vs. RTT, we developed a reference multioverlay ALM protocol partly based on existing designs and introduced a new adaptive mechanism to make use of achievable-bandwidth/RTT information to refine the overlay topology at runtime.

A. Overlay Construction

In a multioverlay ALM network, the source splits the original data stream into N sub-streams with each sub-stream to be delivered over one of the N overlays. Specifically, the original video data stream is divided into fixed-size packets and each packet is assigned a sequence number to represent its playback order in the stream. Packet i in the original data stream will be delivered to overlay i mod N. Assuming the video data stream is constant bit-rate encoded at a video bit-rate of NR_v b/s then each sub-stream will carry a data stream with rate R_v b/s.

Overlays are constructed independently of each other. There are many existing overlay construction protocols in the literature and for the purpose of this paper we adopted a RTT-based overlay construction method similar to [11], [17], and [19]. Specifically, a designated rendezvous node keeps track of the most recently joined peers, say $\{p_i | i = 0, 1, 2 \ldots M\}$ where M is a system-wide parameter. When the rendezvous node receives a join request from a joining peer, it responds with a random subset of $\{p_i | i = 0, 1, 2 \ldots M\}$. The joining peer then selects from the subset the N peers with the smallest RTT subject to satisfying the peers' outbound degree limit. This method reduces the load of the rendezvous peer and also promotes load balance across existing peers in the ALM network. It is worth noting that the proposed bandwidth probing mechanism is not coupled with the way the overlays are initially constructed and thus can be applied to any overlay construction methods to refine the overlay topologies.

In each overlay video data are delivered from peer to peer using a congestion-aware transport protocol such as TCP or TFRC. In our simulation implementation, we employed the widely-used TCP as transport as it is congestion-aware and is compatible with firewalls—an important feature in an ALM network. As the transport is congestion-aware it could block the sender from sending data in case network bandwidth is insufficient. In that case, data will accumulate inside a peer's forwarding buffer (one for each children) until the buffer is full, in which case the oldest data packet in the buffer will be discarded to make room for the arriving data packet. Thus, although the transport protocol guarantees no data loss, some data may still be discarded due to buffer overflow in the forwarding peers. These losses reflect the lack of bandwidth in distributing the data to the peers at the prescribed data rate.

B. Bandwidth Probing

After the overlay construction phase all peers in the overlay network will begin the measurement of achievable bandwidth periodically using the in-band bandwidth probing tool as described in Section IV. The system uses a fixed-size probing

window and adapts the probing factor f using an *addictive-increase-abrupt-decrease* algorithm resembling the additive increase multiplicative decrease (AIMD) algorithm in TCPs congestion control mechanism. Specifically, each peer maintains its own probing factor which begins with $f = 1$. It will increase f by σ in each probing cycle until it reaches the maximum value as dictated by the delay constraint [see (13)]. At any time if the measured achievable bandwidth, denoted by B, becomes lower than $(f - \sigma)R_v$, then the probing factor will be reset to

$$f = \lfloor B/\sigma \rfloor \sigma + \sigma. \tag{21}$$

This step differs from TCPs AIMD algorithm as unlike TCP we do know the now lower bandwidth B which we can use to reset the point of adaptation directly.

This adaptation algorithm is designed to incrementally probe for additional unused achievable bandwidth. The parameter σ controls the aggressiveness of the bandwidth probing mechanism. Too small a value will increase the time to discover unused bandwidth while too large a value may cause the probing packets to experience longer delay due to queueing time inside the sending peer's transmission buffer. In our experiments (see Section VI), we found that a step size of $\sigma = 0.1$ works well across a wide range of system parameters.

It is worth noting that the above is only one way to make use of the in-band bandwidth probe. In particular, the presented algorithm performs adaptation locally without incorporating other information such as the bandwidth availability of other peers, the current bandwidth demand, load balance across different peers, path delays, path bandwidth stability, and so on. In addition, the above algorithm only probes for unused bandwidth in existing paths (N paths per peer in a N-overlay ALM network). More sophisticated ALM protocols could also explore new paths by reconfiguring the topology within an individual overlay. More research is thus warranted to further explore the design of the adaptation algorithm for bandwidth probing in ALM networks.

C. Overlay Adaptation

Each overlay in the ALM session adapts to network congestions independently of each other. The principle of the adaptation mechanism is to divert part of the data flow from the congested path to another path with unused achievable bandwidth. This process consists of three steps, namely *adaptation triggering*, *data diversion*, and *path selection*.

The adaptation process is triggered by monitoring of incoming data throughput from a peer's parent. Each peer measures the data rate r_i at which data of overlay i are received from its parent, averaged over a sliding window of duration W. Let r_i' be the data rate expected to be received from the parent. If there is sufficient bandwidth then $r_i = r_i'$, otherwise $r_i < r_i'$.

To reduce unnecessary adaptation triggered by random bandwidth fluctuations a peer will select a new path only if the measured bandwidth drops beyond a given threshold defined by T as follows:

$$\frac{r_i' - r_i}{r_i'} > T. \tag{22}$$

Procedure Path Selection

Input: Original-Path

Output: Alternative-Path, Rerouted-Data

```
1.  i ← Original-Path
2.  Alternative-Path ← None
3.  Max-Bandwidth ← 0
4.  if max{d_ij | j = 1, 2…N} ≠ 0 then
5.      k ← argmax(d_ij) for j = 1,2…N
6.      R_d ← d_ik
7.  else
8.      k ← i
9.      R_d ← r_k'-r_k
10. end if
11. P ← candidate peers for overlay k
12. for each p in P do
13.     if B(p) > Max-Bandwidth then
14.         Max-Bandwidth ← B(p)
15.         Alternative-Path ← p
16.     end if
17. end for
18. Rerouted-Data ← min(R_d, Max-Bandwidth)
19. return Alternative-Path, Rerouted-Data
```

Fig. 5. Pseudo-code for the path selection algorithm.

For example, if $T = 0.1$ then the peer will trigger adaptation when the incoming data rate drops below the expected data rate by 10% or more.

Once triggered the adaptation process will find a new parent peer to divert data traffic from the congested path. Fig. 5 lists the pseudo-code for the path selection algorithm. A subtle complexity is that in addition to its normal data traffic, a path may have been previously assigned to carry diverted traffic from another overlay in a previous round of overlay adaptation. If such a path becomes congested then instead of diverting data traffic to yet another alternative path, the system will instead re-divert the diverted traffic it is currently carrying to remedy congestion. This mechanism helps reduce the topological complexities of the ALM network.

Specifically, each peer maintains a 2-D array $\{d_{i,j} | j = 1, 2, \ldots, N\}$ where $d_{i,j}$ is the proportion of data of overlay j which were received through overlay i due to traffic diversion. If max $\{d_{i,j}\, j = 1, 2 \ldots N\} = 0$, then there is no diverted traffic in the congested path so the algorithm will simply divert the excess of the normal data traffic, denoted by the data rate R_d, to another path

$$R_d = r_i' - r_i. \tag{23}$$

Otherwise if $\max\{d_{i,j} | j = 1, 2 \ldots N\} \neq 0$, then the system will attempt to re-divert the largest existing diverted traffic from overlay k instead

$$k = \arg\max_{j=1,2\ldots N} \{d_{i,j}\} \tag{24}$$

and the corresponding data rate of the to-be-re-diverted traffic is given by

$$R_d = d_{i,k}. \tag{25}$$

Next, the algorithm selects a new path to carry the diverted data traffic. First, peers who will create loop in the overlay and peers with insufficient unused achievable bandwidth are eliminated from the set of candidate peers P. Let $B(p)$ be the unused achievable bandwidth of peer p as measured using the bandwidth probing mechanism in Section IV. Then, the system will select the peer, denoted by q, with the largest unused achievable bandwidth (lines 12–17)

$$q = \arg\max_{p \in P}(B(p)) \tag{26}$$

and the data rate of the diverted data traffic is equal to (line 18)

$$\Delta = \min(R_d, B(q)). \tag{27}$$

D. RTT-Based Path Selection

To enable comparison we describe below a RTT-based path selection procedure which employs the same adaptation triggering procedure as described in Section V-C. First, each peer periodically measures the RTT to its N parent peers using echo-reply (i.e., ping) messages. The measured RTT of parent j is smoothed according to

$$d_j^{\text{new}} = \alpha d_j^{\text{old}} + (1 - \alpha) d_j^{\text{cur}} \tag{28}$$

where d_j^{new} and d_j^{old} are the new and old smoothed RTT estimates; d_j^{cur} is the current measured RTT value, and α is a smoothing factor with a value of 0.9 as used in the similar smoothing equation in TCP [21].

Second, when adaptation is triggered the peer will select the parent peer with the smallest RTT estimate, i.e., $\arg\min_{j=1,\ldots,N} \{d_j^{\text{new}}\}$, to divert data from the congested path at a data rate as determined by (23). Note that this differs from the achievable bandwidth-based case where the diverted data rate is determined by (27). This is because RTT measurements cannot reveal the amount of unused bandwidth in the alternate path and thus the system simply diverts all excess data traffic to the alternate path.

E. Topology-Adaptation-Induced Data Loss

In our experiments, we observed substantial data loss during some of the overlay adaptations. These losses are not due to insufficient bandwidth, but are the direct consequence of data delivery sequence differences between the old and the new parent peer. Specifically, peers in the ALM network receive a copy of the same data packet at different times depending on their relative location in the overlay tree, network delays, etc. Let $s_i(t)$ be the data sequence number being forwarded by peer i at time t. Consider peer i who switch from its old parent peer j to a new parent peer k at time t, then the incoming data stream from peer j will stop at $s_j(t)$. On the other hand the new parent peer k will be able to begin forwarding data to peer i starting from data sequence number $s_k(t)$. Now, if $s_k(t) > s_j(t)$ then the data between the two sequence numbers are no longer available from the new parent peer k and will appear as data loss to peer i.

TABLE III
DEFAULT SIMULATION PARAMETERS

Parameter	Value
Physical Node Count	5000
Transit–Transit Bandwidth	100 Mb/s
Transit–Stub Bandwidth	10 Mb/s
Stub–Stub Bandwidth	5 Mb/s
Simulation Time	1200 s
Simulation Trials	10
Multioverlay ALM	
Data Packet Size	10 kbytes
Data Stream Bit-Rate	800 kb/s
Forwarding Buffer	80 kbytes
Peer Count	64
Overlay Count (N)	5
Throughput Estimation Window (W)	5 s
Adaptation Triggering Threshold (T)	0.9
Bandwidth Probing Algorithm	
Maximum Probing Window (K)	30
Probing Cycle	90s

To tackle this problem which may otherwise skew the performance comparisons in Section VI we introduce a *make-before-break* mechanism where data forwarding from the old parent will not be stopped until its sequence number catches up with the new parent, e.g., peer j will keep forwarding data to peer i until time t' where $s_j(t') = s_k(t)$.

VI. SIMULATION EVALUATION

In this section, we compare performance of path selection using achievable-bandwidth, RTT, and residual bandwidth in the multioverlay ALM system described in Section V.

A. Simulation Setting

The simulator implements the ALM protocol described in Section V using the packet-level simulator NS2 [29]. The simulated network comprises 5000 nodes arranged in the transit–stub graph model where the transit–transit, transit–stub, and stub–stub link bandwidth were set to 100 Mb/s, 8 Mb/s, and 4 Mb/s, respectively. Peers in the ALM system are randomly attached to stub routers.

We introduced cross TCP traffics into the network by placing 50 FTP server/client pairs in random positions of the network. Their starting times are uniformly distributed in the first half of the simulation time and their transmission durations are 600 s.

For each simulation run, one sender is chosen at random to distribute application data packets of size 10 kB each at 800 kb/s. The rest of the peers then execute the ALM protocol to construct five overlays to distribute data throughout the ALM network to all peers.

Each peer conducts bandwidth probing with a probing window of 5 s, using up to 50 probing packets. Overlay adaptation will trigger if the measured bandwidth drops below 0.9 of the expected throughput. Table III summarizes the default system parameters used in the simulation.

B. Topology-Adaptation-Induced Data Loss

As explained in Section V-E, data loss may occur in overlay adaptation due to data delivery sequence differences between the old and the new parent peer. This topology-adaptation-induced data loss was often lumped together with congestion-induced data loss in previous works. However we found that topology-adaptation-induced loss is far from insignificant as depicted in Fig. 6(a) which tallies the three types of data losses: 1) overlay adaptation induced data loss; 2) forwarding buffer overflow induced data loss; and 3) missing playback deadline induced data loss.

In case of RTT-based ALM, topology-adaptation-induced data loss accounted for over 20% of the total data loss, and as such could skew conclusions drawn solely based on the total loss ratio. This problem can be eliminated using the make-before-break mechanism introduced in Section V-E, which can effectively reduce the topology-adaptation-induced loss to negligible levels as depicted in Fig. 6(b) and thus it was employed by default for the rest of the simulation results.

C. Data Delivery Performance

Data loss (or its complement data delivery ratio) is the primary performance metric of ALM protocols. Data loss is defined as the difference between the amounts of data expected to be received and the amount actually received, averaged over all peers in the ALM network. Note that we used TCP as the transport for data delivery and thus data are *not* lost in the network. Instead, every peer has a forwarding buffer of finite size (e.g., 80 kB) for each overlay so in case the outgoing path to the downstream peer does not have sufficient bandwidth to carry the forwarded traffic, data will accumulate inside the forwarding buffer of the sending peer until it is full—at that point the peer will discard the oldest queueing packets when new packet arrives. These discarded packets constitute the data loss measured in this section.

Fig. 7 plots the mean data loss experienced by a receiver using the achievable bandwidth, residual bandwidth, and RTT metrics, as a function of streaming bit-rate. Residual bandwidth estimation is done by integrating the network simulator simulation code from pathChirp [14] into our reference ALM protocol. As expected the loss ratio increases with higher streaming bit-rates. In all cases, achievable bandwidth outperforms both residual bandwidth and RTT as the metric used in the ALM network and the differences widen significantly at higher streaming bit-rates. For example, at a streaming bit-rate of 880 Kb/s, achievable bandwidth achieved a loss ratio of only 5% which is less than 1/3 that of RTT (at over 15%).

A second important performance metric for ALM protocols is data delivery latency, defined as the time it takes for data to travel from the source to the destination peer across the ALM network. Fig. 8 plots the data delivery latency averaged over all peers as a function of streaming bit-rate. The error bars represent the 95% confidence interval.

At lower streaming bit-rates, RTT and residual bandwidth achieve lower latency than achievable bandwidth. In this region, congestion-induced delay is relatively small and the scheduling delay due to bandwidth probing dominates the end-to-end delay.

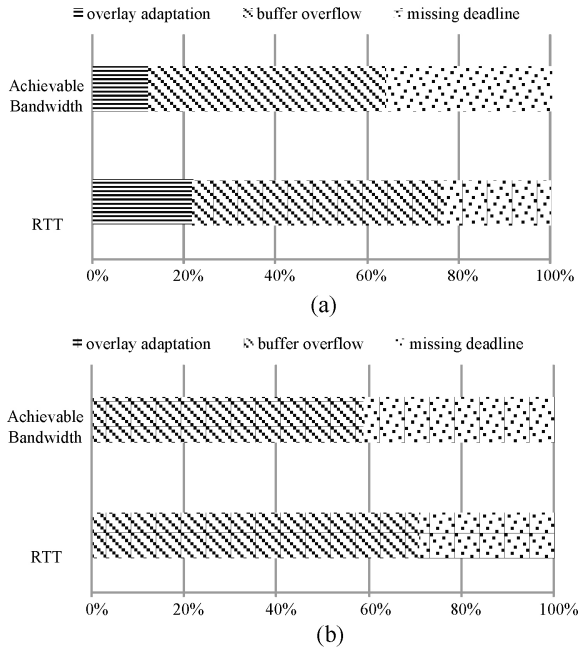

Fig. 6. Compositions of data losses. (a) Without meet-before-break mechanism. (b) With meet-before-break mechanism.

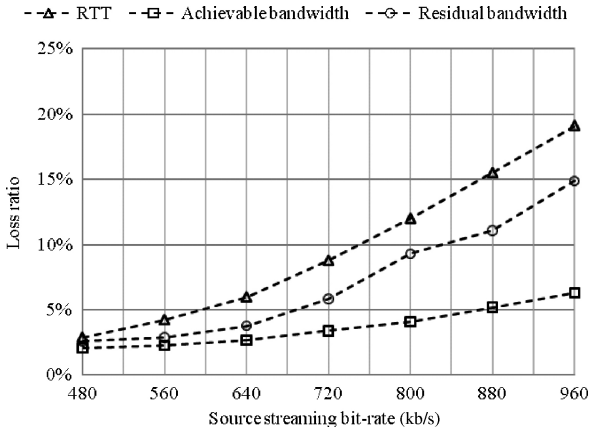

Fig. 7. Comparison of data loss ratio vs. streaming bit-rate.

By contrast, the latency differences narrow considerably at higher streaming bit-rates as congestion-induced delays in residual bandwidth and RTT-based ALM become significant. At a streaming bit-rate of 880 kb/s the latencies for RTT and achievable bandwidth are 2.45 s and 2.68 s, respectively, well within practical limits for video streaming applications.

D. Video Streaming Performance

To evaluate video streaming performance, we also need to model the video player operation at the peer. Most video player adopts a buffer-and-play model where a video buffer will be used to pre-fetch video data up to a certain threshold before playback begins. The buffered data thus allow the video player to absorb small variations in video packet arrival times. In case of insufficient bandwidth, the buffered data will eventually be depleted and the video player will suspend playback until the

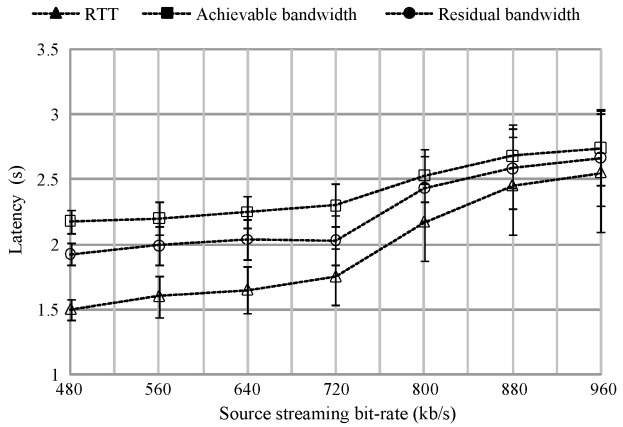

Fig. 8. Comparison of data latency vs. streaming bit-rate.

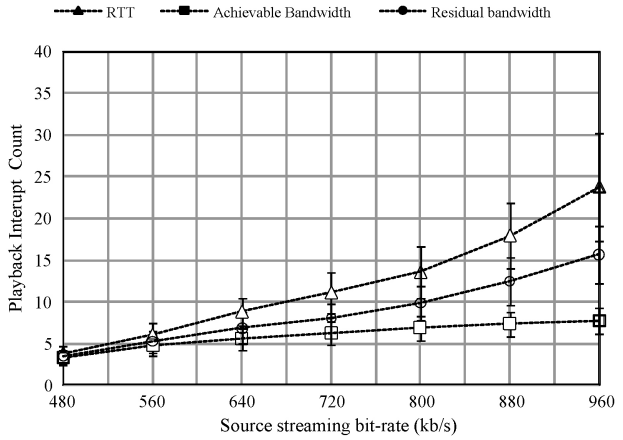

Fig. 9. Number of video playback interruptions vs. video streaming bit-rate.

pre-fetch buffer is refilled with video data.

Using this buffer-and-play model, we evaluate two video streaming performance metrics. The first one is the number of playback interruptions during the video streaming session and the second one is the rebuffering time, defined as the total time at which playback is suspended when the pre-fetch buffer is being refilled. Figs. 9 and 10 plot these two performance metrics, averaged over all peers in the ALM network, against video streaming data rate from 480 kb/s to 960 kb/s.

Consistent with the data delivery performance results the video streaming performance of achievable bandwidth outperforms residual bandwidth, which in turn outperforms RTT across all the simulated video data rates, despite the fact that additional scheduling delay is incurred by the in-band bandwidth probe. Moreover, the performance gap widens significantly at higher video data rates, suggesting that the achievable bandwidth metric is significantly more efficient in discovering and utilizing bandwidth in the network.

E. Performance Variation Across Peers

Next, we study the performance variations across peers in the ALM network, using the methodology proposed by Chu *et al.* [8]. For each simulation run receivers are sorted in ascending order according to their loss/latency performance where the worst performing receiver has rank 1. After a set of

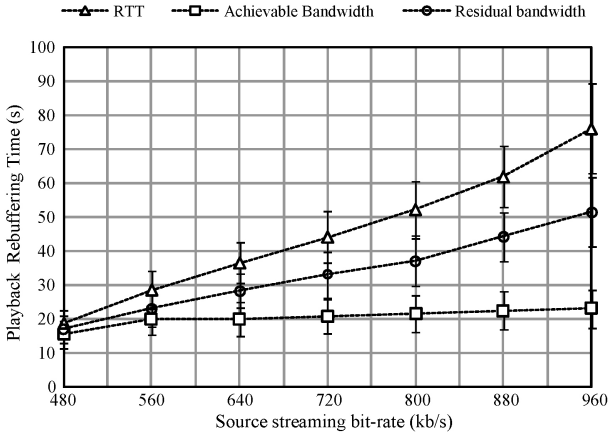

Fig. 10. Total video rebuffering time vs. video streaming bit-rate.

simulation runs, the mean performance for the rank r receiver is computed.

Fig. 11 plots the mean data loss and Fig. 12 plots the mean data delivery latency as a function of receiver rank, with the error bar marking the 95% confidence interval. It can be seen that generally achievable bandwidth metric results in fairer performance among all the peers as the performance deviation between high rank peer and low rank peer is smaller.

A second observation is that achievable bandwidth results in more consistent data loss performance as indicated by the small standard deviation.

To further quantify the fairness among peers in the ALM network, we compute the fairness index introduced by Jain *et al.* [30]

$$f = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}. \tag{29}$$

Here, x_i is the rank number and n is the number of peers.

For data loss, the fairness indexes of achievable bandwidth, residual bandwidth, and RTT are 0.998, 0.992, and 0.987, respectively. This indicates slight advantage for the former metric. For data delivery latency, the difference widens substantially, with fairness indices of 0.852, 0.834, and 0.797 for achievable bandwidth, residual bandwidth, and RTT, respectively. This indicates that achievable bandwidth can result in significantly more even latency performance across all peers in the ALM network.

F. Overlay Topology Convergence

An adaptation protocol is designed to adapt to changes in the network. However a subtle complication is that the ALM network does not run independently of the physical network—it is in fact an active part of the physical network. Thus, when the overlay topology adapts, data traffic will be rerouted through different paths and this by definition changes the network condition compared to the case *before* the adaptation takes place. This change may trigger another round of adaptation if the metric and threshold for adaptation are met, leading to undesirable oscillations.

To investigate this effect, we recorded the accumulative number of overlay adaptations as a function of the simulation

time in Fig. 13. All three metrics require substantial number of topology adaptations (over 20) during the initial 2 min. This is because the initial topology may be far from optimal and thus an intense period of topology adaptations is required to arrive at a better-performing topology.

Beyond the initial 2 min, achievable bandwidth triggered a small number of additional topology adaptations (e.g., a total of 35 until $t = 20$ min). These additional adaptations are triggered by changes in the randomly generated background competing traffic. By contrast, RTT triggered significantly more adaptations. Analyzing the simulation traces reveal that RTT suffered from route flapping over the entire simulation period—precisely the topology oscillation problem described previously. This is caused by the use of an indirect metric (RTT) to estimate the required resource availability (bandwidth).

Worst still, the RTT metric is affected by re-routed data traffic. A new path selected by overlay adaptation will carry additional data traffic re-routed from the old path. Consequently the new path's utilization will increase, leading to increased queuing delay and thus the RTT itself. The old path, now being relieved of the data traffic, may now become underutilized, leading to reduced RTT. Thus, in the next round of measurement the adaptation algorithm will find that the old path is now better performing RTT-wise than the new path, and so may trigger the re-routing of the data traffic from the new path back to the old path, and so on. This oscillation is undesirable as frequent topology adaptation will induce more data loss and reduce the stability of the ALM network.

G. Topology Convergence With Multiple ALM Networks

In addition to interaction with competing traffics, multiple ALM network sessions will also interact with each other and thus may affect their overlay topology convergence. To investigate the impact we conducted a simulation with 1, 3, and 5 concurrent ALM network sessions with disjoint peer groups, i.e., a peer joins no more than one ALM session. The physical network is the same as in previous simulations while the number of peers were increased by a factor of n, where n is the number of concurrent ALM sessions.

Fig. 14 compares the number of topology changes over time for the three simulation settings. We observe that more concurrent ALM sessions indeed will result in slightly more topology changes, but the increase is relatively small. More importantly, increasing the number of sessions from 3 to 5 did not result in proportional increases in topology changes. This suggests that the ALM protocol's topology convergence is not sensitive to concurrently running ALM sessions.

H. Impact of Overlay Adaptation Triggering Threshold

The sensitivity of overlay adaptation is controlled by the triggering threshold T (see Section V-C). We evaluate the impact of this parameter in Fig. 15. Considering data loss performance as depicted in Fig. 15(a), we observe that the triggering threshold can consistently keep the loss ratio within $(1 - T)$. This shows that with achievable bandwidth the triggering threshold can be used to tradeoff between quality of service and overlay adaptation overheads.

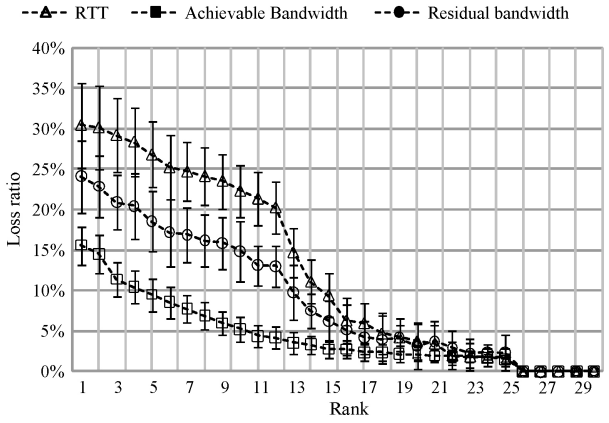

Fig. 11. Data loss ratio sorted by rank.

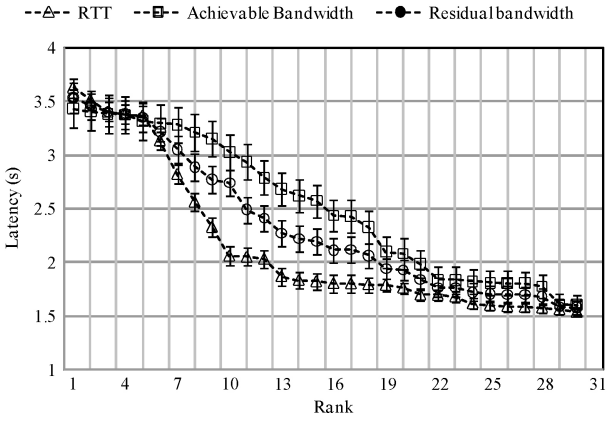

Fig. 12. Data latency sorted by rank.

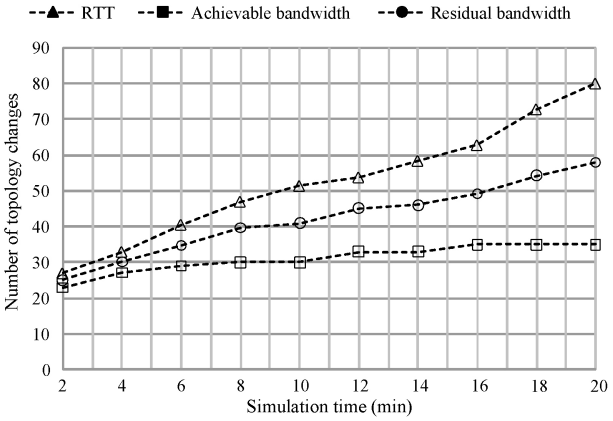

Fig. 13. Cumulative number of topology changes vs. time.

By contrast, when residual bandwidth and RTT are used the system failed to keep the loss ratio below $(1 - T)$ for $T \geq 0.9$ and $T \geq 0.95$, respectively. Worst still, the loss ratio increases when T was increased from 0.9 to 0.95. This is because when the loss ratio exceeds $(1 - T)$, the system will keep adapting its overlay topology as the triggering threshold is frequently exceeded. The frequent topology adaptations in turn led to higher level of system instability and likely result in even more data loss.

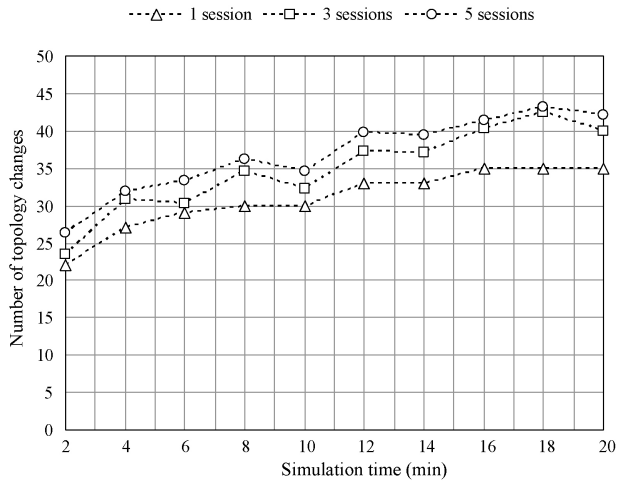

Fig. 14. Cumulative number of topology changes with multiple video sessions vs. time.

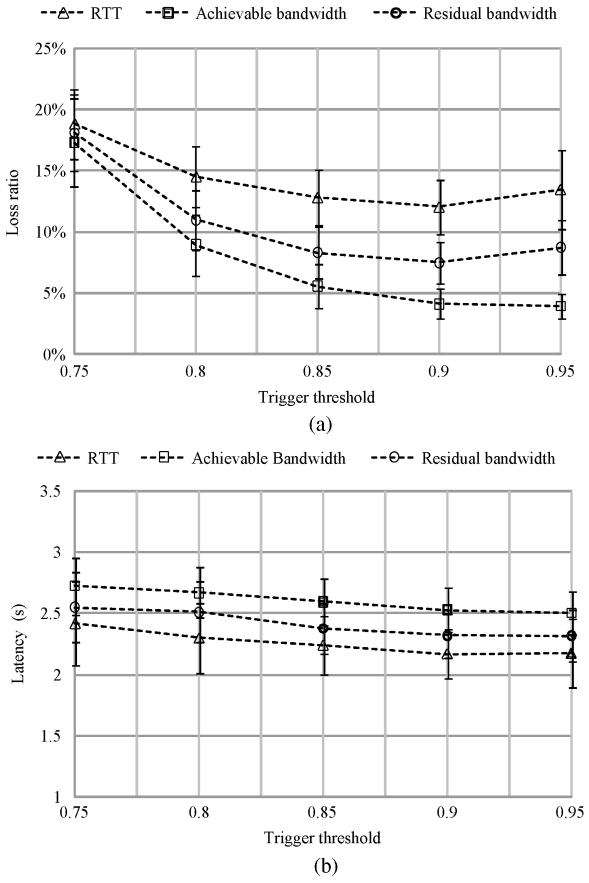

Fig. 15. Sensitivity of data delivery performance to the overlay adaptation trigger threshold. (a) Data loss ratio. (b) Data latency.

Next, we evaluate the sensitivity of data delivery latency to the triggering threshold in Fig. 15(b). The results clearly show that latency is relatively insensitive to the triggering threshold as the latency is dominated by propagation delay in the case of RTT and bandwidth probing scheduling delay in the case of achievable bandwidth.

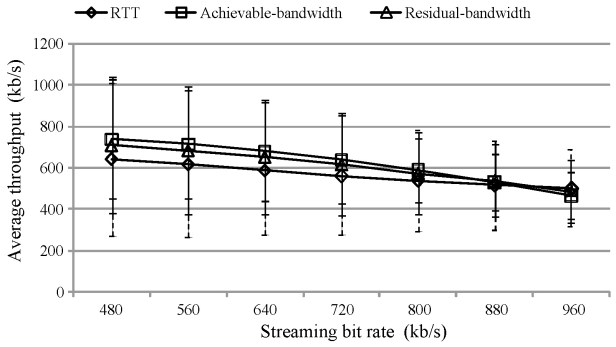

Fig. 16. Average throughput of competing cross traffic vs. streaming bit-rate.

I. Impact to Cross Traffic

In all the previous simulations, a total of 50 random FTP flows were introduced to create competing cross traffics. Intuitively we would expect the throughput of competing traffics to suffer when the overlay network achieves higher throughput, e.g., by switching from the RTT metric to the achievable bandwidth metric. Surprisingly this is not necessary the case as shown in Fig. 16 which plots the average throughput of the competing cross traffics vs. the streaming bit-rate of the ALM protocol. Except for the highest streaming bit-rate of 960 kb/s, in all other cases the cross traffics' throughput is higher when achievable bandwidth is used in place of residual bandwidth and RTT, despite the fact that more data are transported with achievable bandwidth (see Fig. 7).

This counter-intuitive result is explained by the observation that achievable bandwidth can more accurately measure the fair share of bandwidth in a target path in case there are other competing traffic flows. Thus, it will not divert more data than the new path can forward and as a result prevent inducing congestion in the new path. The excess data are then further diverted to other available paths. At the highest bit-rate the throughput of competing flows do decrease slightly compared to residual bandwidth and RTT-based metrics as their streaming throughput, at loss rates of 15%–20%, is significantly lower than the prescribed data rate (see Fig. 7).

VII. Conclusion

Despite its widespread adoption, our measurement experiments clearly show that RTT may not correlate well with bandwidth availability between peers in the Internet, thus (mis)leading ALM protocols into constructing overlays which are sub-optimal. By contrast, the proposed achievable bandwidth metric is inherently more robust as it takes into account all the factors, including link capacity, competing traffic, and protocol behavior, which affect the data throughput that can be realized along the network path. Extensive simulations from a reference multioverlay ALM implementation demonstrated that achievable bandwidth metric consistently out-performs both RTT and residual bandwidth metrics in data delivery ratio and has comparable performance in data delivery latency at high data rates. Moreover, achievable bandwidth metric can prevent overlay topology oscillations and improve fairness among peers.

One interesting discovery in this paper is the in-band bandwidth probing tool that performs bandwidth probing by rescheduling the transmission timings of outgoing packets in a peer. As it does not send extra probing packets, no bandwidth overhead will be incurred. The only tradeoff is increased data delivery delay. This tool clearly is not limited to ALM protocols and could be applied to any other applications where bandwidth estimation is needed. For example, it could be applied to adaptive video streaming where the video encoding rate could be adapted to fit the bandwidth available in the network path to the receiver (e.g., HTTP progressive download).

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the anonymous reviewers for their helpful suggestions to improve the paper.

REFERENCES

[1] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 282–297, Dec. 2003.

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

[3] 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.

[4] J. J. D. Mol, D. H. J. Epema, and H. J. Sips, "The orchard algorithm: P2P multicasting without free-riding," in *Proc. IEEE Int. Conf. Peer-to-Peer Comput.*, Sep. 2006, pp. 275–282.

[5] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang, "On maximizing tree bandwidth for topology-aware peer-to-peer streaming," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1580–1592, Dec. 2007.

[6] S. Xie, B. Li, G. Y. Keung, and X. Zhang, "Coolstreaming: Design, theory, and practice," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1661–1671, Dec. 2007.

[7] 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.

[8] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[9] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 205–217.

[10] D. Tran, K. Hua, and S. Sheu, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *Proc. IEEE INFOCOM*, vol. 2. 2003, pp. 1283–1292.

[11] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, "Overcast: Reliable multicasting with an overlay network," in *Proc. 4th USENIX OSDI Symp.*, Oct. 2000, pp. 197–212.

[12] X. Xiao, Y. Shi, B. Zhang, and Y. Gao, "OCals: A novel overlay construction approach for layered streaming," in *Proc. ICC*, May 2008, pp. 1807–1812.

[13] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proc. PAM Workshop*, Aug. 2002, pp. 14–25.

[14] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proc. PAM Workshop*, Apr. 2003.

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

[16] K. K. To and J. Y. B. Lee, "Parallel overlays for high data-rate multicast data transfer," *Comput. Netw.*, vol. 51, no. 1, pp. 31–42, 2007.

[17] V. Venkararaman, P. Francis, and J. Calandrino, "Chunkyspread: Heterogeneous unstructured end system multicast," in *Proc. IEEE ICNP*, Nov. 2006, pp. 2–11.

[18] B. Zhang, S. Jamin, and L. Zhang. "Host multicast: A framework for delivering multicast to end users," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 1366–1375.

[19] Y. Okada, M. Oguro, J. Katto, and S. Okubo, "A new approach for the construction of ALM trees using layered video coding," in *Proc. ACM Workshop Adv. P2P Multimedia Streaming*, 2005, pp. 59–68.

[20] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun Surveys Tuts.*, vol. 9, no. 3, pp. 58–74, Jul.–Sep. 2007.

[21] *TCP Congestion Control*, The Internet Society, RFC 2581, Apr. 1999.

[22] *TCP Friendly Rate Control (TFRC): Protocol Specification*, The Internet Society, RFC 3448, Jan. 2003.

[23] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, "LION: Layered overlay multicast with network coding," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 1021–1032, Oct. 2006.

[24] S. J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu, "Bandwidth-aware routing in overlay networks," in *Proc. IEEE INFOCOM*, 2008, pp. 1732–1740.

[25] M. Jain and C. Dovrolis, "Path selection using available bandwidth estimation in overlay-based video streaming," in *Proc. IFIP Netw.*, 2007, pp. 2411–2418.

[26] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proc. NOSSDAV*, 2002, pp. 127–136.

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

[28] *SpeedTest* [Online]. Available: http://www.speedtest.net/

[29] *NS2* [Online]. Available: http://www.isi.edu/nsnam/ns/

[30] R. Jain, D. M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Digital Equipment Corporation, Maynard, MA, Tech. Rep. 301, Sep. 1984.

Yang Y. Lin received the Bachelors degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2007, and the Masters degree in information engineering from the Chinese University of Hong Kong, Shatin, Hong Kong, in 2009.

From 2007 to 2009, he was a member of the Multimedia Communications Laboratory, where he participated in the research of algorithms and systems for application layer multicast video streaming. He currently works with a software company, i.e., Harmonic, Inc., Kowloon, Hong Kong, focusing on multimedia communications and video streaming systems.

Jack Y. B. Lee (M'95–SM'03) received the B.Eng. and Ph.D. degrees in information engineering from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1993 and 1997, respectively.

He is currently an Associate Professor with the Department of Information, Chinese University of Hong Kong. He and his research group have conducted extensive research in multimedia communications systems. Since 1996, they have investigated and developed three generations of large-scale video streaming systems employing parallel-server architectures, multicast streaming algorithms, and decentralized architectures. In addition to multimedia system architectures, they also investigate application and transport protocols for multimedia communications, such as adaptive streaming algorithms, and flow/congestion control algorithms for multimedia applications. He is also active in technology transfer. Several technologies developed by his research group had been commercialized, resulting in 7 U.S. patents and three spin-off companies.