Collaborative Measurements of Upload Speeds in P2P Systems

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Collaborative Measurements of Upload Speeds in P2P Systems

John R. Douceur, James Mickens, Thomas Moscibroda
Distributed Systems Research Group,
Microsoft Research,
Redmond, WA
{johndo, mickens, moscitho}@microsoft.com

Debmalya Panigrahi*
Computer Science and Aritificial Intelligence Laboratory,
Massachusetts Institute of Technology,
Cambridge, MA
debmalya@mit.edu

***Abstract*—In this paper, we study the theory of collaborative upload bandwidth measurement in peer-to-peer environments. A host can use a bandwidth estimation probe to determine the bandwidth between itself and any other host in the system. The problem is that the result of such a measurement may not necessarily be the sender's upload bandwidth, since the most bandwidth restricted link on the path could also be the receiver's download bandwidth. In this paper, we formally define the *bandwidth determination problem* and devise efficient distributed algorithms. We consider two models, the free-departure and no-departure model, respectively, depending on whether hosts keep participating in the algorithm even after their bandwidth has been determined. We present lower bounds on the time-complexity of any collaborative bandwidth measurement algorithm in both models. We then show how, for realistic bandwidth distributions, the lower bounds can be overcome. Specifically, we present $O(1)$ and $O(\log \log n)$-time algorithms for the two models. We corroborate these theoretical findings with practical measurements on a implementation on PlanetLab.**

I. Introduction

Determining the bandwidth information between two hosts in the Internet is a well-studied problem in the networking literature and there are numerous existing tools available for measuring the available bandwidth between two hosts, e.g., [14], [17], [22], [25], [24], [28], [20]. In many peer-to-peer network scenarios, however, we are interested in quickly determining the available upload bandwidth of *a large number of hosts* simultaneously.

In this paper, we study the question of how we can efficiently utilize point-to-point bandwidth probes to determine the upload bandwidth of a set of hosts. Formally, we define and study the *peer-to-peer bandwidth determination problem*: We consider a set of hosts, each having a defined but initially unknown upload bandwidth u_i and download bandwidth d_i. We divide time into rounds; during each round, any pair of nodes h_i and h_j can perform a *bidirectional bandwidth probe*, which reveals both the minimum of u_i and d_j as well as the minimum of u_j and d_i. Our goal is to use these probes in a coordinated and distributed manner to efficiently determine every node's upload bandwidth. Specifically, we seek to minimize the *time complexity* of bandwidth estimation, in terms of the number of rounds required to determine the upload bandwidth of all nodes.

The need to solve this particular problem arose in the context of a P2P game system called Donnybrook [2], which we designed and built to enable high-speed online games to scale up to large numbers of simultaneous players. Donnybrook uses several techniques to overcome the upload bandwidth limitations of hosts in this context, one of which is to use higher-bandwidth nodes to relay and fan-out state updates from lower-bandwidth nodes. However, for this technique to work correctly, the (upload) bandwidths of all nodes in the game must be determined quickly before the start of each game. Although P2P games provided the immediate impetus for our addressing the bandwidth-determination problem, the problem is relevant in other contexts as well. For instance, P2P streaming systems can improve their delivery rates by adjusting the selection of neighbors according to peer bandwidth [13], [18], and multicast systems can benefit by changing their interconnection topology in a bandwidth-sensitive fashion [3], [4]. Generally, there is extensive prior work which assumes that the peers' upload bandwidth constraints are known a-priori, yet efficiently determining these constraints is a previously unsolved problem.

The challenge in determining bandwidths in a P2P system is that a pair-wise bandwidth estimation probe between two hosts does not necessarily reveal the sending host's maximum upload speed. Instead, the effective bandwidth measured by the probe is constrained by the most restrictive link on the path from the host to the remote endpoint. Since access links are typically much more restrictive than the Internet routing core [9], the result of the bandwidth measurement is usually the lesser of the sending host's upload speed and the remote endpoint's download speed. Thus, to confidently measure its upload speed, a host must select a remote endpoint with a greater download bandwidth than its own (as yet undetermined) upload bandwidth. However, it has no a priori way of finding such a host.

In this paper, we theoretically study algorithms that use bandwidth probes to quickly determine all upload bandwidths in the system. We start by showing a straightforward way to orchestrate the use of these probes in a fully distributed (and deterministic) algorithm with time-complexity $O(\log n)$.

*Part of this work was done while the author was an intern at Microsoft Research, Redmond, WA.

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Improving this bound turns out to be challenging. In fact, we prove that the algorithm is optimal by deriving a corresponding lower bound of $\Omega(\log n)$ on the time-complexity of any deterministic algorithm. We also prove an $\Omega(\log \log n)$-lower bound on any randomized algorithm.

In practice, the bandwidth distributions encountered in real systems often exhibit nicer properties than the ones used to construct the above lower bounds. To characterize the impact of the bandwidth distribution on the time-complexity of algorithms, we define *bandwidth distribution slack* χ, a measure that captures the inherent "harshness" of the system's bandwidth distribution. In this paper, we show that even under mild assumptions about this slack, substantially faster, (sublogarithmic) algorithms can be devised. Specifically, we present two algorithms, achieving a time-complexity of $O(\log \log n)$ and $O(1)$, respectively. The reason for the difference in time-complexity is that the $O(\log \log n)$-algorithm operates under the more restrictive *free-departure model*. In this model, a node can leave the system as soon as its bandwidth is determined. On the other hand, the constant-time algorithm operates under the less restrictive *no-departure model* in which nodes remain in the system until *all* nodes know their bandwidth, and hence, nodes whose upload bandwidth has already been determined may be utilized to figure out the bandwidth of yet undetermined nodes.

In addition to our theoretical findings, we evaluate our practical implementation of the algorithms using simulations on empirically-derived bandwidth distributions. We also validate these simulations with a real-world wide-area deployment using PlanetLab [27].

II. Background and Related Work

The *capacity* of an individual link in the Internet is the maximum rate at which it can transmit packets. At any given time, the link's *available bandwidth* is its unused residual capacity. An Internet path connecting two hosts consists of multiple routers and physical links. The capacity of an end-to-end path is the minimum capacity among its constituent links. The path's available bandwidth is defined in a similar way. In this paper, we focus on determining available bandwidths in ad-hoc distributed groups. We defer a discussion of capacity estimation to other work [7], [12], [19].

There are many tools that estimate the available bandwidth along an end-to-end path [14], [17], [22], [25], [24], [28], [15], [20]. These tools use different techniques. In the *packet rate method* (e.g., PTR [14], pathload [17], TOPP [22], and pathchirp [25]), for instance, the source generates traffic at a variety of speeds, and the receiver reports its observed download rates. Above a certain transmission speed, the bottleneck link becomes saturated, causing the receiving rate to fall below the sending rate. The available bandwidth is the lowest transmission rate which triggers this congestion. The issue with this and other techniques is that they do not report *which* link is the bottleneck. It is therefore impossible for a host A to know for sure whether the measured result is indeed its own upload bandwidth. It could as well be the remote host's download bandwidth. Hence, a collaborative effort is required in order for nodes to reliably determine their upload bandwidth.

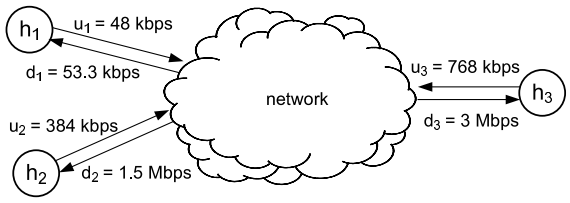


Fig. 1. Bandwidth probe $h_1 \leftrightarrow h_3$ reveals only h_1's upload bandwidth, since $P(h_3 \rightarrow h_1) = d_1$. In contrast, $h_2 \leftrightarrow h_3$ actually reveals both uploads (i.e., $P(h_2 \rightarrow h_3) = u_2$ and $P(h_3 \rightarrow h_2) = u_3$), but h_3 cannot be sure that the measurement $P(h_3 \rightarrow h_2)$ is indeed its own upload.

We are not aware of any theoretical work on this problem, and none of the classic distributed computing problems seems to have a similar combinatorial structure. Moreover, the hub-and-spoke network model is different from the models most typically studied in distributed computing. In contrast to work on complete communication graphs (e.g. [21]), only access links have weights attached to them, the Internet's core is assumed to have infinite bandwidth. Other algorithmic work in the hub-and-spoke network model include for example, bandwidth maximization in [10].

In our recent work on ThunderDome, we build on the theoretical ideas and underpinnings derived in this work to develop a practical system for the P2P bandwidth estimation problem [11]. Specifically, ThunderDome provides a practical solution for bandwidth estimation in P2P systems that also takes care of several issues that are abstracted from the theoretical models we study in this paper, most notably, how to deal with bandwidth estimation errors. Besides Thunderdome, another related practical system is BRoute [15], which however uses dedicated network infrastructure—hosts discover their edge links by issuing traceroutes to well-known landmark nodes. BRoute also requires BGP data to determine which edge segments connect end hosts. In our problem domain, groups are ad-hoc, meaning that dedicated infrastructure is unlikely to exist.

III. Model & Problem Definition

The network consists of n hosts (nodes) $H = \{h_1, \ldots, h_n\}$. Each node h_i is connected to the Internet via an access link of download bandwidth d_i and upload bandwidth u_i. We assume *directional asymmetry*, i.e., that for each host, its download bandwidth is at least as large as its upload bandwidth, i.e., $d_i \geq u_i$. This is true for virtually all existing network access technologies (e.g., dial-up modems [16], cable modems [6], and ADSL [1]). Furthermore, since in practice, the bandwidth bottlenecks between two end-hosts occur on the last-mile access link, we model the Internet as a hub-and-spoke network with the core of the Internet having unlimited bandwidth [9]. Finally, we assume that bandwidths are stationary during the execution of the algorithms, which, given the relatively short duration of all of our algorithms is usually valid.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Bandwidth Probe: A bandwidth probe $h_i \rightarrow h_j$ is a (directed) pairing of two hosts h_i and h_j, such that h_i transmits data to h_j. The result of a bandwidth probe, denoted by $P(h_i \rightarrow h_j)$, is the bandwidth at which data is transmitted from h_i to h_j. It is the minimum of h_i's upload and h_j's download, i.e.,

$$P(h_i \rightarrow h_j) = \min\{u_i, d_j\}.$$

Bandwidth Determination Problem: We assume that time is divided into *rounds*, where a round is the unit of time required to conduct one bandwidth probe. In each round, every host can participate in two bandwidth probes: one in the outgoing direction, and one in the incoming direction. Thus, up to n bandwidth probes can be done in parallel in one round. The *upload bandwidth determination problem* asks for a schedule of bandwidth probes such that all upload bandwidths can be determined after the schedule is executed. (For technical reasons, we do not require the algorithm to find the upload bandwidth of the node with the maximum upload bandwidth. This is necessary because there might exist a host h_i for which $u_i > d_j, \forall j \neq i$. In this case, this host's upload bandwidth cannot be determined using bandwidth probes.) The number of rounds required by the algorithm is called its *time complexity*—our goal is to design an algorithm with minimum time complexity for this problem.

No-Departure Model vs. Free-Departure Model: An important consideration is whether each node is expected to remain in the system until the upload bandwidths of *all* nodes have been determined, or whether each node has the freedom to depart from the system after *its own* bandwidth is determined. We call these two models the *no-departure model* and the *free-departure model*, respectively. Thus, in a particular round, an algorithm in the no-departure model is allowed to use nodes whose bandwidth has been determined in a previous round, while an algorithm in the free-departure model is not allowed to use such nodes.

Clearly, any algorithm for the free-departure model is also a valid algorithm for the no-departure model. Conversely, a lower bound on the performance of algorithms in the no-departure model holds for algorithms in the free-departure model as well.

Bandwidth Distribution Slack: Notice that if *every host's* download bandwidth was higher than *every host's* upload bandwidth ($d_i \geq u_j, \forall i, j$), the problem would be trivial: a single round would be sufficient. On the other hand, we will show later that if each host's download capacity exactly equals its upload capacity, then the problem can be difficult to solve—all our lower bounds will use bandwidth distributions with this property. In practice, bandwidth distributions are typically between these two extremes. To characterize bandwidth distributions, we define *bandwidth distribution slack* χ as follows.

Definition 1 (Bandwidth Distribution Slack)**.** *Consider nodes $h_1, \ldots, h_n$ in non-decreasing order of their upload bandwidth. The bandwidth distribution slack χ of a bandwidth distribution is the smallest fraction $\frac{1}{n} \leq \chi \leq 1$ such that for every pair of nodes h_i and h_j with $j \geq \chi i$, $d_j > u_i$.*

Clearly, the two extreme situations described above correspond to value of χ of $1/n$ and 1 respectively. We will use the fact that the bandwidth distribution slack of a typical network is between the extreme values of $1/n$ and 1 in designing randomized algorithms in section V.

IV. Deterministic Algorithms and Lower Bound

In this section, we will resolve (upto a small constant) the deterministic time complexity of the upload bandwidth determination problem. Before presenting our results, let us introduce the concept of a pairwise bandwidth probe.

Pairwise bandwidth probe: A *pairwise bandwidth probe* $h_i \leftrightarrow h_j$ of two hosts h_i and h_j is said to be executed when each of the two hosts executes a bandwidth probe to the other in the same round. The result of a bandwidth probe $h_i \leftrightarrow h_j$ are the two measurements $P(h_i \rightarrow h_j) = \min\{u_i, d_j\}$ and $P(h_j \rightarrow h_i) = \min\{u_j, d_i\}$. The following lemma claims that the minimum of these two measurements is the minimum of u_i and u_j. Then, after a pairwise bandwidth probe, the participating node with the smaller upload bandwidth gets to know its own upload bandwidth. This node is said to *lose* the bandwidth probe, while the other node is said to *win* it.

Lemma 1. *A pairwise bandwidth probe reveals the smaller of the upload bandwidths of the two nodes involved.*

Proof: Suppose h_i and h_j participate in the pairwise bandwidth probe, and let $u_i \leq u_j$, wlog. Then, $d_j \geq u_j \geq u_i$, and hence, $u_i = P(h_i \rightarrow h_j)$. On the other hand, $d_i, u_j \geq u_i$, and hence $P(h_j \rightarrow h_i) \geq u_i$. Thus, $u_i = \min(P(h_i \rightarrow h_j), P(h_j \rightarrow h_i))$, i.e. it equals the smaller of the two measured bandwidths. ■

The following algorithm uses pairwise bandwidth probes, and works for the free-departure model (and therefore the no-departure model as well).

A. Baseline Algorithm (BASE):

In each round, the algorithm creates an arbitrary pairing of all nodes which do not yet know their upload bandwidth. (If the number of nodes who do not know their upload bandwidth is odd, then one node does not participate in the pairing.) For each such pair, we perform pairwise bandwidth probes. One node in each pair (the one with the smaller upload bandwidth) gets to know its upload bandwidth and no longer participates in the subsequent rounds. Finally, a single node is left, which is guaranteed to be the node with the maximum upload bandwidth since any node whose upload bandwidth is revealed in a round must have been paired with a node with greater upload bandwidth.

It is clear that the algorithm works in the free-departure model. The following theorem analyzes the time complexity of the algorithm. (Note that $\lg$ denotes $\log$ to the base 2.)

Theorem 1. *The time complexity of the above algorithm, $BASE(n) \leq \lceil \lg n \rceil$.*

Proof: Clearly, the time complexity of the algorithm is a monotonic function of n, i.e. $BASE(n+1) \geq BASE(n)$.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Since for all $1 \leq k \leq 2^{\ell-1}$, $\lceil \lg(2^{\ell-1}+k) \rceil = \ell$, we only need to consider a case where $n = 2^\ell$ for some ℓ. In this case, the number of nodes whose upload bandwidth is unknown at the end of each round is even, and exactly half of them are revealed in the next round. Thus, all but one of the upload bandwidths are revealed in ℓ rounds. ■

B. Lower Bound:

We now give a matching (upto a constant of $\log 3$) lower bound for deterministic algorithms in the no-departure model. Recall that all lower bounds in the no-departure model also apply to the free-departure model.

Theorem 2. *For any deterministic algorithm $\mathcal{A}$ in the no-departure model, there exists a bandwidth distribution for which $\mathcal{A}$ has a time complexity of $\lceil \log_3 n \rceil$.*

Proof: Suppose $n = 3^\ell$, and therefore $\ell = \lceil \log_3 n \rceil$. Also, let the upload and download bandwidth of each node be equal, i.e. $u_i = d_i$ for each i—we call this the bandwidth of the node. We prove the theorem by induction on the value of ℓ. Clearly, the theorem holds for $\ell = 1$, since at least one round is required for 3 nodes. Now, assume that the theorem holds for $n = 3^{\ell-1}$—let the bandwidth of node h_i under the corresponding distribution be b_i. We create a new distribution for $n = 3^\ell$, where two-thirds of the nodes have bandwidth 1, and any node h'_i in the remaining one-third is matched to a unique node h_i in the inductive set of $3^{\ell-1}$ nodes and has bandwidth $b_i + 2$. Let us call the first category of nodes *easy* and the remaining nodes *hard*. Thus, the bandwidth of any hard node is strictly greater than the bandwidth of any easy node. The actual category that a node belongs to depends on the algorithm $\mathcal{A}$ and is described below.

Define a graph for the first round of the algorithm, where the set of vertices is the set of nodes, and the set of edges is as described below. In the first round, algorithm $\mathcal{A}$ makes a set of bandwidth probes. Let us represent a probe from node h_i to node h_j by a directed edge from h_i to h_j. Since each vertex has both in-degree and out-degree of at most 1, the directed graph formed is a collection of directed paths and directed cycles (including isolated vertices, which are considered to be paths of length 1). Clearly, there exists an independent set (i.e. a set of vertices no two of which have an edge between them) of size $n/3$ in this graph (a set of triangles being the worst scenario). Let the vertices of this independent set be hard and all the remaining nodes be easy. Since there is no bandwidth probe between a pair of hard nodes and all easy nodes have bandwidth strictly smaller than hard nodes, no information other than the fact that hard nodes have bandwidth at least 1 is revealed after the first round. Suppose the algorithm is actually provided the information that each hard node has bandwidth at least 2. With this additional information, finding the bandwidths of the hard nodes amounts to finding the bandwidths in the inductive scenario and takes an additional $\ell - 1$ rounds. Thus, algorithm $\mathcal{A}$ takes ℓ rounds overall.

■

C. Deterministic Algorithm for No-Departure Model:

We now give an improved algorithm for the no-departure model. This algorithm has a time complexity of $\min(3\lceil \lg \frac{2}{1-\chi} \rceil, 3\lceil \lg n \rceil)$, if the bandwidth distribution slack of the input is χ. So, if χ is bounded away from 1 (i.e. $1-\chi$ is lower bounded by a constant), this algorithm takes constant time. Note that this does not contradict the lower bound since the lower bound was constructed with $\chi = 1$, in which case the algorithm does take $\lceil \lg n \rceil$ time. We need the technical assumption that $d_i > u_i$ (as against $d_i \geq u_i$ previously) for this algorithm.

The algorithm interleaves *pairing*, *matching* and *verification* rounds. The pairing round is exactly the same as the baseline algorithm above—nodes whose upload bandwidth are not known yet are paired and a pairwise bandwidth probe is executed for each pair. The smaller of the observed bandwidths is the observed upload bandwidth of the sender of that probe. Let X and Y be respectively the set of nodes with known and unknown upload bandwidth respectively, after the pairing round. If $|X| < |Y|$, we skip the matching and verification rounds, and go to the next pairing round. Otherwise, we pick the $|Y|$ nodes in X with largest upload bandwidth (call this set Z) and match each node in Y (call it h_i) with a unique node in Z (call it h_j). We now execute bandwidth probes $h_i \rightarrow h_j$. The measured bandwidth in this probe is our guess for u_i (let us call this guess $\tilde{u_i}$). We verify our guess in the verification round. We order the nodes whose upload bandwidth we have guessed in decreasing order of $\tilde{u_i}$ and execute a bandwidth probe from each node to the node immediately before it in the list. If the measured bandwidth of the probe from h_i equals $\tilde{u_i}$, then $u_i = \tilde{u_i}$. Otherwise, the node is added to set of nodes for which the upload bandwidth is unknown. The next pairing round now starts with these nodes. Note that in the verification round, we cannot verify the upload bandwidth of the node with the maximum upload bandwidth—so the upload bandwidth of this node is left unspecified.

We now show that the algorithm is correct, i.e. it terminates with the upload bandwidths of all the nodes, except possibly the node with the maximum upload bandwidth. Clearly, the upload bandwidths revealed by the pairing rounds are correct. So, we only need to check the correctness of the verification rounds. Note that for any h_i, $\tilde{u_i} \leq u_i$. Suppose we send a probe $h_i \rightarrow h_j$ in the verification round. Then, $\tilde{u_i} \leq \tilde{u_j}$. There are two cases. First, suppose $u_i \leq \tilde{u_j}$. Then, the probe $h_i \rightarrow h_j$ reveals u_i since

$$u_i \leq \tilde{u_j} \leq u_j \leq d_j.$$

Hence, if $u_i \neq \tilde{u_i}$, then the algorithm discards $\tilde{u_i}$. The second case is that $u_i > \tilde{u_j}$. Now,

$$d_j > u_j \geq \tilde{u_j}.$$

Thus, the observed bandwidth of the probe is

$$\min(u_i, d_j) > \tilde{u_j} \geq \tilde{u_i}.$$

Hence, the algorithm discards $\tilde{u_i}$ after the verification round.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Theorem 3. *The time complexity of the above algorithm is* $\min(3\lceil \lg \frac{2}{1-\chi} \rceil, 3\lceil \lg n \rceil)$.

Proof: If the matching rounds are not successful is revealing the upload bandwidth of any node, then using Theorem 1, the time complexity is $3\lceil \lg n \rceil$ (since there are at most $\lceil \lg n \rceil$ pairing rounds). Now, there are two cases. In the first case, assume

$$\left(\frac{1-\chi}{2}\right) n \leq 1.$$

Then,

$$\lceil \lg n \rceil \leq \left\lceil \lg \left(\frac{2}{1-\chi} \right) \right\rceil .$$

So, we only need to consider the case

$$\left(\frac{1-\chi}{2}\right) n > 1.$$

Consider the pairing round after which $|Y| \leq (1-\chi)n/2$. Let us now order all the nodes according to increasing upload bandwidth. Then, the maximum index of a node in Y in this ordering is at most n. On the other hand, the index of a node with the minimum upload bandwidth in Z is at least $\lceil \chi n \rceil$. This follows from the observation that Y contains at most $\lfloor (1-\chi)n/2 \rfloor$ nodes among the $\lfloor (1-\chi)n \rfloor$ nodes with indices greater than $\lceil \chi n \rceil$, and therefore the index of any node in Z is at least $\lceil \chi n \rceil$. By the definition of χ, a probe from any node in Y to any node in Z (say $h_i \rightarrow h_j$) reveals the upload bandwidth of h_i. It is easy to verify that if all nodes have the correct guess $\tilde{u_i}$, then the verification round sets all $u_i = \tilde{u_i}$, except the maximum u_i. The number of pairing rounds till we reach the above stage is $3\lceil \lg \frac{2}{1-\chi} \rceil$, and this can be proved on the lines of Theorem 1. ∎

We may note that we can combine the above algorithm with the baseline algorithm, i.e. run the above algorithm for $\lceil \lg n \rceil$ rounds, and if the algorithm does not terminate, then run pairing rounds only from that point onward. This improves the time complexity to $\min\left(3\lceil \lg \left(\frac{2}{1-\chi} \right) \rceil, \lceil \frac{5 \lg n}{3} \rceil \right)$.

V. Randomized Algorithm and Lower Bound

In this section, we will partially resolve the randomized time complexity of the upload bandwidth determination problem. Our first result is a lower bound for randomized algorithms in the no-departure model (and therefore, in the free-departure model as well).

A. Lower Bound for Randomized Algorithms in No-Departure Model:

Theorem 4. *Any randomized algorithm for the upload bandwidth determination problem in the no-departure model uses* $\Omega(\log \log n)$ *rounds in expectation.*

Proof: We construct a probability distribution over bandwidth distributions, $\mathcal{P} : \mathcal{B} \rightarrow [0,1]$ ($\mathcal{B}$ is the set of bandwidth distributions forming the support of the probability distribution), such that every deterministic algorithm for the upload bandwidth determination problem in the no-departure model takes $\Omega(\log \log n)$ time in expectation for input drawn from $\mathcal{P}$. From Yao's minimax principle (for e.g., [23]), the desired lower bound for randomized algorithms follows.

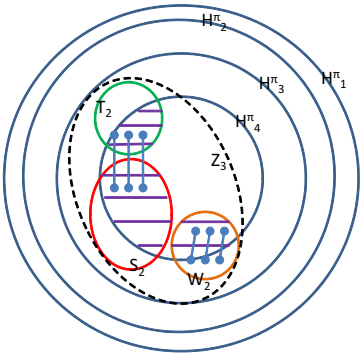


Fig. 2. A description of the notation in the proof of Theorem 4: the concentric blue circles are H_i^πs, the red, green and orange circles are respectively S_2, T_2 and W_2, the black dotted area is $Z_3 = S_2 + W_2 + T_2$, the blue edges indicate the pairings in round 2, and the purple shaded area is $S_3 = Z_3 \cap H_4^\pi$.

$\mathcal{P}$ is a uniform probability distribution, i.e. every bandwidth distribution in $\mathcal{B}$ has probability $1/|\mathcal{B}|$. So, we only need to describe $\mathcal{B}$. Every bandwidth distribution in $\mathcal{B}$ has equal upload and download bandwidth—we call this the bandwidth of the node in the corresponding bandwidth distribution. The bandwidths of the nodes are a one-to-one mapping with $\{1, 2, \ldots, n\}$, and each such mapping (called a *permutation*) is a different bandwidth distribution in $\mathcal{B}$. Thus, $|\mathcal{B}| = n!$.

We view a permutation as an ordering of the nodes in decreasing order of bandwidth. In such a permutation π, H_k^π denotes the set of the first $n^{1/2^{k-1}}$ nodes in the ordering, i.e. the set of $n^{1/2^{k-1}}$ nodes with the largest bandwidth. (Note that $H_1^\pi \supset H_2^\pi \supset H_3^\pi \supset \ldots$) If a node does not receive from or transmit to (henceforth called *compared to*) a node in H_i^π in the first $j-1$ rounds, then we call the node (i,j)-*pure*.

We need to prove that the expected time complexity of any deterministic algorithm in the free-departure model for this input distribution is $\Omega(\log \log n)$. Let $\mathcal{A}$ be any deterministic algorithm. The input is a random permutation π on n nodes (i.e. drawn from $\mathcal{P}$ described previously). We use the following notation (refer to Figure 2:

- X_i is the random variable (r.v.) denoting the set of nodes whose upload bandwidth is unknown at the beginning of round i,
- Y_i is the r.v. denoting the set of (i,i)-pure nodes in H_i^π,
- Z_i is the r.v denoting the set of nodes in H_i^π that are not (i,i)-pure,
- S_i is the r.v denoting the set of nodes in H_{i+1}^π that are not $(i+1,i)$-pure,
- T_i is the r.v. denoting the set of nodes in H_{i+1}^π that are $(i+1,i)$-pure and are compared to a node in S_i in round i, and
- W_i is the r.v denoting the set of nodes in H_{i+1}^π that are $(i+1,i)$-pure and are compared to a node in $H_{i+1}^\pi - S_i$ in round i.

Clearly, $X_i \supseteq Y_i$. We will show that

$$|Y_i| = \Omega(n^{1/2^{i-1}}),$$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

with high probability (whp).[1] This will immediately imply that

$$|X_i| = \Omega(n^{1/2^{i-1}})$$

whp, and therefore,

$$E[|X_{c \log\log n}|] > 0$$

for some small enough constant c. By Markov bound, algorithm $\mathcal{A}$ runs for $\Omega(\log\log n)$ rounds with constant probability. Thus, the expected number of rounds is $\Omega(\log\log n)$.

We need to prove that $E[|Y_i|] = \Omega(n^{1/2^{i-1}})$. First, we observe that

$$Z_i = S_{i-1} + T_{i-1} + W_{i-1}.$$

Now,

$$Y_i = H_i^\pi - Z_i = H_i^\pi - (S_{i-1} + T_{i-1} + W_{i-1}),$$

and

$$S_{i+1} = Z_{i+1} \cap H_{i+2}^\pi \subseteq Z_{i+1} = S_i + T_i + W_i.$$

Each node in S_i can be compared to at most 2 other nodes in a round. Hence,

$$|T_i| \leq 2|S_i|.$$

Hence,

$$|S_{i+1}| \leq 3|S_i| + |W_i|,$$

for $i \geq 1$, and $S_1 = 0$. Thus,

$$|S_{i+1}| \leq \sum_{j=1}^{i} 3^{i-j}|W_j|,$$

for $i \geq 1$, and $S_1 = 0$.

We now prove that $W_i = O(\log n)$ whp, for each i. We prove this by induction on i. By the assumption that uploads are equal to downloads and by the definition of Y_i, the algorithm has no information about nodes in Y_i at the beginning of round i. Therefore, the input distribution restricted to is uniform. Hence, the probability that a particular $(i+1, i)$-pure node in H_{i+1}^π is compared to another $(i+1, i)$-pure node in H_{i+1}^π in round i is at most $n^{1/2^i}/Y_i$. But, $Y_i = \Omega(n^{1/2^{i-1}})$ whp, by the inductive hypothesis. Now, observe that $|W_i|$ is stochastically dominated by a random variable indicating the number of *successes* among $n^{1/2^i}$ *independent* coin tosses, with biased coins having probability of success $1/n^{1/2^i}$. This follows from the anti-correlation among comparisons between $(i+1, i)$-pure node in H_{i+1}^π (i.e. the knowledge that two $(i+1, i)$-pure nodes in H_{i+1}^π are compared in round i decreases the probability of two other $(i+1, i)$-pure nodes in H_{i+1}^π being compared). Using Chernoff bounds, we can therefore conclude that $W_i = O(\log n)$ whp. ■

B. Randomized Algorithm for Free-Departure Model:

We now complement our lower bound with an efficient randomized algorithm for the free-departure model (which also works in the no-departure model). The algorithm has time complexity of $O(\frac{\lg\lg n}{\lg(1/\chi)})$ for $\chi < 1$ and $O(\lg n)$ for $\chi = 1$, with high probability. The algorithm needs to make the additional assumption that the upload and download bandwidths of all nodes are distinct. While this is not immediately practical (see the discussion in [11]), making this assumption allows us gain interesting structural insight into the problem and devise very efficient algorithms.

0: Let X be the set of nodes that have not yet determined their upload bandwidth. Initially $X := H$.
1: **while** $|X| > 1$ **do**
2: Randomly pair nodes in X, and conduct pairwise bandwidth probes.
3: All losers of a pairwise probe know their u_i and are removed from X.
4: Every node that has twice measured the same bandwidth knows its u_i and is removed from X.
5: **end while**
6: **if** $|X| = 1$ **then**
7: Let h_i be the only node in X.
8: Let $h_j := \arg\max_{h_j \notin X} u_j$.
9: Execute $h_i \leftrightarrow h_j$; if h_i loses, u_i is the smaller observed measurement.
10: **end if**

Algorithm 1: Randomized Algorithm for the Free-Departure Model

The idea of the algorithm is to enhance the baseline deterministic algorithm with the following additional rule. We know that of the two measurement results of a pairwise bandwidth probe, the lesser must correspond to an upload bandwidth and the greater of the two might be either an upload or download bandwidth. If a node h_i participates in multiple bandwidth probes (without having been able to determine its upload bandwidth), only the maximum of the greater bandwidths observed in the different rounds is a candidate for u_i; all other observed measurements must be download bandwidths of the other nodes involved in the pairwise probes. Therefore, if h_i observes the same bandwidth measurement in two transmissions (to different nodes and in different rounds), the observed measurement must necessarily be u_i.

Thus, we adjust the algorithm as shown in Algorithm 1. Implementation is easy: every node keeps a list of all measurement values it has seen so far on transmissions, until it either loses a pairwise probe, or it sees a measurement value that is already in its list. Clearly, in either of these two cases, the observed measurement is the node's upload bandwidth—hence the algorithm is correct. We terminate when only one node (call it h_i) does not know its upload capacity. At this point, we have a single round where a pairwise bandwidth probe is executed between h_i and the node with the maximum upload bandwidth among the known nodes (call it h_j). If h_i loses, then the smaller of the observed bandwidths is the upload bandwidth of h_i; else, h_i has the maximum upload bandwidth among all nodes and the algorithm terminates without determining its upload bandwidth.

To analyze this algorithm, we need some additional terminology. We say that a host h_i is *marked* in a round if the greater of the measured bandwidths is its upload bandwidth u_i.[2] Let $p_i(r)$ be the probability that host h_i survives round r

[1] A property is said to hold *with high probability* if the probability of its violation can be bounded by an arbitrary inverse polynomial in n, the number of nodes.

[2] Note that the algorithm is not aware of the marking of a node; it is used only in the analysis.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

and $q_i(r)$ be the probability that host h_i survives round r but has been marked. Then, $t_i(r) = p_i(r) - q_i(r)$ represent the probability that host h_i survives round i unmarked.

We further define

$$X_i(r) = \frac{\sum_{\chi i < j \leq i} p_j(r)}{\sum_{1 \leq j \leq n} p_j(r)}$$
$$Y_i(r) = \frac{\sum_{1 \leq j \leq \chi i} p_j(r)}{\sum_{1 \leq j \leq n} p_j(r)},$$

where $X_i(r)$ represents the conditional probability that a host which has survived unmarked until round r is marked in round r or a node which has already been marked gets to know its upload bandwidth; and $Y_i(r)$ represents the conditional probability that a host which has already been marked before round r does not get to know its upload bandwidth in round r or a host which has survived unmarked till round $r-1$ does not get marked. The following lemma follows directly from the above interpretation.

Lemma 2. *The following equations hold:*

$$t_i(r+1) = t_i(r) \cdot Y_i(r)$$
$$q_i(r+1) = t_i(r) \cdot X_i(r) + q_i(r) \cdot Y_i(r).$$

Using this lemma and plugging in the definitions of $X_i(r)$ and $Y_i(r)$, we can now derive expressions that characterize the decline of $t_i(r)$ and $q_i(r)$ as the number of rounds r increases. (The detailed proof is in the appendix.)

Lemma 3. *The following equations hold:*

$$t_i(r) = \beta \left(\frac{i}{n}\right)^{2^r-1}, \textit{where } \beta = \chi^{2^r-1}$$
$$q_i(r) = \gamma \left(\frac{i}{n}\right)^{2^r-1}, \textit{where}$$

$$\gamma = \chi^{2^{r-1}-1}(1 + \chi^{2^{r-2}} + \chi^{2^{r-2}+2^{r-3}} + \ldots + \chi^{2^{r-2}+\ldots+2^0} - r\chi^{2^{r-1}}).$$

Observe that (omitting asymptotically smaller factors) $p_i(r) = \left(\frac{i}{n}\right)^{2^r-1} \cdot \chi^{2^{r-1}-1}$. If $\chi < 1$, for any host h_i, $p_i(r)$ is inverse polynomial in n after $r = O(\frac{\lg \lg n}{\lg(1/\chi)})$ rounds. This allows us to use the union bound to assert that all nodes know their upload bandwidth after $O(\frac{\lg \lg n}{\lg(1/\chi)})$ rounds whp—so the algorithm terminates in $O(\frac{\lg \lg n}{\lg(1/\chi)})$ rounds. If $\chi = 1$, we use the fact that we need not find the maximum upload bandwidth. Thus, $i \leq n-1$ and $\frac{1}{n})^{2^r-1}$ is inverse polynomial in n when $r = O(\lg n)$. Again, using the union bound, we conclude that all nodes except the one with the maximum upload bandwidth know their upload bandwidth after $O(\lg n)$ rounds whp—hence the algorithm terminates in $O(\lg n)$ rounds. The following theorem follows from this analysis.

Theorem 5. *The time complexity of the above algorithm is* $O(\frac{\lg \lg n}{\lg(1/\chi)})$ *for* $\chi < 1$ *and* $O(\lg n)$ *for* $\chi = 1$*, with high probability.*

Note that while the value of χ appears in the analysis, the algorithm does not need to know this value.

VI. Practical Implementation – Deployment Environment

In this section, we discuss implementation of different bandwidth determination algorithms, and present measurement results obtained from simulation and a real PlanetLab [27] deployment. We also present a discussion of the *intended deployment environment*, the practical *probing overhead* as well as *distributed implementations*.

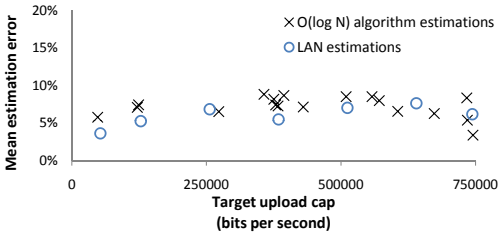
Evaluation: Due to space limitations, we can only present a small subset of the evaluations that we have conducted. For our empirical experiments, we implemented the basic $O(\log n)$-time algorithm and deployed it on the PlanetLab distributed testbed [27]. To provide ground-truth about available bandwidths, we capped transfer speeds on each node using a custom rate limiter. Caps were set using a bandwidth distribution of 30% dial-up hosts and 70% hosts from the DSL Report data set [5]. The DSL Report data represents an empirical measurement of broadband transfer speeds across America. We added the dial-up hosts to cohere with survey data showing that many people still lack broadband connectivity [8].

Fig. 3(left) shows the result of a sample run of the algorithm. The graph compares the estimation accuracy of our algorithm (over the wide-area Internet) to upload measurements made between hosts on the same LAN. The LAN estimates show the inherent measurement bias of our one-way bandwidth estimator; ideally, our wide-area algorithm should not introduce additional biases. Fig. 3(left) shows that this is the case. Essentially, our wide-area algorithm did not add *new* estimation artifacts to those that already arose from our simple bandwidth estimators.

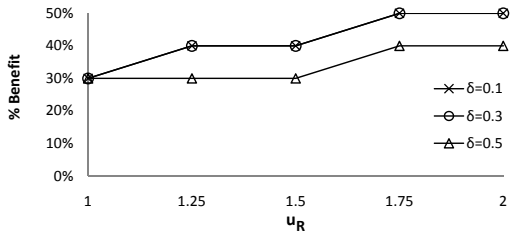
To better understand the benefit of the robust $O(\log \log n)$-time algorithm versus the baseline $O(\log n)$-algorithm, we also conducted simulations using synthetic bandwidth data. The analytic distribution consists of a δ-fraction of dial-up hosts and a $(1-\delta)$-fraction of broadband hosts. For broadband hosts, we generate two independent random variables. U represents the distribution of each host's upload bandwidth, and R represents the distribution of the ratio of each host's download bandwidth to its upload bandwidth. U and R follow log normal distributions with parameters μ_U, σ_U, μ_R, and σ_R. To preserve directional asymmetry, R is lower-bounded to unity. Random variable D, which governs each host's download bandwidth, is then computed as the product of random variables U and R, i.e., $D = U \cdot R$. Because the product of log normals is log normal, the marginal distribution of D is a log normal distribution with parameters $\mu_D = \mu_U + \mu_R$ and $\sigma_D = \sqrt{\sigma_U^2 + \sigma_R^2}$. When fitted to the empirical distribution from the DSL Report data, ($\delta = 0.3$, $\mu_U = 6.6$, $\sigma_U = 0.68$, $\mu_R = 1.4$, $\sigma_R = 0.49$). In the simulations, we varied δ and μ_R, changing the fraction of dial-up hosts and the "offset" between a host's upload and download speed.

Fig. 3(right) depicts the benefit of the randomized algorithm with respect to the baseline $O(\log n)$ algorithm in a system

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.



(a) PlanetLab deployment of $O(\log n)$-time algorithm ($n = 32$).



(b) Relative benefit of $O(\log \log n)$-time algorithm ($n = 1024$)

Fig. 3. Practical Evaluation

with 1024 hosts. The "benefit" is defined as the reduction in rounds needed for the algorithm to terminate. An algorithm terminates when each host knows its upload speed. In the $O(\log n)$ algorithm, this happens when each host has lost a pairwise exchange; in the randomized algorithm, this happens when each host has lost an exchange *or* seen the same upload speed twice. The deterministic algorithm always requires $O(\log n)$ steps. As shown in Fig. 3(right), the randomized algorithm reduces termination time by 30% to 50%. The benefits increase as μ_R grows—the bigger the stretch between hosts' upload and download speeds, the more likely that a node will see its real upload speed when it is the sender in a pairwise transfer. In turn, this makes it more likely that a node will see its upload speed twice and stop.

Intended Deployment Environment: There are many types of peer-to-peer systems, each with varying levels of host churn. Our algorithm is intended for systems with medium to long session lengths. In particular, we do not target environments like file-trading where peers often exit the system after a few minutes of participation. Instead, we target online gaming or collaborative streaming applications in which users show up to participate in the service, receive coordination metadata from a service manager, and then use the service for tens of minutes or longer.

In a wide-area peer-to-peer system, some hosts may reside behind network address translators or "NATs". NATs may prevent two hosts in different domains from communicating directly; alternatively, they may permit only one member of the pair to initiate a bandwidth probe, thus ruling out the possibility of pairwise probes between these nodes. These restrictions are a hassle for the designers of any peer-to-peer service, and making our algorithm robust to all possible NAT constraints is beyond the scope of this paper. Thus, in our evaluations, we assume that each host has a public IP address, or resides behind a cone NAT [26] that allows outside parties to initiate communication with the host.

Probing Overhead: Our distributed algorithm is essentially a scheduler for pairwise bandwidth probes. However, the algorithm is agnostic with regard to how two hosts in a pairing actually determine their unidirectional bandwidths. In the PlanetLab experiments described below, we generated these estimates by timing simple bulk data transfers. However, as explained in Section II, there are more sophisticated techniques for generating unidirectional bandwidth estimates. In a real-life deployment of our algorithm, we would use one of these tools to measure one-way bandwidths. Since our algorithm relies on active network probing, a natural concern is whether a real deployment would generate excessive traffic. Fortunately, modern probing tools do not produce an overwhelming volume of packets. For example, the Spruce tool [28] generates 300 KB of traffic per one-way measurement, and the IGI tool [14] produces 130 KB. Given that the size of an average web page is 300KB [29], the volume of probing traffic is quite acceptable. The rate of traffic generation is also reasonable. For example, the maximum transmission rate of the Spruce tool is 240 Kbps, which is similar to the streaming rate of a YouTube video. Thus, the probing traffic generated by our distributed algorithm will be reasonable in both volume and rate.

Distributed Implementation: All our bandwidth determination algorithms in this paper are inherently distributed in nature since bandwidth probes are executed in parallel in each round. However, as presented above, some leader node requires coordination at each round, to determine pairings for the subsequent round. For large systems, the network-transmission load on the leader can become substantial. Therefore, we are interested in fully distributed solutions. Fortunately, the basic $O(\log n)$-algorithm can be implemented in a fully distributed fashion in a standard way: A binary tree spanning all nodes is constructed, and each inner node is responsible for one pairwise bandwidth probe. Let h_i be responsible for a bandwidth probe $h_k \leftrightarrow h_\ell$. The winner of this probe reports the result to h_i, and asks h_i's parent-node in the tree for its next pairing. Since there are $n-1$ inner nodes, the procedure can be implemented such that every node only needs to send a constant number of control messages.

Unfortunately, finding a fully distributed for either the $O(1)$-algorithm or the $O(\log \log n)$-algorithm without any central coordination by a leader node is more challenging. Specifically, the constant time algorithm needs a central leader for computing maxima across all known upload bandwidths. As for the $O(\log \log n)$, it is an intriguing open problem whether a pseudo-deterministic schedule like for the $O(\log n)$-algorithm above exists. The problem is that the number of nodes surviving round k is not a-priori clear, and hence generating a pseudo-random sequence that would inform nodes as to their next match in a distributed fashion seems challenging.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

VII. Conclusions & Future Work

There are many challenging directions for future research. It will be interesting to derive lower bounds for the case of $\chi < 1$. Similarly, for the download problem, our upper and lower bounds are still divided by a logarithmic factor. Another interesting algorithmic problem arises due to the NAT problem discussed in Section VI, which would generalize the problem from a star-topology to a general graph. More generally, the hub-and-spoke model with access links is a model hat has received relatively little attention in the distributed computing community (say, compared to classical message passing models on general graphs). We believe that there are numerous practically relevant distributed computing problem in this model.

References

[1] American National Standards Institute. Network and Customer Installation Interfaces–Asymmetric Digital Subscriber Line (ADSL) Metallic Interface, 1998. ANSI T1.413.

[2] A. Bharambe, J. Douceur, J. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games. In *Proceedings of SIGCOMM*, pages 389–400, August 2008.

[3] B. Biskupski, R. Cunningham, J. Dowling, and R. Meier. High-Bandwidth Mesh-based Overlay Multicast in Heterogeneous Environments. In *Proceedings of ACM AAA-IDEA*, October 2006.

[4] A. Bozdog, R. van Renesse, and D. Dumitriu. SelectCast—A Scalable and Self-Repairing Multicast Overlay Routing Facility. In *Proceedings of SSRS*, pages 33–42, October 2003.

[5] Broadband Reports. Broadband reports speed test statistics. http://www.dslreports.com/archive, October 29, 2008.

[6] Cable Television Laboratories, Inc. Data-Over-Cable Service Interface Specifications (DOCSIS 3.0): MAC and Upper Layer Protocols Interface Specification, January 2009.

[7] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27–28:297–318, October 1996.

[8] Communication Workers of America. Speed Matters: A Report on Internet Speeds in All 50 States. http://www.speedmatters.org/document-library/sourcematerials/sm_report.pdf, July 2007.

[9] M. Dischinger, A. Haeberlen, I. Beschastnikh, K. Gummadi, and S. Saroiu. SatelliteLab: Adding Heterogeneity to Planetary-Scale Network Testbeds. In *Proceedings of SIGCOMM*, pages 315–326, August 2008.

[10] J. R. Douceur, J. R. Lorch, and T. Moscibroda. Maximizing total upload in latency-sensitive p2p applications. In *Proceedings of 19th SPAA*, pages 270–279, 2007.

[11] J. R. Douceur, J. W. Mickens, T. Moscibroda, and D. Panigrahi. ThunderDome: Discovering Upload Constraints Using Decentralized Bandwidth Tournaments. In *Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNext)*, 2009.

[12] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-Dispersion Techniques and a Capacity-Estimation Methodology. *IEEE/ACM Transactions on Networking*, 12(6):963–977, December 2004.

[13] J. Ghoshal, B. Ramamurthy, and L. Xu. Variable Neighbor Selection in Live Peer-to-Peer Multimedia Streaming Networks. Technical Report TR-UNL-CSE-2007-021, University of Nebraska-Lincoln Department of Computer Science and Engineering, September 2007.

[14] N. Hu and P. Steenkiste. Evaluations and Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communication*, 21(6):879–894, August 2003.

[15] N. Hu and P. Steenkiste. Exploiting Internet Route Sharing for Large Scale Available Bandwidth Estimation. In *Proceedings of IMC*, pages 187–192, October 2005.

[16] International Telecommunication Union, Standardization Sector. ITU-T Recommendation V.92: Enhancements to Recommendation V.90, November 2000.

[17] M. Jain and C. Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *Proceedings of the Passive and Active Measurements Workshop*, pages 14–25, March 2002.

[18] X. Jin, W. Yiu, S. Chan, and Y. Wang. On Maximizing Tree Bandwidth for Topology-Aware Peer-to-Peer Streaming. *IEEE Transactions on Multimedia*, 9(8):1580–1592, December 2007.

[19] K. Lai and M. Baker. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, pages 123–134, March 2001.

[20] K. Lakshminarayanan, V. Padmanabhan, and J. Padhye. Bandwidth Estimation in Broadband Access Networks. In *Proceedings of IMC*, pages 314–321, October 2004.

[21] Z. Lotker, E. Pavlov, B. Patt-Shamir, and D. Peleg. MST construction in $O(\log \log n)$ communication rounds. In *Proceedings of 15th SPAA*, pages 94–100, 2003.

[22] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proceedings of the GLOBECOM*, pages 415–420, November 2000.

[23] R. Motwani and P. Raghavan. *Randomized Algorithms*.

[24] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, and R. Baraniuk. Multifractal cross traffic estimation. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement*, September 2000.

[25] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Proceedings of the Passive and Active Measurement Workshop*, March 2003.

[26] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389, October 2008.

[27] N. Spring, L. Peterson, A. Bavier, and V. Pai. Using PlanetLab for network research: myths, realities, and best practices. *SIGOPS Operating Systems Review*, 40(1):17–24, January 2006.

[28] J. Strauss, D. Katabi, and F. Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. In *Proceedings of IMC*, pages 39–44, October 2003.

[29] WebSiteOptimization.com. *Average Web Page Size Triples Since 2003*. http://www.websiteoptimization.com/speed/tweak/average-web-page/.

Appendix

Proof of Lemma 3: We prove the theorem by induction. For the base case, we know that for any h_i, $p_i(0) = 1$ and $q_i(0) = 0$. For the inductive case, assume that the hypothesis is true for round r. Then we have (replacing all summations by integrations—this does not affect asymptotic behavior)

$$Y_i(r) = \frac{(\beta+\gamma)\int_0^{\chi i}(\frac{x}{n})^{2^r-1}dx}{(\beta+\gamma)\int_0^{n}(\frac{x}{n})^{2^r-1}dx} = \left(\frac{\chi i}{n}\right)^{2^r}$$

$$X_i(r) = \frac{(\beta+\gamma)\int_{\chi i}^{i}(\frac{x}{n})^{2^r-1}dx}{(\beta+\gamma)\int_0^{n}(\frac{x}{n})^{2^r-1}dx} = (1-\chi^{2^r})\cdot\left(\frac{i}{n}\right)^{2^r}.$$

Plugging these values into Lemma 2 implies

$$t_i(r+1) = \left(\frac{\chi i}{n}\right)^{2^r-1}\cdot\left(\frac{\chi i}{n}\right)^{2^r} = \left(\frac{\chi i}{n}\right)^{2^{r+1}-1}$$

$$\begin{aligned} q_i(r+1) &= \left(\frac{\chi i}{n}\right)^{2^r-1}\left(1-\chi^{2^r}\right)\left(\frac{i}{n}\right)^{2^r} + \\ &\quad (\frac{i}{n})^{2^r-1}\chi^{2^{r-1}-1}(1+\chi^{2^{r-2}} + \\ &\quad \ldots+\chi^{2^{r-2}+\ldots+2^0}-r\chi^{2^{r-1}})\left(\frac{\chi i}{n}\right)^{2^r} \\ &= (\frac{i}{n})^{2^{r+1}-1}\chi^{2^r-1}(1+\chi^{2^{r-1}} + \\ &\quad \ldots+\chi^{2^{r-1}+\ldots+2^0}-(r+1)\chi^{2^r}) \end{aligned}$$

■