Feedback-assisted robust estimation of available bandwidth

Kyung-Joon Park[a], Hyuk Lim[b,1], Jennifer C. Hou[a], and Chong-Ho Choi[c]

[a]*Department of Computer Science*
University of Illinois, 201 N. Goodwin Avenue, Urbana, IL USA
Email: {kjp, jhou}@uiuc.edu

[b]*Department of Information and Communications*
Gwangju Institute of Science and Technology, Gwangju, Korea
Email: hlim@gist.ac.kr

[c]*School of Electrical Engineering and Computer Science*
Seoul National University, Seoul, Korea
Email: chchoi@csl.snu.ac.kr

Abstract

The packet pair technique is a widely used method for estimating the available bandwidth of an end-to-end network path. We characterize the stochastic nature of the packet pair dispersion caused by bursty cross traffic and its adverse effect on available bandwidth estimation. In order to overcome this difficulty, we introduce a novel concept of the *relative distance*, which can be obtained from the relation between input and output gaps of packet pairs. By exploiting the *quasi-invariant* characteristic of the relative distance, we develop a feedback-assisted, robust and non-intrusive approach for estimating the available bandwidth. The method entitled *bTrack* periodically sends two pairs of probe packets of different sizes and exploits the relative distance for accurate estimation of the available bandwidth. The amount of the probing traffic is independent of the available bandwidth and is adjustable by tuning of the probing period, which shows the non-intrusive nature of *bTrack*. We give the convergence analysis of *bTrack* based on the theory of the stochastic approximation, which guarantees the robust performance of *bTrack* under bursty cross traffic. We verify via extensive *ns-2* simulations and empirical experiments (over campus intranets and the Internet) that *bTrack* tracks the available bandwidth very well and is not intrusive.

Key words: Network measurement, available bandwidth, packet pair technique.

[1] Corresponding author.

1 Introduction

To facilitate design and development of efficient resource management protocols, it will be greatly helpful to better understand the dynamic properties and behaviors of end-to-end paths on the Internet. In order to prevent routers from overloading with traffic measurement and report tasks, it is desirable for end hosts to infer these properties on an end-to-end basis. One of the most useful path attributes is the *available bandwidth* along an end-to-end path. The definition of the available bandwidth is given in [1] as "the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic on the path." (In contrast, the bottleneck bandwidth on an end-to-end path is defined as the maximum possible bandwidth along the path, i.e., the minimum link capacity along the path.) The knowledge of the available bandwidth can be exploited in numerous applications [1]. For example, in the QoS sensitive Internet services such as peer-to-peer file sharing applications and on-demand multimedia streaming applications, a client can greatly benefit if it is connected to a peer/server via a path of sufficient available bandwidth [2–4]. The performance of overlay networks can also be improved if the available bandwidth between nodes can be measured on an end-to-end basis [3].

In general, designing a mechanism to accurately measure the available bandwidth on an end-to-end path is a challenging research problem. Usually an end-to-end mechanism injects one or more unicast/multicast *probe* packets, measures/records the round-trip time (the single-packet technique [5,6]) or the time dispersion between two consecutive packets (the packet-pair technique [7–13]), and uses the measured information to infer the network attribute of interest. It is a critical design criterion how to ensure that the end-to-end measurement method is accurate and yet non-intrusive (i.e., it neither introduces a significant amount of probe packets into the network nor interferes other traffic on the path.) The mechanism should also be robust to dynamic traffic changes.

Many end-host-based mechanisms that use the packet train technique [1,14–16] and the packet-pair technique [11,13] have been proposed in the literature to measure the available bandwidth along a path. We will provide a detailed summary of existing work in Section 5. In summary, tools using the packet train technique could be intrusive if the number of probe packets used in each packet train is large. Approaches that are based on the packet pair technique can adjust the sending rate and pattern of probe packets, by varying the time dispersion between two consecutive packets, and hence are less intrusive. However, since the available bandwidth is inferred by measuring at the receiver the inter-arrival time of two probe packets in each packet pair, if the analytic model that infers the available bandwidth does not take account of bursty cross traffic, the estimation results may not be accurate.

Recently, several researchers have presented stochastic analysis of the packet pair technique from different points of view [17–20]. It was revealed from these work

(despite their different points of view) that bursty cross traffic makes available bandwidth estimation a very challenging task by introducing significant estimation errors. Hence, a natural question arises: Can we still devise a robust estimation mechanism for available bandwidth?

This paper deals with an answer to this question, i.e., how to overcome busrtiness of cross traffic in available bandwidth estimation. To this end, we characterize the stochastic nature of the packet pair technique and its adverse effect on available bandwidth estimation. Since the stochastic characteristic of the packet pair technique comes from burstiness in cross traffic that is not controllable at end hosts, it becomes a very challenging task to devise a robust method for estimating the available bandwidth in an end-to-end manner. Here, in order to overcome burstiness in cross traffic, we introduce a novel concept of the *relative distance*, which is a metric robust to burstiness in cross traffic.

Based on the concept of the relative distance, we propose a feedback-assisted end-to-end mechanism, called *bTrack*, to track the available bandwidth on a path. *bTrack* has the following crucial properties: (i) *bTrack is robust. bTrack* exploits the quasi-invariant characteristic of the relative distance and consequently, is very robust to bursty cross traffic. (ii) *bTrack is non-intrusive. bTrack* is non-intrusive in the sense that the bandwidth consumed by the probing traffic is considerably small, independent of the value of the available bandwidth, and can be adjusted by varying the probing period and the probe packet sizes. (iii) *bTrack is efficient.* Unlike most previous mechanisms that only use the information on the packet dispersion to estimate the available bandwidth, *bTrack* further utilizes the information on the probe packet size to fully exploit all the information available from the packet-pair relation.

The feedback-assisted mechanism of *bTrack* corresponds to a stochastic approximation algorithm and we give the convergence analysis of *bTrack* based on the theory of the stochastic approximation [21,22]. Our convergence analysis guarantees robust performance of *bTrack* under bursty cross traffic. Through extensive *ns-2* simulations and empirical studies on campus intranets and the Internet, we show that *bTrack* tracks the available bandwidth very well with small errors (less than 8% of the value to be tracked), and is non-intrusive (i.e., the bandwidth consumed by the probing traffic is extremely low).

The rest of the paper is organized as follows. In Section 2, we provide background material that pertains to the problem addressed in the paper. In Section 3, we present the concept of the relative distance and the corresponding end-to-end mechanism for measuring the available bandwidth. Also, the convergence analysis of the proposed mechanism is presented based on the stochastic approximation. Following that, we present in Section 4 both *ns-2* simulation results and empirical results. In Section 5, we provide a detailed summary of the various packet pair models and measuring methods in the literature. Finally, we conclude in Section 6 with a list of

future work.

2 Preliminary

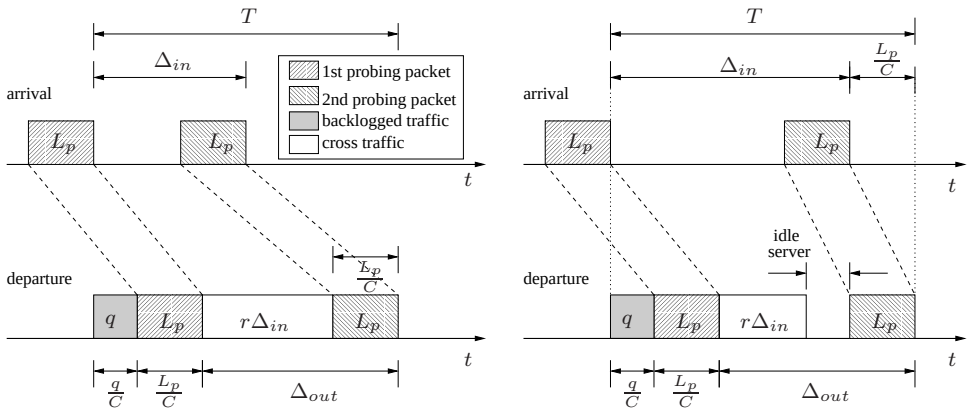
In this section, we introduce the fluid packet pair model [12] (called the single-hop gap model (SHG) in [15]), which describes the relation between the inter-departure time and the inter-arrival time of a packet pair in a single link based on the assumption of fluid cross traffic with a constant rate. Here, we point out that the fluid packet pair model is not so efficient to estimate the available bandwidth under bursty cross traffic. This observation serves as the starting point for developing our algorithm. It will also facilitate our discussion on related work in Section 5.

In most of the packet pair approaches, the sender sends pairs of (back-to-back) probe packets with small *inter-departure* times, and the receiver measures the *inter-arrival* time of each packet pair. Then the available bandwidth is inferred by using the analytical model that characterizes the relation between the inter-departure times of probe packets at the sender and their inter-arrival times at the receiver. Here the inter-departure time of a packet pair is defined as the interval between the times when the the first packet leaves from the sender and when the the second packet leaves. Similarly, the inter-arrival time of a packet pair is defined as the interval between the times when the first packet arrives at the receiver and when the second packet arrives. We denote, respectively, the inter-departure time and the (measured) inter-arrival time of a packet pair as Δ_{in} and Δ_{out}.

Consider a single link with capacity C. The sender transmits a pair of probe packets to the receiver through this link. The size of each probe packet is L_p. Let the stochastic process $q(t)$ denote the amount of traffic (if any) queued when the first probe packet arrives at the queue. We have $\overline{q} = 0$ where $\overline{q} := \mathbb{E}[q(t)]$ since the cross traffic is modeled as a fluid flow with a constant rate of r. Let T denote the interval between the time when the last bit of the first probe packet arrives at a link and that when the last bit of the second probe packet departs that link. As depicted in Fig. 1, two sub-cases are considered depending on whether the link is fully utilized or not. In the case that the link is fully utilized during the interval T (Fig. 1 (a)), we have $\Delta_{out} = (r\Delta_{in} + L_p)/C$. On the other hand, the server may become idle in the interval T if Δ_{in} is sufficiently large (Fig. 1 (b)). We define the characteristic value Δ^* as the minimum value of Δ_{in} that makes the server idle in the interval T. As depicted in Fig. 1, the characteristic value Δ^* can be calculated as

$$\Delta^* = \frac{L_p}{C(1-u)},$$

where $u = r/C$. If the inter-departure time Δ_{in} is larger than Δ^*, the queue is under-utilized, and as depicted in Fig. 1 (b), T is equal to both $\Delta_{in} + L_p/C$ and



(a) When the second probe packet sees the server busy (b) When the second probe packet sees the server idle

Fig. 1. Behavior of a packet pair in a single link with fluid cross traffic [20].

$\Delta_{out} + L_p/C$, and hence $\Delta_{out} = \Delta_{in}$. The inter-arrival time Δ_{out} can be written as

$$\Delta_{out} = \begin{cases} u\Delta_{in} + \frac{L_p}{C}, \ \Delta_{in} \leq \Delta^*; \\ \Delta_{in}, \qquad \text{otherwise.} \end{cases} \tag{1}$$

In (1), Δ_{out} is a piecewise increasing linear function of Δ_{in} with two different slopes u and 1, and the slope change occurs at $\Delta_{in} = \Delta^*$. Note that Δ^* is inversely proportional to the available bandwidth $C(1-u)$. Actually, most of the existing estimation algorithms for available bandwidth exploit the characteristic value Δ^* and estimate the available bandwidth by using (1).

Now, we validate the deterministic model (1) via ns-2 simulations. We perform simulations for a single-hop topology composed of two drop-tail routers that are connected by a link with the capacity of 1Mb/s and the propagation delay of 15ms. In the simulation, the cross traffic consists of 15 Poisson sources, and each source generates packets at the rate of 32 Kb/s with the packet size of $L = 100$ bytes. Figure 2 shows the relation between $\mathbb{E}[\Lambda_{out}]$ versus Δ_{in} where $\Lambda_{out} := \Delta_{in} - \Delta_{out}$. The sizes of probe packets are $500, 1000$, and 1500 bytes, respectively. Note that we introduce Λ_{out} to highlight the change of $\partial\mathbb{E}[\Delta_{out}]/\partial\Delta_{in}$ around the characteristic value $\Delta^* \left(= L_p/C(1-u)\right)$. In Fig. 2, each data point is an average value of 300 trials and the solid lines are obtained from the deterministic model (1).

Figure 2 shows that the simulation data asymptotically converges to the analytical values. However, the model error becomes significant when Δ_{in} is around Δ^*, i.e., $\partial\mathbb{E}[\Lambda_{out}]/\partial\Delta_{in}$ changes smoothly around $\Delta_{in} = \Delta^*$ in the simulation results while it changes abruptly at $\Delta_{in} = \Delta^*$ with the deterministic model. This indicates that, under bursty cross traffic, Δ_{out} is no more a deterministic value obtained from (1), but a stochastic value determined from statistical characteristics of cross traffic. Further, $(\Delta_{in}, \mathbb{E}[\Delta_{out}])$ curve deviates from the deterministic relation of (1), especially around $\Delta_{in} = \Delta^*$ (actually, the region around $\Delta_{in} = \Delta^*$ was termed as the

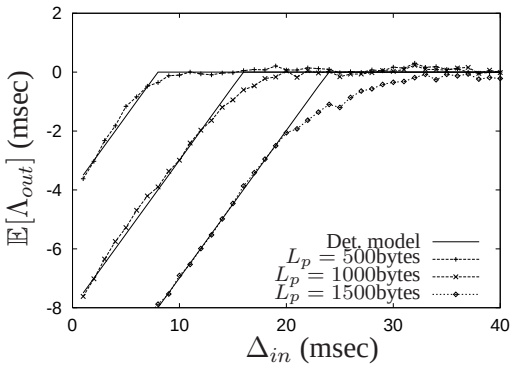


Fig. 2. Deterministic model vs. simulation result under Poisson cross traffic.

biased probing region in [17].) Consequently, as burstiness in cross traffic becomes more severe, the estimation error of available bandwidth based on Δ^* becomes more significant, even in an average sense, because of the error in the deterministic model of (1). Thus, in order to efficiently estimate the available bandwidth under bursty cross traffic, it is not sufficient to exploit the characteristic value based on (1). A metric for the available bandwidth, which is more robust to randomness in cross traffic, is needed.

3 Feedback-assisted Robust Estimation

In this section, we propose a feedback-assisted robust mechanism, called *bTrack*, for estimating the available bandwidth. Here, we give rigorous analysis of *bTrack* under a multi-hop path with a single tight link. Even though *bTrack* is expected to work fairly well under any cases in practice, rigorous analysis on the general multi-hop case will be our future work.

Ideally the available bandwidth on an end-to-end path can be measured by searching for the characteristic value. However, because of the biased probing region (explained in Section 2) that exists in the vicinity of the characteristic value, methods devised by searching for the characteristic value may be unreliable.

Instead of the characteristic value, we propose a novel concept of the *relative distance*, which is the difference between the characteristic values of two pairs of probe packets of *different* packet sizes. A detailed explanation on the relative distance will be given in subsequent subsections. *bTrack* tracks the available bandwidth by exploiting the relative distance. The feedback mechanism of *bTrack* works as follows. The time axis is divided into intervals and in each interval, the sender transmits two pairs of probe packets to the receiver, with the inter-departure times determined based on the results from the previous interval. After measuring the inter-arrival times of the two pairs of probe packets, the receiver estimates the available bandwidth on the end-to-end path, and advises the sender of the inter-departure

time to be used in the next interval.

3.1 Characterization of packet-pair dispersion relation under bursty cross traffic

In the packet-pair technique for available bandwidth estimation, we observe Δ_{out} with the knowledge of Δ_{in} and L_p and try to find out the available bandwidth. The difficulty of bandwidth estimation with the packet-pair technique lies in the randomness of Δ_{out}, which comes from the burstiness of cross traffic. Here, we characterize Δ_{out}, or equivalently, Λ_{out} (=$\Delta_{in} - \Delta_{out}$) in terms of all the related parameters.

For a given time instant t, $\Lambda_{out}(t)$ can be characterized by $\Delta_{in}(t)$, L_p, $\xi(t,\omega)$, and $\boldsymbol{C}$, where $\xi(t,\omega)$ is a sequence of independent random variables, which represent the random effect of bursty cross traffic, defined on some probability space $(\Omega, \mathcal{F}, P)$, and $\boldsymbol{C} = (C_l, l \in L(\mathcal{P}))$, where C_l is the link capacity of link l and $L(\mathcal{P})$ is a set of links that constitute the interested path $\mathcal{P}$. Hence, the functional relation of $\Lambda_{out}(t)$ can be denoted as follows.

$$\Lambda_{out}(t) = F(\Delta_{in}(t), L_p;\ \xi(t,\omega), \boldsymbol{C}). \tag{2}$$

Among the arguments of the function F in (2), since $\boldsymbol{C}$ is fixed for a given path $\mathcal{P}$, $\xi(t,\omega)$ is the only random variable, which comes from bursty cross traffic and cannot be controlled in an end-to-end manner. Δ_{in} and L_p are the control knobs, which can be adjusted to obtain information on the available bandwidth. Therefore, for given $\boldsymbol{C}$, the function F in (2) can be further divided into two parts, i.e., a deterministic part that is responsible for the expected value of $\Lambda_{out}(t)$ and a stochastic part for the random perturbation by bursty cross traffic as follows:

$$F(\Delta_{in}(t), L_p;\ \xi(t,\omega), \boldsymbol{C}) = \mathbb{E}[F(\Delta_{in}(t), L_p;\ \xi(t,\omega), \boldsymbol{C})] + G(t, \Delta_{in}(t), L_p, \xi(t,\omega)), \tag{3}$$

where $G(t,x,y,\xi(t,\omega))$ has a finite variance with $\mathbb{E}[G(t,x,y,\xi(t,\omega))] = 0$ for any x, y and t and $G(t,x,y,z)$ is an unknown function of t, x, y, and z. If we let $\mathbb{E}[F(\Delta_{in}(t), L_p;\ \xi(t,\omega), \boldsymbol{C})] =: R(\Delta_{in}(t), L_p)$ for given $\boldsymbol{C}$, (2) and (3) gives the following relation:

$$\Lambda_{out}(t) = \underbrace{R(\Delta_{in}(t), L_p)}_{\text{Deterministic part}} + \underbrace{G(t, \Delta_{in}(t), L_p, \xi(t,\omega))}_{\text{Stochastic part}}. \tag{4}$$

The deterministic part $R(\Delta_{in}(t), L_p)$ in (4) has been extensively studied in recent studies [17–20]. Here, we introduce several characteristics of $R(\Delta_{in}(t), L_p)$ that will be exploited to devise a robust estimation algorithm.

First, $R(\Delta_{in}(t), L_p)$ is an increasing function of $\Delta_{in}(t) \in [0, \infty]$. Hence,

$$\frac{\partial R(\Delta_{in}(t), L_p)}{\partial \Delta_{in}(t)} \geq 0, \quad \Delta_{in}(t) \in [0, \infty]. \tag{5}$$

Second, $R(\Delta_{in}(t), L_p)$ is upper bounded by the deterministic relation in (1), i.e.,

$$R(\Delta_{in}(t), L_p) \leq U(\Delta_{in}(t), L_p), \tag{6}$$

where

$$U(\Delta_{in}(t), L_p) = \begin{cases} (1-u)\Delta_{in}(t) - \frac{L_p}{C}, \ \Delta_{in}(t) \leq \Delta^*; \\ 0, \qquad\qquad \text{otherwise.} \end{cases}$$

Third, the difference between $U(\Delta_{in}(t), L_p) - R(\Delta_{in}(t), L_p) =: d(\Delta_{in}(t), L_p)$ is a uni-modal function with maximum at $\Delta_{in}(t) = \Delta^*$, increasing function of $\Delta_{in}(t)$ when $\Delta_{in}(t) \leq \Delta^*$, and decreasing when $\Delta_{in}(t) \geq \Delta^*$, i.e.,

$$\frac{\partial d(\Delta_{in}(t), L_p)}{\partial \Delta_{in}(t)} \begin{cases} > 0, \ \Delta_{in}(t) < \Delta^*; \\ = 0, \ \Delta_{in}(t) = \Delta^*; \\ < 0, \ \Delta_{in}(t) > \Delta^*. \end{cases}$$

Finally, there is no difference between $U(\Delta_{in}(t), L_p)$ and $R(\Delta_{in}(t), L_p)$ when $\Delta_{in}(t) \leq L_p/C$ and the difference goes to zero as $\Delta_{in}(t) \to \infty$, i.e.,

$$d(\Delta_{in}(t), L_p) = 0, \text{ when } \Delta_{in}(t) \leq L_p/C, \tag{7}$$
$$d(\Delta_{in}(t), L_p) \to 0, \text{ as } \Delta_{in}(t) \to \infty. \tag{8}$$

It should be noted that $R(\Delta_{in}(t), L_p)$ depends on the statistical properties of cross traffic, even though not denoted explicitly, and cannot be obtained in an exact closed form unless the statistical properties of cross traffic is fully known a priori (which is practically impossible).

In order to obtain reliable information on the available bandwidth from the relation in (4), we should address the following two issues: (i) How can we define a robust metric of the available bandwidth, which can be exploited to estimate the available bandwidth? Note that the characteristic value Δ^* was exploited in the deterministic framework. (ii) If we can define a robust metric, then how can we devise a robust online algorithm to estimate the metric in spite of the measurement error $G(t, \Delta_{in}(t), L_p, \xi(t, \omega))$? Subsequent subsections provide solutions to these two issues.

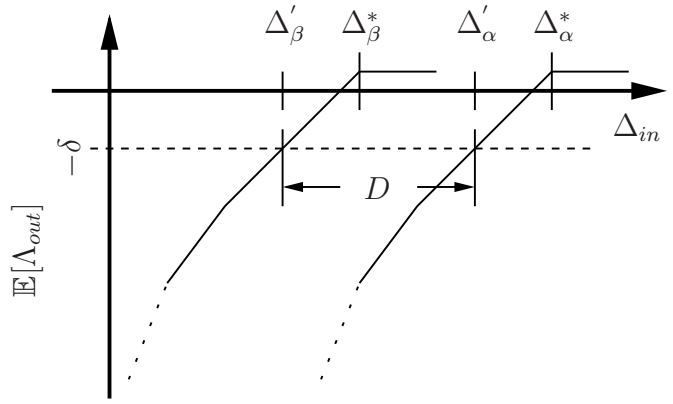


Fig. 3. Concept of the relative distance.

3.2 Concept of relative distance

We can observe from (4) that $\Lambda_{out}(t)$ is a function of two control variables Δ_{in} and L_p with an additive noise by bursty cross traffic. This investigation gives a new viewpoint on available bandwidth estimation, i.e., not only the relationship between $\Lambda_{out}(t)$ and Δ_{in}, but also the relationship between $\Lambda_{out}(t)$ and L_p can be exploited to obtain an efficient available bandwidth measurement algorithm. Note that most of the recent tools for the available bandwidth estimation paid attention only to the relationship between $\Lambda_{out}(t)$ and $\Delta_{in}(t)$. Here, the concept of the *relative distance* is introduced based on the relation between $\Lambda_{out}(t)$ and L_p. The relative distance provides us a robust metric of the available bandwidth.

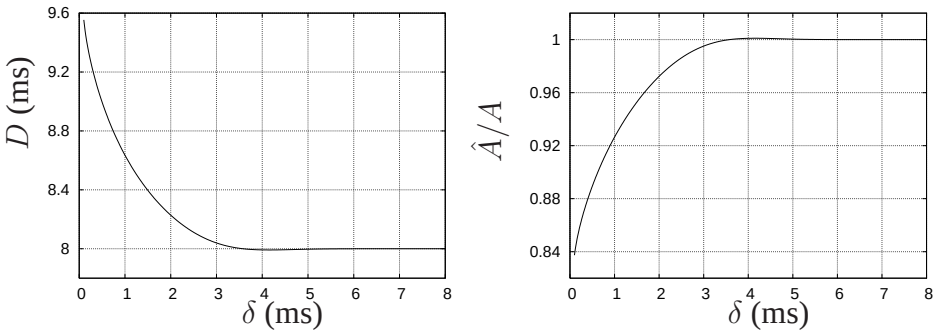
Let the s-th link denote the one with the minimum available bandwidth along the path $\mathcal{P}$. Think of two curves $R(\Delta_{in}(t), L_\alpha)$ and $R(\Delta_{in}(t), L_\beta)$ for probe packet sizes L_α and L_β, respectively. Two graphs can be drawn as in Fig. 3. For a given δ, consider an implicit function $\Delta_{in}(t) = H_\delta(L_p)$ such that $R(H_\delta(L_p), L_p) \equiv -\delta$. Then, the *relative distance* D is defined, as in Fig. 3, as the distance between the two curves in the x-axis direction as follows:

$$D = \int_{L_\beta}^{L_\alpha} \frac{\partial \Delta_{in}}{\partial L_p} dL_p = \int_{L_\beta}^{L_\alpha} \frac{\partial H_\delta(L_p)}{\partial L_p} dL_p.$$

Since it was given in (7) that there exists no error between $R(\Delta_{in}(t), L_p)$ and the deterministic relation, the relative distance D becomes constant when $\Delta_{in}(t) \leq L_p/C_s$. Consequently, a condition on δ can be derived, which makes the relative distance invariant to bursty cross traffic. Note that $\Lambda_{out} = -u_s L_p/C_s$ when $\Delta_{in} = L_p/C_s$. Hence, under the assumption of a single tight link, the following rule for setting the parameter δ can be derived to make the relative distance a constant value:

$$D = \frac{|L_\alpha - L_\beta|}{C_s(1-u_s)}, \text{ if } \delta \geq \frac{u \max(L_\alpha, L_\beta)}{C_s}. \tag{9}$$

Hereafter, as long as there is no concern for ambiguity, we omit the subscript s for



(a) Relative distance

(b) Normalized estimate of available bandwidth

Fig. 4. Relative distance and the corresponding estimate of available bandwidth under Poisson cross traffic.

notational simplicity.

As an illustrative example, we perform a simple simulation with Poisson cross traffic and see how the relative distance behaves. The relative distance with Poisson cross traffic of $L_\alpha = 1500$ B, $L_\beta = 500$ B, $L = 50$ B, $u = 0.5$, and $C = 1$ Mbps is shown in Fig. 4 (a). The estimate of the available bandwidth $A = |L_\alpha - L_\beta|/D$ is also given in Fig. 4 (b). From Fig. 4, we can validate that the relative distance is constant for $\delta \geq 0.5 \times 1500 \times 8/10^6 = 6$ ms. Note that δ can be set to a smaller value than this in practice (4 ms for example) because (9) is a sufficient condition.

From (9), one might think that the value of δ can be set to make the relative distance robust to the burstiness of cross traffic. However, (9) is not so much efficient in practice because information on u and C is not available to the end hosts. We will propose an alternative rule for setting δ and will give the convergence analysis of the proposed mechanism in the next section.

3.3 Feedback-assisted robust mechanism: bTrack

As already mentioned in (9), with $\delta \geq \frac{uL_p}{C}$, a constant value of the relative distance D can be estimated by searching the value, Δ'_{in}, of $\Delta_{in}(t)$ that satisfies $\Delta_{in}(t) - \Delta_{out}(t) + \delta = 0$ for two pairs of probe packets of different sizes, respectively. Figure 5 gives the block diagram of the proposed measurement mechanism. In every interval of T_p seconds, the sender transmits two pairs of probe packets with inter-departure times $\Delta^\alpha_{in}(t)$ and $\Delta^\beta_{in}(t)$ and packet sizes L_α and L_β ($L_\alpha > L_\beta$), respectively. The receiver measures the inter-arrival times $\Delta^\alpha_{out}(t)$ and $\Delta^\beta_{out}(t)$, of the two packet pairs, and calculates the inter-departure times, $\Delta^\alpha_{in}(t+T_p)$

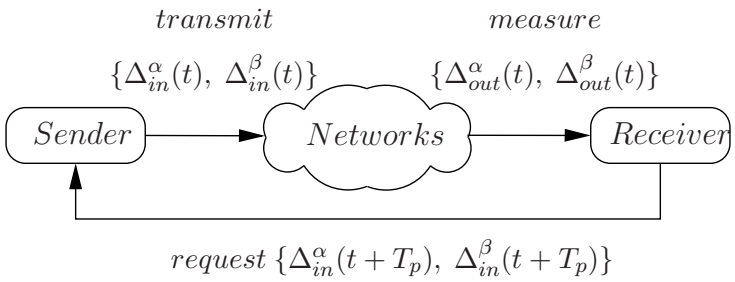


Fig. 5. The measurement mechanism with a simple feedback loop.

and $\Delta^{\beta}_{in}(t+T_p)$, to be used in the next probing period as follows:

$$\begin{aligned} \Delta_e(t) &= \Delta_{in}(t) - \Delta_{out}(t) + \delta, \\ \Delta_{in}(t+T_p) &= \Delta_{in}(t) - \gamma(t)\Delta_e(t), \end{aligned} \tag{10}$$

where $\gamma(t)$ is a positive updating gain at time t. Intuitively speaking, if $\Delta_e(t) < 0$, $\Delta_{in}(t+T_p)$ increases; otherwise, $\Delta_{in}(t+T_p)$ decreases. Hence, $\Delta_{in}(t)$ converges to a value Δ'_{in} that satisfies $\Delta_e(t) = 0$ in a steady state. A detailed pseudo-code for *bTrack* is given in Algorithm 1 and 2.

Now, we give a rigorous convergence analysis of the feedback algorithm (10) based on the theory of the stochastic approximation [21,22]. the measurement $\Delta_{out}(t)$ at each instant deviates from its expected value as given in (4), and this random noise from bursty cross traffic should be carefully considered in the convergence analysis of the proposed algorithm.

Since $\Delta_{in}(t) - \Delta_{out}(t) = \Lambda_{out}(t)$, (4) and (10) gives

$$\Delta_{in}(t+T_p) - \Delta_{in}(t) = -\gamma(t)\Big[\Big(R(\Delta_{in}(t), L_p) + \delta\Big) + G(t, \Delta_{in}(t), L_p, \xi(t,\omega))\Big]. \tag{11}$$

Instead of (9), which is not practical because of no information on C and u, we set δ by using $\delta = \kappa\Delta_{in}(t)$ where κ is a control parameter ($\kappa > 0$.) Then, (11) gives

$$\begin{aligned} \Delta_{in}(t+T_p) - \Delta_{in}(t) = -\gamma(t)\Big[\Big(&R(\Delta_{in}(t), L_p) \\ &+ \kappa\Delta_{in}(t)\Big) + G(t, \Delta_{in}(t), L_p, \xi(t,\omega))\Big]. \end{aligned} \tag{12}$$

Now, we have the following theorem:

Theorem 1 *The feedback mechanism (12) converges to a unique solution, denoted by Δ'_{in}, for a sequence of $\gamma(t)$ such that $\sum_{t=0}^{\infty}\gamma(t) = \infty$, $\sum_{t=0}^{\infty}\gamma^2(t) < \infty$. Furthermore, $\Delta'_{in} = \frac{1}{1-u+\kappa}\Big(\frac{L_p}{C}\Big)$ if $\kappa \geq u$.*

PROOF. We make use of Theorem 1.1 in [22]. Let $P(\Delta_{in}(t)) := -R(\Delta_{in}(t), L_p) - \kappa\Delta_{in}(t)$. First, we need to show that $P(\Delta_{in}(t)) = 0$ has a unique solution of Δ'_{in}.

Algorithm 1 *bTrack*::Sender

// initialize
$\Delta_{in}^{\alpha}(t) \leftarrow \Delta_{in}^{\alpha}(0)$
$\Delta_{in}^{\beta}(t) \leftarrow \Delta_{in}^{\beta}(0)$
loop
  // send the packet pair with the inter-departure time $\Delta_{in}^{\alpha}(t)$
  $t_1 \leftarrow$ now
  transmits the first packet with size L_{α}
  repeat
    $t_2 \leftarrow$ now
  until $(t_2 - t_1) \geq \Delta_{in}^{\alpha}(t)$
  transmit the second packet with size L_{α}
  sleep for $T_p/2$

  // send the packet pair with the inter-departure time $\Delta_{in}^{\beta}(t)$
  $t_1 \leftarrow$ now
  transmit the first packet with size L_{β}
  repeat
    $t_2 \leftarrow$ now
  until $(t_2 - t_1) \geq \Delta_{in}^{\beta}(t)$
  transmit the second packet with size L_{β}
  sleep for $T_p/2$

  // compute the available bandwidth
  $D(t) = \Delta_{in}^{\alpha}(t) - \Delta_{in}^{\beta}(t)$
  $A(t) = \frac{L_{\alpha} - L_{\beta}}{D(t)}$

  // wait for feedback from receiver and update the inter-departure time
  $\Delta_{in}^{\alpha}(t) \leftarrow \Delta_{in}^{\alpha}(t + T_p)$
  $\Delta_{in}^{\beta}(t) \leftarrow \Delta_{in}^{\beta}(t + T_p)$
end loop

We have $P(0) = \frac{L_p}{C} > 0$ and, since $R(\infty, L_p) = 0$ from (8), $P(\infty) = -\infty$. Also, $P(\Delta_{in}(t))$ is a decreasing function of $\Delta_{in}(t)$ from (5). Consequently, $P(\Delta_{in}(t))$ has a unique solution, which is denoted by Δ_{in}'.

Now, in order to show the convergence of (12), it is sufficient from Theorem 1.1 in [22] to show that the following two conditions are satisfied:

$$\sup_{\epsilon < |\Delta_{in} - \Delta_{in}^{'}| < 1/\epsilon} P(\Delta_{in}(t))(\Delta_{in}(t) - \Delta_{in}^{'}) < 0, \quad \text{for any } \epsilon, \qquad \textbf{(C1)}$$

and

$$P^2(\Delta_{in}(t)) \leq K(1 + \Delta_{in}^2(t)), \quad \text{for some constant } K. \qquad \textbf{(C2)}$$

First, we show that **(C1)** is satisfied. From the fact that $P(\Delta_{in}(t))$ is decreasing

Algorithm 2 *bTrack*::Receiver

loop
  // wait for the packet pair with the inter-departure time $\Delta_{in}^{\alpha}(t)$
  receive the first packet with size L_α
  $t_1 \leftarrow$ now
  receive the second packet with size L_α
  $t_2 \leftarrow$ now
  $\Delta_{out}^{\alpha}(t) \leftarrow (t_2 - t_1)$

  // wait for the packet pair with the inter-departure time $\Delta_{in}^{\beta}(t)$
  receive the first packet with size L_β
  $t_1 \leftarrow$ now
  receive the second packet with size L_β
  $t_2 \leftarrow$ now
  $\Delta_{out}^{\beta}(t) \leftarrow (t_2 - t_1)$

  // compute the next inter-departure times and feed back to sender
  $\Delta_{in}^{\alpha}(t + T_p) \leftarrow \Delta_{in}^{\alpha}(t) - \gamma(\Delta_{in}^{\alpha}(t) - \Delta_{out}^{\alpha}(t) + \delta)$
  $\Delta_{in}^{\beta}(t + T_p) \leftarrow \Delta_{in}^{\beta}(t) - \gamma(\Delta_{in}^{\beta}(t) - \Delta_{out}^{\beta}(t) + \delta)$
  feed back $\Delta_{in}^{\alpha}(t + T_p)$ and $\Delta_{in}^{\beta}(t + T_p)$
end loop

with $\Delta_{in}(t)$, together with the fact that $P(\Delta_{in}') = 0$, we have

$$\sup_{\epsilon<|\Delta_{in}-\Delta_{in}'|<1/\epsilon} P(\Delta_{in}(t))(\Delta_{in}(t) - \Delta_{in}') = \max\Big[-\epsilon P(\Delta_{in}' - \epsilon), \epsilon P(\Delta_{in}' + \epsilon)\Big] < 0. \tag{13}$$

Now, show that **(C2)** is satisfied. Since $P(\Delta_{in}(t)) = -R(\Delta_{in}(t), L_p) - \kappa\Delta_{in}(t)$ and $-\frac{L_p}{C} \leq R(\Delta_{in}(t), L_p) < 0$, we have

$$-\kappa\Delta_{in}(t) \leq P(\Delta_{in}(t)) \leq -\kappa\Delta_{in}(t) + \frac{L_p}{C}, \quad \text{for } \Delta_{in}(t) \in [0, \infty]. \tag{14}$$

(14) gives

$$P^2(\Delta_{in}(t)) \leq \max\left[\left(\frac{L_p}{C}\right)^2, \kappa^2\Delta_{in}^2(t)\right] \leq K(1 + \Delta_{in}^2(t)), \tag{15}$$

where $K = \max\left[\left(\frac{L_p}{C}\right)^2, \kappa^2\right]$. Here, $\gamma(t)$ is a sequence of positive numbers such that

$$\sum_{t=0}^{\infty}\gamma(t) = \infty, \quad \sum_{n=0}^{\infty}\gamma^2(t) < \infty. \tag{16}$$

Note that the condition (16) can be further weakened; for example, in some cases it is sufficient that $\sum_{t=0}^{\infty}\gamma^n(t) < \infty$ for some $n > 0$, instead of $\sum_{t=0}^{\infty}\gamma^2(t) < \infty$.

Further results can be found in [21,22]. From Theorem 1.1 in [22] with (13), (15), and (16), the process $\Delta_{in}^{\Delta_0}(n)$, defined by (12) with $\Delta_{in}(0) = \Delta_0$, converges with probability 1 as $t \to \infty$ to the root Δ'_{in} of $P(\Delta_{in}(t)) = 0$, i.e.,

$$\text{Prob}\{\lim_{t\to\infty} \Delta_{in}^{\Delta_0}(t) = \Delta'_{in}\} = 1. \tag{17}$$

Hence, from (17), the feedback algorithm (12) is guaranteed to converge to a unique solution Δ'_{in}. Finally, from (7), we have $R(\Delta_{in}(t), L_p) = (1-u)\Delta_{in}(t) - \frac{L_p}{C}$ if $\Delta_{in}(t) \le \frac{L_p}{C}$. Since Δ'_{in} is the solution of $P(\Delta_{in}(t)) = -R(\Delta_{in}(t), L_p) - \kappa\Delta_{in}(t) = 0$, $\Delta'_{in} = \frac{1}{1-u+\kappa}(\frac{L_p}{C})$ if $\kappa \ge u$.

Since the estimate of D, denoted by $\hat{D}$, is given by $\hat{D} = \Delta'_\alpha - \Delta'_\beta$, we have the following result from Theorem 1:

Corollary 1 *If $\delta = \kappa\Delta_{in}^{\beta}(t)$ is used for both $\Delta_{in}^{\alpha}(t)$ and $\Delta_{in}^{\beta}(t)$ and κ satisfies $(1-cu)\kappa \ge cu(1-u)$ where $c := \frac{L_\alpha}{L_\beta}$, the feedback algorithm (12) makes $\Delta_{in}^{\alpha}(t)$ and $\Delta_{in}^{\beta}(t)$ converge to $\Delta'_\alpha = \frac{1}{1-u+\kappa}\Big[c + \frac{\kappa(c-1)}{1-u}\Big]\Big(\frac{L_\beta}{C}\Big)$ and $\Delta'_\beta = \frac{1}{1-u+\kappa}\Big(\frac{L_\beta}{C}\Big)$, respectively. Furthermore, the estimate of the relative distance, $\hat{D}$, becomes a constant value of $\frac{L_\alpha - L_\beta}{C(1-u)}$, of which the denominator $C(1-u)$ corresponds to the available bandwidth.*

PROOF. From Theorem 1, $\Delta_{in}^{\beta}(t)$ converges to $\Delta'_\beta = \frac{1}{1-u+\kappa}\Big(\frac{L_\beta}{C}\Big)$ if $\kappa \ge u$. (It is straightforward to show that $(1-cu)\kappa \ge cu(1-u)$ implies $\kappa \ge u$.) Also, from (1), we have $(1-u)\Delta'_\alpha - \frac{L_\alpha}{C} = -\delta$ for $\delta \ge \frac{uL_\alpha}{C}$. By using $\delta = \kappa\Delta'_\beta = \frac{\kappa}{1-u+\kappa}\Big(\frac{L_\beta}{C}\Big)$, we get $\Delta'_\alpha = \frac{1}{1-u+\kappa}\Big[c + \frac{\kappa(c-1)}{1-u}\Big]\Big(\frac{L_\beta}{C}\Big)$ for κ such that $(1-cu)\kappa \ge cu(1-u)$. Finally, we have $\hat{D} = \Delta'_\alpha - \Delta'_\beta = \frac{L_\alpha - L_\beta}{C(1-u)}$.

As we have already mentioned in the previous subsection and also shown in Fig. 4, $\hat{D}$ converges to $\frac{L_\alpha - L_\beta}{C(1-u)}$ quite quickly as δ increases, and just a reasonable value of δ will be enough to estimate the available bandwidth in practice. We will verify the performance of *bTrack* via extensive simulation and empirical results in the subsequent section.

To convey to the sender the calculated values of $\Delta_{in}(t+T_p)$ to be used in the next probing period, the receiver sends a control packet that contains $\Delta_{in}^{\alpha}(t+T_p)$ and $\Delta_{in}^{\beta}(t+T_p)$. The sender will then transmit two pairs of probe packets with the suggested inter-departure times in the next probing period. Note that one should set $T_p > RTT$ so that the information can be relayed back to the sender by the next probing period. The bandwidth consumed by the probing traffic is $B_p = 2(L_\alpha + L_\beta)/T_p$, which is independent of the amount of the available bandwidth and can be kept low by selecting proper values of L_α, L_β, and T_p. We will do an

in-depth study on the effects of these parameters on the bandwidth overhead in Section 4.

3.4 Robustness of bTrack

The use of *bTrack* to determine the relative distance D and to infer the available bandwidth has the following critical advantages: The relative distance D can be estimated quite accurately without the knowledge of characteristic value, which is very difficult to estimate due to randomness in cross traffic, and hence the available bandwidth can be accurately estimated based on the relative distance. Another desirable property of *bTrack* is that it consists of a feedback loop, which can average out the effect of the measurement error. Thus, as long as the time average of the measurement error is negligible, *bTrack* will work properly. Only when there exists a bias in the measurement (e.g., the measurement value is always larger than the true value by some fixed amount), *bTrack* may not work very well.

4 Performance Evaluation

4.1 Simulation Results

To validate the design of *bTrack* and to evaluate its performance, we have implemented it in *ns-2* and conducted a simulation study in networks of various topologies. The parameters of *bTrack* used in the simulation study are listed in Table 1, unless otherwise specified. The initial inter-departure times are set to $\Delta_{in}^{\alpha}(0) = 3$ ms and $\Delta_{in}^{\beta}(0) = 1.5$ ms. Given the parameters in Table 1, we have B_p = 48 Kb/s. In particular, the reference bias δ is set to 10% of Δ_{in}^{α} in order to keep δ a reasonable value. Our extensive simulation and empirical results verify that this value of δ is large enough to keep track of the available bandwidth. We will also study the effects of varying L_{α}, T_p, and γ on the performance of the proposed mechanism.

4.1.1 Results under the two link network topology

The two-link network topology is depicted in Fig. 6. The network is composed of three drop-tail routers connected via two links. The link capacity and the propagation delay of each link are labeled in the figure. The link between each host and a router has a capacity of 10 Mb/s and a propagation delay drawn from the uniform distribution of $[10, 20]$ milliseconds. The buffer size of each of the drop-tail routers is set to 100 Kbytes. Each host is equipped with a Pareto source that sends packets at a rate of 32 Kb/second. The total, aggregated traffic is long-range dependent with

Table 1
Parameters used in the simulation study.

symbol	description	value
L_α	first probe packet size	1000 bytes
L_β	second probe packet size	$L_\alpha/2$
T_p	probing period	0.5 seconds
γ	updating gain	1% of Δ_{in}
δ	reference bias	10% of Δ_{in}

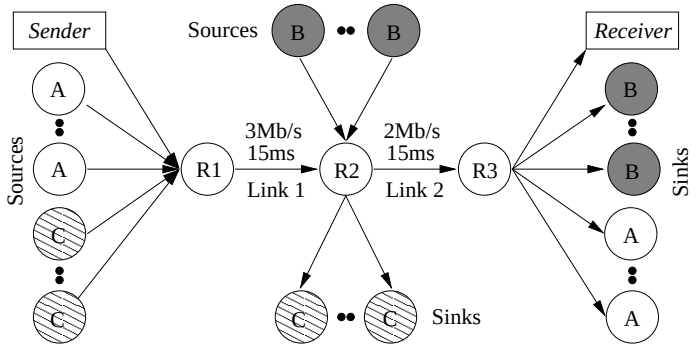


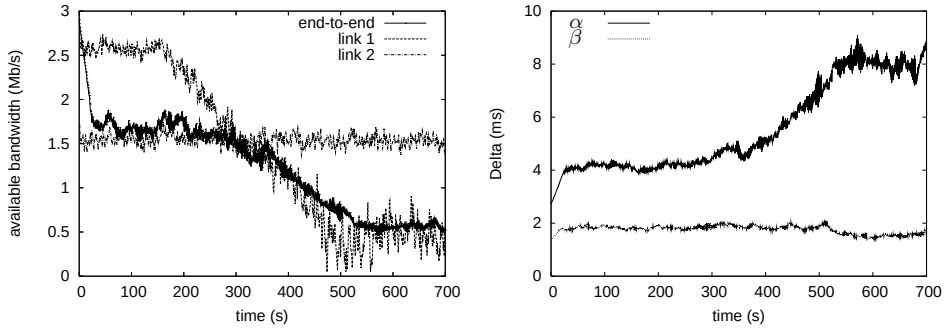
Fig. 6. The two link network topology used in Section 4.1.1.

the Hurst parameter H = 0.75. The distribution of the packet size is the same as that used in [1]: about 40% of the packets are of 40 bytes, 50% of 550 bytes, and the rests of 1500 bytes.

To simulate the situation in which the cross traffic at link 1 increases while that at link 2 is kept steady, we establish 3 connections of type A (totally with the sending rate of 96 Kb/s) and 10 connections of type B (totally with the sending rate of 320 Kb/s), but vary the number of connections of type C.

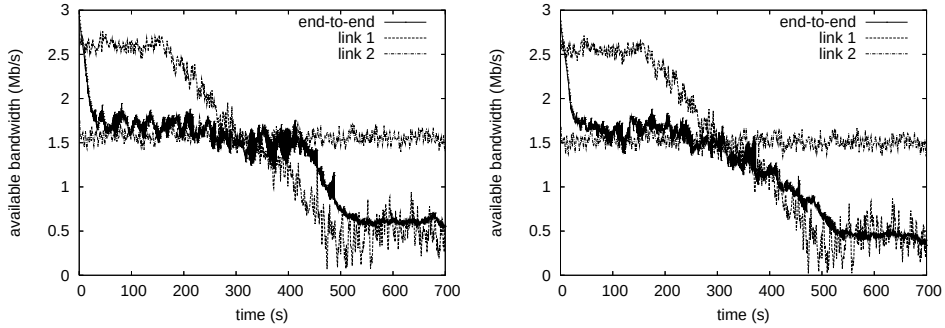
Figure 7 (a) shows that the available bandwidth at link 1 decreases from 2.6 Mb/s at $t = 150$ seconds to 0.5 Mb/s at $t = 500$ seconds. At $t = 300$ seconds, the link with the minimum available bandwidth changes from link 2 to link 1. The proposed mechanism keeps track of the end-to-end available bandwidth accurately except for $t < 50$ seconds, which correspond to 100 probing intervals. The discrepancy when $t < 50$ seconds is due to the fact that Δ_{in}^{α} and Δ_{in}^{β} are in the transient state (which is evidenced in Fig. 7 (b)). The duration of this transient state can be reduced by using either a smaller probing interval T_p or a larger updating gain γ.

Figure 8 depicts the effect of the probe packet size L_α on the measurement performance. The bandwidth, B_p, incurred in the measurement is linearly proportional to L_α. Hence if L_α is large, the probing traffic may affect the measurement intrusively, and result in a significant decrease in the available bandwidth. On the other hand, if L_α is small, the convergent value of D may be small, and its accuracy is susceptible



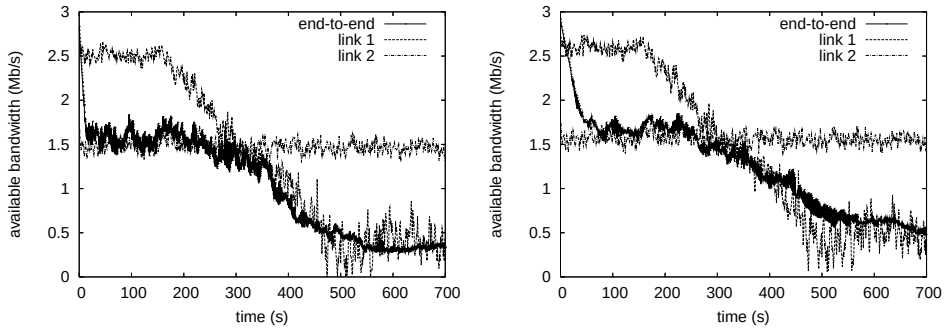
(a) available bandwidth (b) Δ_{in}^{α} and Δ_{in}^{β}

Fig. 7. Actual and measured available bandwidth under the two link network topology (L_α=1000 bytes, T_p=0.5 second, B_p=48 Kb/s, γ=1% of Δ_{in}).



(a) L_α=500 bytes (B_p=24 Kb/s) (b) L_α=1500 bytes (B_p=72 Kb/s)

Fig. 8. Effects of probing packet size L_α on measurement accuracy.

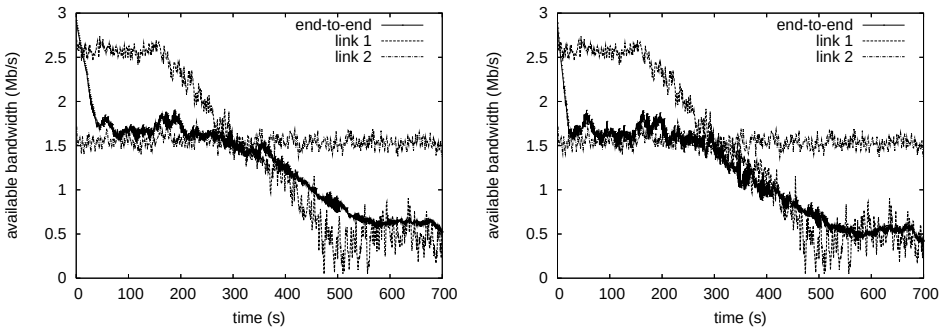


(a) T_p=0.25 second (B_p=96 Kb/s) (b) T_p=0.75 second (B_p=32 Kb/s)

Fig. 9. Effects of probing interval T_p on measurement accuracy.

to jitters or perturbations in the delay. This is evidenced by the observation that the end-to-end bandwidth measured in Fig. 8 (a) converges more slowly than that in Fig. 8 (b).

Figure 9 gives the effect of the probing period T_p on the measurement performance. The bandwidth, B_p, incurred in the measurement is inversely proportional to T_p. As shown in Fig. 9, if T_p is set to be 0.25 second, the available bandwidth is rapidly tracked (Fig. 9), but a bandwidth of 96 Kb/s is consumed for the measurement. On



(a) γ=0.5% of Δ_{in} (b) γ=1.5% of Δ_{in}

Fig. 10. Effects of updating gain γ on measurement accuracy.

the other hand, if T_p is set to be 0.75 second, the measurement slowly approaches the actual value with a larger time lag (Fig. 9 (b)). Roughly speaking, the measurement time of *bTrack* is proportional to the probing period T_p. Thus, the smaller T_p, the smaller the measurement time. In the meantime, there is a tradeoff between the measurement time and the intrusiveness: The probing period T_p and the bandwidth consumed by *bTrack*, B_p are inversely proportional, which can be verified in Figs. 8 and 9. Also, from the simulation results, about less than 100 iterations ($= 100T_p$) is sufficient to estimate the available bandwidth.

Figure 10 depicts the effect of the updating gain γ on the measurement performance. Recall that the default value for γ is 1% of Δ_{in} in Table 1. If γ is set to be smaller than the default value (γ=1% of Δ_{in}), it takes a relative long time for $\Delta_{in}(t)$ to converge from the initial values of $\Delta_{in}(0)$. On the other hand, if γ is to be 1.5% of Δ_{in}, the measured value of the available bandwidth converges quickly. Note that γ corresponds to the increase/decrease amount of the inter-departure times Δ_{in} in every probing interval. Through our simulation runs, we found that the proposed mechanism keeps track of the end-to-end available bandwidth robustly in the range of $\gamma = 0.1 \sim 5$% of Δ_{in}.

4.1.2 Results under the dumbbell network topology

The dumbbell network topology is depicted in Fig. 11. The network is composed of six drop-tail routers interconnected in the fashion shown in Fig. 11. The link capacity and the propagation delay of each link are labeled in the figure. 10 connections of type A, 10 connections of type B, and N connections of type C are established, where N varies from 0 to 100. Each source sends packets at a rate of 32 Kb/s. Probe packets are injected by a sender (labeled as "Sender" in Fig. 11) and traverse the path $R_1 \rightarrow R_4 \rightarrow R_5 \rightarrow R_6$ before they arrive at the receiver (labeled as "Receiver" in Fig. 11). The link with the minimum capacity on the path is the link between R_1 and R_4 (with a capacity of 4 Mb/s), and its available bandwidth is 4 Mb/s $-$ 32 Kb/s $\times$ 10 $-$ B_p = 3.584 Mb/s. (Recall that $B_p = 96$ Kb/s with the parameters in Table 1.) The available bandwidth of the link between R_5 and R_6 is

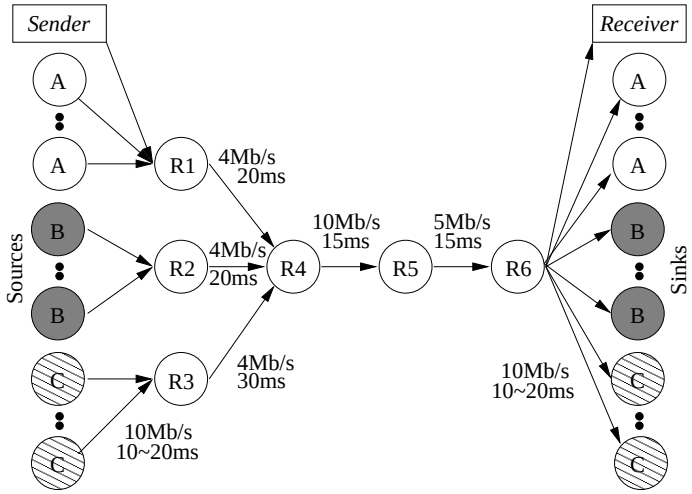


Fig. 11. The dumbbell network topology used in Section 4.1.2.

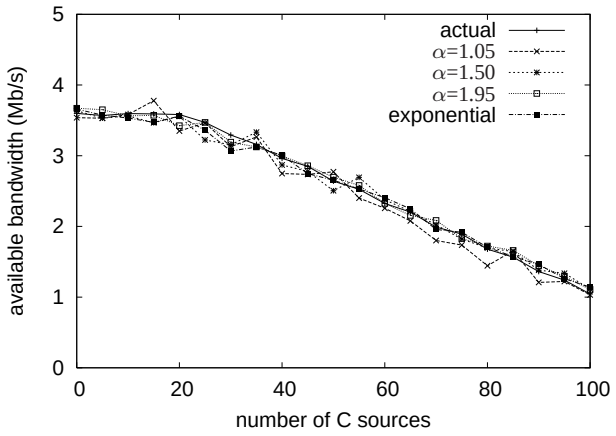


Fig. 12. Actual and measured available bandwidth under the dumbbell network topology (B_p=96 Kb/s).

given by 5 Mb/s $-$ $(N + 20) \times$ 32 Kb/s $-B_p$. As N increases, its utilization varies from 14.72% to 78.72%, and the link $R_5 \rightarrow R_6$ eventually becomes the link with the minimum available link bandwidth.

Figure 12 gives the simulation results when the sources are either Pareto sources with parameter $\alpha = 1.05$, 1.50, or 1.95, or sources that generate packets with exponential inter-packet generation times with a rate of 32 Kb/s in the above dumbbell network. Each simulation run lasts for 1200 seconds, and the data plotted in the figure are values averaged over the last 1000 seconds. If $N \leq 20$, the available bandwidth on the end-to-end path (3.584 Mb/s) is determined by the available bandwidth of the link between R_1 and R_4; otherwise, it is determined by the available bandwidth of the link between R_5 and R_6. The average measurement errors are 92.5 Kb/s (4.69%), 82.7 Kb/s (3.42%), 54.2 Kb/s (2.53%), and 64.0 Kb/s (2.89%) in each of the four cases in Fig. 12. In summary, the proposed mechanism accurately estimates the available bandwidth on the end-to-end path with a maximum

error of 5% for a wide spectrum of traffic patterns and a wide range of utilization.

4.2 Empirical Results

We have implemented the proposed mechanism on top of Linux 2.4.20 to measure the available bandwidth between two hosts on the Internet. The resulting prototype implementation is called *bTrack*. The reason why we implement the proposed method outside the OS kernel, in spite of the fact that both probe packets in a packet pair are subject to the OS processing latency, is because we are primarily interested in the difference of the departure/arrival times, and the effect of OS processing latency is canceled out in the average sense in *bTrack* if the expected OS latency is assumed to be the same. Still, note that there exist instances in practice, in which this assumption may not hold, e.g., the case when a deterministic context switch exists only for every other packet.

As shown in Algorithm 1 and 2, the sender of *bTrack* transmits probing traffic with different-sized packet pairs, and the receiver measures the the inter-arrival times of probing packet pairs. For measuring the inter-arrival and inter-departure time on Linux, we use the function of `gettimeofday()`, which enables us to measure time differences in the unit of microsecond. For generating fine-controlled packet pairs, the sender first sets a reference time t_1, sends the first probing packet, and measures the current time t_2 repeatedly within a while-loop statement. When the time elapse from the reference time (t_2–t_1) is equal to or greater than Δ_{in}, it immediately sends the second probing packet. Please note that there is a relation among the time granularity, the probe packet size, and the available bandwidth. For example, when the probe packet size is 1 KB, the available bandwidth should be smaller than 800 Mb/s in order that the probe gap is larger than 10 microseconds. Otherwise, the current time granularity of a microsecond may not be sufficient for estimating the available bandwidth.

Note that, in the while-loop statement, we do not include a sleep function because it does not guarantee an accurate time elapse on a non-realtime OS. In fact, a well-known measurement tool for the available bandwidth, *Pathload* [23,24], has also been implemented in this way. While this implementation makes it possible to build a measurement system with an off-the-shelf desktop, it may result in high CPU utilization. Related work that addresses this issue with a commercial OS can be found, e.g., in [25]. Also, a stopping rule is not shown in Algorithm 1 and 2 because we assume that *bTrack* continually estimates available bandwidth. Whenever needed, we can simply incorporate with *bTrack* a stopping rule, e.g., "Stop if $|\Delta_{in} - \Delta_{out} + \delta| < \epsilon$ for both packet pairs", where ϵ is a sufficiently small value.

We have conducted an empirical study and compared experimental results between *bTrack*, *Pathload* [23], *IGI*, and *PTR* [15]. Each mechanism reports the estima-

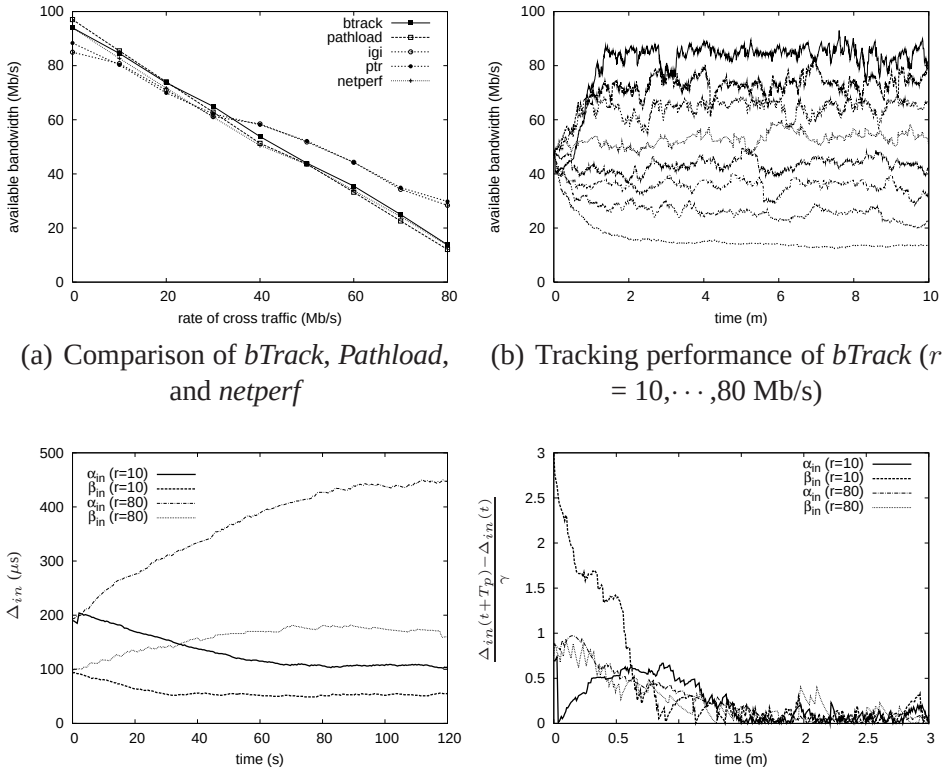
tion result in a different manner: *IGI* and *PTR* give an estimate of the available bandwidth, *bTrack* creates a time trace of the available bandwidth, while *Pathload* provides a range of available bandwidth [23]. (For consistency, we use the average of the range as an estimate of the available bandwidth in *Pathload*). For comparison purposes, we also use a benchmark program *netperf* [26]. *Netperf* attempts to transmit as many UDP (or TCP) packets as possible in a given time interval, and measures the maximum throughput. The TCP measurement gives a good estimate of the available bandwidth because a TCP connection are responsive and self-adaptive to cross traffic. However, it may nevertheless compete against, and grasp the bandwidth of, other connections along the path. Therefore, we consider the available bandwidth to be equal to or less than the TCP throughput measured by *netperf*. The only exception when the TCP measurement is not indicative of available bandwidth occurs when two end hosts are located far from the other. In this case, the attainable TCP throughput is limited by the maximum window size and the round trip time and does not reflect well the available bandwidth.

The parameter values listed in Table 1 are used as default values in *bTrack* except the following changes. The probe packet size L_α is set to 1200 bytes. To expedite convergence of Δ_{in}^*, we set the initial inter-departure time, $\Delta_{in}(0)$, as follows: $\Delta_\alpha^* = L_\alpha / C_s(1-u_s)$, where $C_s(1-u_s)$ is the available bandwidth on the end-to-end path. Hence we set $\Delta_{in}(0)$ to be 0.2 millisecond with a rough estimate of $C_s(1-u_s)$ = 50 Mb/s. Note that the initial value of Δ_{in} does not affect measurement accuracy, but can effectively reduce the convergence time should the value be properly chosen. We set the value of T_p to be 0.5 second, and the bandwidth consumed by *bTrack* is 57.6 Kb/s.

4.2.1 Empirical results in a campus intranet

We first measure the available bandwidth between two hosts on a campus network. Each host is a Dell Precision 340 machine (with a single 2.2 GHz processor and 1 GByte RAM) that runs Redhat Linux 8.0 with kernel version 2.4.9-34, and is connected to the campus backbone through a 100 Mb/s Ethernet link. The path consists of 4 hops with a RTT = 0.488 millisecond. To generate cross traffic on the network, we develop a traffic generating program *netloader*, that transmits UDP packets at a given rate. This gives a well-controlled intranet environment for evaluating and experimenting with *bTrack*, *Pathload*, *IGI*, *PTR*, and *netperf*.

Figure 13 and Table 3 give the empirical results of *bTrack*, *Pathload*, *IGI*, *PTR*, and *netperf*. Several observations are made: first, without cross traffic over the Ethernet link of capacity 100 Mb/s *bTrack*, *Pathload*, *IGI*, *PTR*, and *netperf* give estimates of 93.96, 97.04, 84.96, 88.36, and 94.0 respectively. The estimates obtained by *bTrack* are values averaged over the last 8 minutes. If the throughput obtained by *netperf* is considered to be the effective link capacity (i.e., C_s = 94.0 Mb/s), then *bTrack* gives the estimate that is closest to the value obtained by *netperf*. *Pathload* and *PTR* give



(a) Comparison of *bTrack*, *Pathload*, and *netperf*

(b) Tracking performance of *bTrack* (r = 10,$\cdots$,80 Mb/s)

(c) Δ_{in}^{α} and Δ_{in}^{β} for $r = 10$ and 80 Mb/s

(d) Stopping rule for $r = 10$ and 80 Mb/s

Fig. 13. Experimental results in a campus intranet.

slightly higher and lower estimates respectively. *IGI* reports 88.32 and 84.96 Mb/s as the bottleneck and available bandwidths, respectively. Second, when the rate of cross traffic is in the range of 10 $\sim$ 80 Mb/s, *bTrack*, *Pathload*, and *netperf* give almost the same estimate, which linearly decreases with respect to the rate of the cross traffic. *IGI* and *PTR* over-estimate the available bandwidth in the case of high link utilization ($r \geq 40$ Mb/s).

Also shown in Table 3 are the percentage errors calculated as

$$P.E. = \left| 1 - \frac{B_a}{C_s - r_s} \right| \times 100,$$

where B_a and r_s are the estimate of the available bandwidth and the rate of the cross traffic, respectively. We observe that *bTrack* accurately estimates the available bandwidth with a maximum error of 4.43% when the available bandwidth changes in the range of 13.80 $\sim$ 93.96 Mb/s. In contrast, *Pathload* incurs a maximum error of 13.67% in the same range of available bandwidth change.

Figure 13 (b) shows the time trace of available bandwidth estimated by *bTrack* with respect to different rates of cross traffic. The initial value of the estimate is set to be 50 Mb/s. When $t > 50$ seconds, the estimate of the available bandwidth converges. Also, we observe that the fluctuation in the estimates decreases as the rate of cross

Table 2
Measurement time (in seconds) with respect to ϵ.

ϵ	rate of cross traffic (Mb/s)							
	10	20	30	40	50	60	70	80
0.3	71.2	52.0	45.8	30.7	9.8	21.3	42.2	67.6
0.4	70.2	37.9	42.7	29.2	6.2	15.1	35.4	43.6
0.5	58.2	35.3	31.3	22.9	4.1	10.9	22.9	31.7

Table 3
Experimental results in a campus intranet (the unit of data is Mb/s and the percentage values in the parentheses denote the percentage error of the corresponding estimate).

cross traffic	netperf	bTrack		Pathload	IGI	PTR
		mean	std			
0	94.0 (0%)	93.96 (0.03%)	3.57	97.04 (3.23%)	84.96 (9.61%)	88.36 (6.00%)
10	82.7 (1.55%)	84.51 (0.61%)	2.80	85.47 (1.75%)	80.75 (3.86%)	80.26 (4.45%)
20	71.7 (3.11%)	73.65 (0.46%)	3.10	74.03 (0.04%)	70.97 (4.09%)	69.92 (5.51%)
30	61.0 (4.69%)	64.97 (1.52%)	2.49	62.83 (1.82%)	61.17 (4.42%)	62.22 (2.78%)
40	50.5 (6.48%)	53.78 (0.39%)	2.27	51.28 (5.02%)	58.48 (8.29%)	58.14 (7.66%)
50	43.5 (1.14%)	43.98 (0.03%)	2.14	43.62 (0.85%)	51.79 (17.70%)	52.16 (18.54%)
60	33.9 (0.29%)	35.50 (4.43%)	2.50	33.24 (2.22%)	44.30 (30.29%)	44.23 (30.08%)
70	24.2 (0.83%)	24.97 (4.04%)	2.05	22.60 (5.81%)	34.29 (42.87%)	34.91 (45.45%)
80	13.6 (2.86%)	13.80 (1.39%)	0.67	12.08 (13.67%)	28.38 (102.71%)	29.81 (112.92%)

traffic increases. The standard deviation decreases from 3.57 Mb/s when there in no cross traffic to 0.67 Mb/s when the rate of cross traffic is 80 Mb/s. The standard deviation calculated in each case is smaller than 8% of the corresponding available bandwidth. This suggests that *bTrack* is able to estimate the available bandwidth with a small variance. Figure 13 (c) illustrates how the packet pair gaps converge when the rate of cross traffic r is 10 Mb/s and 80 Mbp/s. Initially, Δ_{in}^{α} and Δ_{in}^{β} are set to 0.2 ms and 0.1 ms, respectively. We can observe that the gap values converges faster when $r = 10$ Mb/s than those when $r = 80$ Mb/s. The reason is that the initial gap values are closer to the actual values used for available bandwidth when $r = 10$ Mb/s than when $r = 10$ Mb/s. Thus, we can conclude that the initial gap values also affect the convergence speed of the algorithm.

4.2.2 Empirical results on the Internet

Next we report our Internet experiments measured between end hosts at Rice University and the University of Illinois at Urbana Champaign (UIUC). With the use of *traceroute*, we know that probe packets traverse 13 hops along the path from Rice to UIUC. As the UDP throughput measured by *netperf* is 89.81 Mb/s, we believe each host is connected through a 100 Mb/s Ethernet link. The TCP throughput is almost constantly 15.2 Mb/s, and seems to be constrained by the maximum window size, the RTT, and the dropping probability. Hence we believe that the available bandwidth in the experiment duration is quite stable and is in the range between the

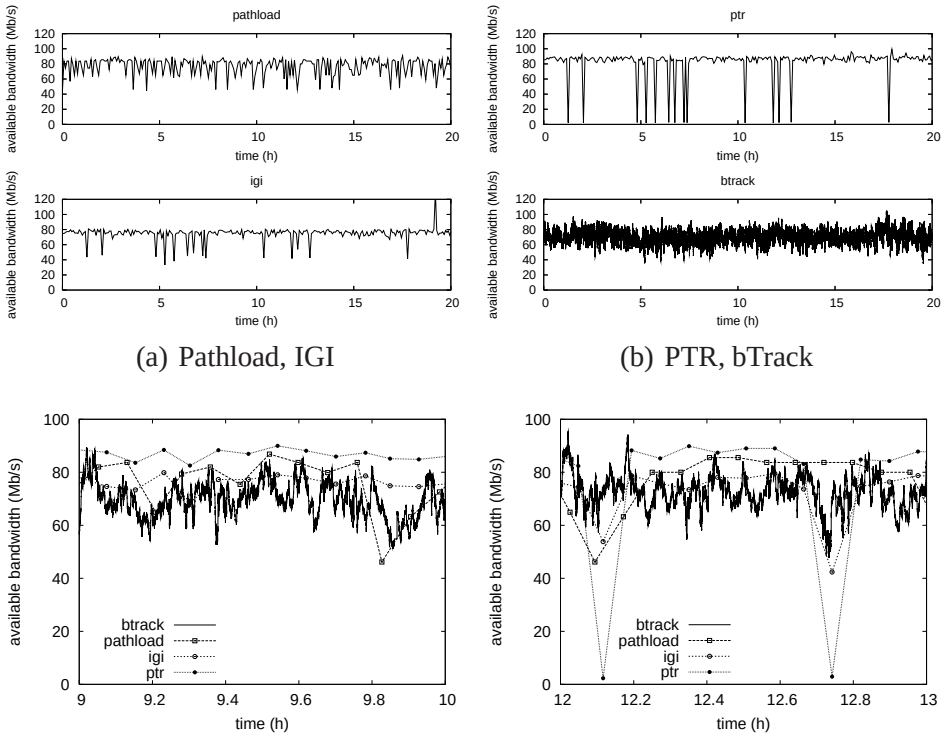
TCP and UDP throughputs attained by *netperf*. To further verify this, we measure (with the use of *netloader*) the packet loss rate between the hosts as the sending rate of UDP connections increases. The packet loss rate on the three paths falls below 1% when the rate of cross traffic is lower than about 70 Mb/s. This implies that the available bandwidth is equal to or greater than 70 Mb/s. It should be noted that, since the path is not dedicated to our experiment, it is practically not allowed to make the entire end-to-end path crowded with *netperf* packets. Hence, we estimate the range of the available bandwidth as above mentioned, instead of finding an exact value of the available bandwidth.

Figure 14 depicts the time traces of the available bandwidth for the Internet experiments. *Pathload*, *IGI*, and *PTR* were repeatedly executed in every 5 minutes for a duration of 20 hours while *bTrack* ran continuously. The patterns in which probe traffic is sent under *bTrack* is quite different from that under the other schemes. The traffic used by *bTrack* is extremely low (57.6 Kb/s), and is evenly spread over the measurement duration. However, the other mechanisms generates bursty traffic at a rate that is equal or close to the available bandwidth for tens of seconds. We observe that the available bandwidth fluctuates in the range of $60 \sim 85$ Mb/s. The average value of the available bandwidth under *bTrack* is 70.66 Mb/s, which coincides with the rate at which packet losses fall below 1%, while those under *Pathload*, *IGI*, and *PTR* are 75.694, 75.91, and 85.21 Mb/s, respectively. Figure 14 (c) and (d) give the enlarged views in the intervals of 9 to 10 and 12 to 13 hours, respectively. We observe that the trend of the available bandwidth measured by *bTrack* matches that of either *Pathload* or *IGI*. For example, pronounced decreases in the available bandwidth are observed at 9.8 h in Fig. 14 (c) and 12.75 h in Fig. 14 (d). This shows that *bTrack* adaptively keeps track of the available bandwidth.

5 Related work

5.1 Methods for measuring bottleneck bandwidth

Several packet pair models and methods have been proposed for measuring the bottleneck bandwidth. Jacobson developed the *pathchar* tool, using the one-packet delay model to measure the link bandwidth on a path [6,5]. Succinctly, *pathchar* sends probe packets with varying packet sizes, and measures the round trip times of probe packets. The link to be measured is controlled by properly selecting the time-to-live (TTL) value in the IP headers of probe packets. The underlying packet model is ”RTT = packet size/bandwidth + queueing delay”. To eliminate the queueing delay term, *pathchar* sends, for each fixed packet size, a large number of packets and takes the smallest value of RTT as the one that incurs no queueing delay. It then repeats the measurement process for different packet sizes.



(a) Pathload, IGI (b) PTR, bTrack

(c) Probing interval (9-10 h) (d) Probing interval (12-13 h)

Fig. 14. Experimental results on the Internet (RICE → UIUC).

The packet pair technique [10] is widely used for measuring the bottleneck bandwidth on a path. Under the assumptions that (**A1**) the inter-departure time is small, (**A2**) no cross traffic exists on the bottleneck link, and (**A3**) the first probe packet in a packet pair does not queue after any other packets at or after traversing the bottleneck and the second probe packet does not queue at or before traversing the bottleneck, one can anticipate the two probe packets queue only at the bottleneck link, and hence the inter-arrival time of the packet pair at the receiver is equal to the packet size of the second probe packet divided by the bandwidth of the bottleneck link. Note that under assumptions (**A1**) and (**A2**), $\Delta_{in} \leq \Delta^*$ and $r \approx 0$, and hence (1) reduces to $\Delta_{out} \approx L/C$. As mentioned in [27], the packet pair technique primarily suffers from two problems in practice: time compression and time extension. This is due to the fact that assumptions (**A2**) and (**A3**) do not hold in practice. Specifically, if the first probe packet encounters queueing when it traverses the bottleneck link, the "distance" between the two probe packets is time-compressed, i.e., $\Delta_{out} = \Delta_{in} - q/C < \Delta_{in}$. On the other hand, if there exists cross traffic between the two probe packets at or before the bottleneck link, the distance between the packet pair will be time-extended, i.e., $\Delta_{out} = r\Delta_{in}/C + L/C \geq \Delta_{in}$.

To remedy the above problems, Carter *et al.* [8] processed the measured data using union and intersection operations, Lai *et al.* [9] used a kernel density estimator algorithm, and Paxson [10] and Dovrolis *et al.* [12] exploited multi-modalities observed in histograms of bandwidth measurements to better estimate the link bandwidth.

Several promising packet-pair-based methods were recently proposed to measure the available bandwidth on an end-to-end path. They can be categorized into those that are based on the single hop model and those that are based on the multiple hop model.

Methods based on the single hop model Methods in the first category [14,15,13] assume that (**A4**) there exists a single bottleneck link on the end-to-end path, and (**A5**) the link capacity of the bottleneck link C is given *a priori*. Under these assumptions, the following *packet pair formula* has been commonly used for measuring cross traffic:

$$r = \frac{C\Delta_{out} - L}{\Delta_{in}}. \tag{18}$$

The available bandwidth is obtained by subtracting the calculated rate, r, of cross traffic from the capacity, C, of the bottleneck link. Note that (18) is equivalent to the case of a fully utilized link in (1) and assumptions (**A1**) ($\Delta_{in} \leq \Delta^*$) and (**A3**) are implicitly assumed in the equation.

Ribeiro *et al.* [14] proposed an end-to-end approach, called *Delphi*, to measure cross traffic. The inter-departure time Δ_{in} is initially set to be L/C in order to ensure that the queue does not become empty between arrivals of two consecutive probe packets. Then in order not to over-consume the bandwidth and disturb other network traffic, temporally-spaced probe packet trains (called "chirp packet trains") are sent in which the inter-departure time for two consecutive probe packets is increased by a factor of two. The rate of cross traffic is inferred based on the multifractal model that characterizes the long range dependence of Internet traffic.

Hu *et al.* [15] derived a packet train formula for a single bottleneck link by replacing Δ_{in} and Δ_{out} in the packet pair formula in (18) with the summations of Δ_{in} and $\Delta_{out}^{'}$, respectively, where $\Delta_{out}^{'}$ is the inter-arrival time satisfying $\Delta_{in} < \Delta_{out}$. They then proposed the initial gap increasing (*IGI*) method to characterize the available bandwidth on a network path. *IGI* attempts to seek a breakpoint Δ_p at which the inter-departure time at the sender is equal to the inter-arrival time at the receiver by increasing the inter-departure time. At the breakpoint, i.e, $\Delta_{in} = \Delta_p$, *IGI* evaluates the rate of cross traffic using the packet train formula. Similarly, they proposed another algorithm called the packet transmission rate method (*PTR*), which estimates the available bandwidth by dividing the summation of Δ_{out} into the total packet length of a packet train at the breakpoint.

Strauss *et al.* [13] proposed a simpler method, called *Spruce*, to measure the available bandwidth. *Spruce* transmits a sequence of packet pairs instead of packet trains and averages instantaneous rate estimates obtained by each packet pair with

Table 4
Comparison between tools measuring available bandwidth

	single hop model	multiple hop model
packet pair	*Spruce* [13]	*TOPP* [11], *bTrack* (proposed)
sparse packet train	Delphi [14]	*pathChirp* [16]
packet train	*IGI* [15]	*Pathload* [1], *PTR* [15]

$\Delta_{in} = L/C$ based on the packet pair formula. The interval between two packet pairs follows the Poisson distribution.

Unfortunately, *Delphi* [14], *IGI* [15], and *Spruce* [13] give an accurate estimate of the available bandwidth only when the bottleneck link (*narrow link*) coincides with the link of the minimum available bandwidth along the path (*tight link*).

Methods based on the multiple hop model Melander *et al.* [11] developed an end-to-end method, called *TOPP*, to measure the link capacity and the available bandwidth along a path. Under the assumption that all the flows obtain their share of link bandwidth proportional to their offered rates, the sender of *TOPP* transmits N packet pairs at certain rates, and the relation between the sending rate and the receiving rate is analyzed based on a segmented regression method. The regression method works well when the breakpoint of each segment is known. However, under most cases, it is difficult to obtain these breakpoints and apply the regression method.

Jain *et al.* [1] showed that the one-way delay of a packet train increases if the sending rate of the packet train is higher than the available bandwidth. They proposed a feedback-based mechanism, called *Pathload*, which enables the probing sender to cooperate with the receiver to determine the available bandwidth. As *Pathload* is primarily using a binary-search algorithm that does not continuously estimate the available bandwidth, it is not adaptive to traffic changes. If the available bandwidth varies during the course of running the binary-search algorithm, *Pathload* easily falls into an ambiguous region (grey-region) and cannot estimate the available bandwidth accurately. Another drawback of *Pathload* is that it sends packet trains at a rate that is equal or close to the available bandwidth and hence may be intrusive, even though the duration of each packet train is short.

Summary The aforementioned mechanisms for measuring the available bandwidth are summarized in Table 4. Delphi, IGI, and Spruce consider the end-to-end path as a single hop link and do not give an accurate estimate of the available bandwidth when the bottleneck link (*narrow link*) is not equal to the link with the minimum available bandwidth along the path (*tight link*). *Pathload*, *IGI*, and *PTR* use probe packet trains and may be intrusive, as the packet trains may eventually

consume the available bandwidth, even though the duration of each packet train is short. *Spruce* and *TOPP* are based on the packet pair technique, can adjust the sending rate and pattern of probe packets by varying the time interval between two consecutive packet pairs, and hence may be less intrusive.

In contrast, we characterize the stochastic relation between the inter-departure time and the inter-arrival time for a packet pair along a multiple-link path. We then exploit the notion of the relative distance to overcome the random estimation error caused by bursty cross traffic, and devise our proposed end-to-end measurement method, *bTrack*. *bTrack* periodically transmits only two packet pairs in a relatively large time interval, the period of which is adjustable so as not to disturb other network traffic.

6 Conclusion

In this paper, we have given a stochastic analysis of the packet-pair technique for estimation of available bandwidth. Based on the analysis, we have proposed a novel concept of the relative distance, which gives accurate information on the available bandwidth under bursty cross traffic. By exploiting the relative distance, we have devised an end-to-end, non-intrusive feedback mechanism, entitled *bTrack*, to keep track of the available bandwidth. We have also given the convergence analysis of *bTrack* based on the theory of the stochastic approximation. We have shown via extensive *ns-2* simulation and empirical experiments (over campus intranets and the Internet) that the proposed mechanism tracks available bandwidth quite well with a small variance (less than 8% of the value to be tracked), and is non-intrusive (i.e., the bandwidth consumed by the probing traffic is extremely low).

Our future work includes a further investigation on (i) how to determine the tunable parameters of *bTrack* for fast and stable estimation of available bandwidth; and (ii) how to further expedite the convergence of the estimation process so as to eliminate initial transient errors. Finally, we are collaborating with Telecordia Technologies to incorporate *bTrack* in their measurement framework and carry out large-scale measurement on the Internet.

References

[1] M. Jain, C. Dovrolis, End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput, in: Proceedings of ACM SIGCOMM, 2002.

[2] E. Ng, H. Zhang, Predicting Internet network distance with coordinates-based approaches, in: Proceedings of IEEE INFOCOM, 2002.

[3] S. Ratnasamy, M. Hadley, R. Karp, S. Shenker, Topologically-aware overlay construction and server selection, in: Proceedings of IEEE INFOCOM, 2002.

[4] K. M. Hanna, N. Natarajan, B. N. Levine, Evaluation of a novel two-step server selection metric, in: Proceedings of IEEE International Conference on Network Protocol (ICNP), 2002.

[5] A. B. Downey, Using patchar to estimate Internet link characteristics, in: Proceedings of IEEE ACM SIGCOMM, 1999.

[6] V. Jacobson, Pathchar: A tool to infer characteristics of Internet paths, `ftp://ftp.ee.lbl.gov/pathchar` (1997).

[7] J. C. Bolot, Characterizing end-to-end packet delay and loss in the Internet, in: Proceedings of ACM SIGCOMM, 1993, pp. 289 – 298.

[8] R. L. Carter, M. E. Crovella, Measuring bottleneck link speed in packet-switched networks, Performance Evaluation 27/28 (1996) 297 – 318.

[9] K. Lai, M. Baker, Measuring bandwidth, in: Proceedings of IEEE INFOCOM, 1999, pp. 235 – 245.

[10] V. Paxson, End-to-end Internet packet dynamics, IEEE/ACM Transactions on Networking 7 (3) (1999) 277–292.

[11] B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estiamting bandwidth bokttlenecks, in: Proceedings of IEEE Global Internet Symposium, 2000.

[12] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure?, in: Proceedings of IEEE INFOCOM, 2001, pp. 905 – 914.

[13] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of ACM Internet Measurement Conference (IMC), 2003.

[14] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, R. Baraniuk, Multifractal cross-traffic estimation, in: Proceedings of ITC specialist seminar on IP traffic measurement, modeling and management, 2000.

[15] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal of Selected Areas in Communications 21 (6) (2003) 879 –894.

[16] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, L. Cottrell, Pathchirp: Efficient available bandwidth estimation for network paths, in: Proceedings of Passive and Active Measurement Workshop (PAM), 2003.

[17] X. Liu, K. Ravindran, B. Liu, D. Loguinov, Single-hop probing asymptotics in available bandwidth estimation: Sample-path analysis, in: Proceedings of ACM Internet Measurement Conference (IMC), 2004.

[18] X. Liu, K. Ravindran, D. Loguinov, What signals do packet-pair dispersions carry?, in: Proceedings of IEEE INFOCOM, 2005.

[19] P. Hága, K. Diriczi, G. Vattay, I. Csabai, Understanding packet pair separation beyond the fluid model: The key role of traffic granularity, in: Proceedings of IEEE INFOCOM, 2006.

[20] K.-J. Park, H. Lim, C.-H. Choi, Stochastic analysis of packet-pair probing for network bandwidth estimation, Elsevier Computer Networks, Special Issue on "Network Modeling and Simulation" 50 (12) (2006) 1901–1915.

[21] H. J. Kushner, G. Yin, Stochastic Approximation and Recursive Algorithms and Applications, Springer, 2003.

[22] M. B. Nevelson, R. Z. Khasminskii, Stochastic Approximation and Recursive Estimation, American Mathematical Society, Providence, RI, 1976.

[23] M. Jain, C. Dovrolis, Pathload: A measurement tool for end-to-end available bandwidth, in: Proceedings of Passive and Aciteve Measurements (PAM) workshop, 2002.

[24] Pathload, `http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathload.html`.

[25] A. Goel, L. Abeni, C. Krasic, J. Snow, J. Walpole, Supporting time-sensitive applications on a commodity OS, in: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), 2002.

[26] Netperf: A network performance benchmark, `http://www.netperf.org`.

[27] K. Lai, M. Baker, Measuring link bandwidths using a deterministic model of packet delay, in: Proceedings of ACM SIGCOMM, 2000, pp. 283–294.