Quick and Simultaneous Estimation of Available Bandwidth and Effective UDP Throughput for Real-Time Communication

Takashi Oshiba and Kazuaki Nakajima
Service Platforms Research Laboratories, NEC Corporation, Japan
{oshiba@cp, nakajima@ah}.jp.nec.com

***Abstract*—We propose a method, called PathQuick2, for quickly and simultaneously estimating both of the *available bandwidth* that is unused capacity of an end-to-end path and the *effective UDP throughput* that is the receiving rate of an UDP flow which can pass through the path, with a single measurement. In PathQuick2, a sender transmits a probing packet train (i.e., a set of multiple probing packets) that each packet is placed at an equal time interval, and each packet size increases as the packet sequence proceeds. A receiver produces the estimated available bandwidth and the estimated effective UDP throughput at the same time from the single packet train. To this end, the receiver detects a packet at which the observed time intervals begin increasing, and the per-packet receiving rate becomes the estimated available bandwidth. Then, the receiver detects a packet at which the observed per-packet receiving rates stop increasing even if the packet sequence proceeds, and the per-packet receiving rate becomes the estimated effective UDP throughput. Our evaluation of PathQuick2 has shown that its estimation duration is only 182 ms, and its probing load is about 90 kB which is more than 160 times as light as a direct measurement of effective UDP throughput. PathQuick2 can provide two useful metrics for real-time communication applications such as video chat and video conferencing; the available bandwidth indicates an upper limit of video sending rate which enables stable video communication without packet loss, and the effective UDP throughput indicates an upper limit of video receiving rate that can pass through an end-to-end path, with a given video sending rate.**

Keywords: available bandwidth, effective UDP throughput

I. INTRODUCTION

Real-time communication over IP networks such as video chat [1] and video conferencing [2] have gained in popularity in recent years. Since these applications have strict delay requirements and the retransmission mechanism of TCP does not fit the requirements, most of these applications use UDP packets for video transmission [3]. In addition to the delay, the *available bandwidth* (i.e., physical capacity minus bandwidth being used during a certain time period [4]) has a great influence on these applications. Unfortunately, it is reported that the available bandwidth is often insufficient for these applications [5]. When the available bandwidth is insufficient, it is reported that Skype, a major video chat application, doubles its video sending rate by using FEC coding to counteract packet losses [1]. Windows Live Messenger, another major video chat application, behaves similarly [3]. Knowledge of an upper limit of video receiving rate that can pass through an end-to-end path, with a given video sending rate, can improve this greedy behavior of the applications. The reason is that if the doubled video sending rate exceeds the upper limit, the exceeded UDP packets are surely dropped at some router. So, with the knowledge of the upper limit, the waste of bandwidth can be avoided by restricting the video sending rate under the upper limit (we discuss this issue in Section VI).

In this paper, we propose a method, PathQuick2, to quickly and simultaneously estimate both of the available bandwidth and the *effective UDP throughput* (the definition is given in Section II) of an end-to-end path with a single measurement. PathQuick2 is the successor to our quick available bandwidth estimation method, PathQuick [6]. In PathQuick2, a sender transmits a probing packet train (i.e., a set of multiple probing packets), and a receiver produces the estimated available bandwidth and the estimated effective UDP throughput at the same time from the single packet train. These two metrics are useful for the real-time communication applications; the available bandwidth indicates an upper limit of video sending rate which enables stable video communication without packet loss, and the effective UDP throughput indicates an upper limit of video receiving rate that can pass through the end-to-end path, with a given video sending rate.

Conventional available bandwidth measurement methods [4] and effective UDP throughput measurement methods have a critical restriction in that they require a long measurement time. Using these methods for real-time communication would cause a degradation of real-time responsiveness. Therefore, they are not suitable for real-time communication.

The main contributions of this paper are summarized as follows:

- We propose the *probe slope model* (PSM) which extends and includes the concept of the *probe rate model* (PRM) [7] which is employed by existing available bandwidth estimation methods.
- At the best of our knowledge, our PSM-based method is the first attempt to *quickly* estimate the effective UDP throughput within only several hundred milliseconds.
- Also, our method is the first attempt to *simultaneously* estimate both of the available bandwidth and the effective UDP throughput from a single measurement.

II. Effective UDP Throughput

The end-to-end effective UDP throughput is defined as the receiving rate of an UDP flow that can pass through an end-to-end path, with a given sending rate; i.e., successfully received UDP data size divided by a given transmission time. For simplicity, we assume the given sending rate is constant bit-rate (CBR).

Note that the effective UDP throughput has dynamic characteristics because it depends on both of the given sending rate and cross-traffic. Namely, the higher sending rate is given, the higher effective UDP throughput tends to obtain. When a new UDP flow whose sending rate is higher than the available bandwidth is injected, it may take the bandwidth of existing cross-traffic away. If the cross-traffic is also UDP flows, the new and existing UDP flows compete and experience some packet losses each other, due to unresponsive nature of UDP. On the other hand, if the cross-traffic is TCP flows, the new UDP flow overcomes TCP flows, due to responsive and elastic nature of TCP's congestion control mechanisms [8].

Since the effective UDP throughput is UDP-specific, whereas the available bandwidth does not depend on a specific transport protocol, thus these two are fundamentally different metrics [9]. Also, the *achievable UDP throughput* [9] and the effective UDP throughput are not the same; the former is the maximum value of the latter. So, in order to obtain the former, a given sending rate must be very high, at least more than the end-to-end capacity. From the point of view of real-time communication applications, however, the achievable UDP throughput is usually too higher than the maximum video sending rate, e.g., the maximum video sending rate of Skype is about 1Mbps [1] and that of Polycom HDX 9004 (a commercial video conferencing product) is 6Mbps [10]. So, the effective UDP throughput is more preferable for these applications rather than the achievable UDP throughput.

III. Related Work

A. Available Bandwidth Estimation

Much prior work has been done on end-to-end available bandwidth estimation [4]. Representative examples are Pathload [11], pathChirp [12] and Spruce [13]. However, they have a critical restriction in that they require long estimation duration, so they are not suitable for real-time communication. Indeed, it has been reported that the estimation durations of Pathload, pathChirp and Spruce are as much as 7.0 to 22.0 s, 5.5 s and 11.0 s, respectively [14]. Several other methods [15][16][17][5] have been proposed. However, the shortest estimation durations reported for them are 5.6 s [15], 10.0 s [16], 20.0 s [17] and 30.0 s [5], respectively. Our previous method, PathQuick, is an only exception that it achieves quick available bandwidth estimation within only several hundred milliseconds [6].

B. UDP Throughput Estimation

At the best of our knowledge, UDP throughput estimation is scarcely studied. On the other hand, there are several direct UDP throughput measurement (not estimation) methods; Iperf [18] (with –u option) and Netest [9]. Iperf spaces out each probing packet to match a user-specified CBR sending rate, and reports observed effective UDP throughput and packet loss rate after the measurement. However, they also have a critical restriction in that they require a long measurement time. Indeed, the measurement time reported for Iperf is 10.0 s [14]. To make matters worse, Iperf is quite intrusive. One possible way to improve Iperf's problems may be to shorten its measurement time. If its CBR transmission time is shortened ultimately, Iperf transmits a pair of probing packets. However, it is reported that a packet pair method is less accurate than a packet train method in general since its accuracy is highly affected by the packet size of cross-traffic [19], as empirical evaluations have confirmed [14]. The shortest measurement time reported for Netest is 30.0 s [20]. So, they are not suitable for real-time communication.

C. Estimation of Two Metrics

There are several methods that estimate two metrics [21][22][23][24][25][26]. However, all of them estimate the available bandwidth and the capacity of end-to-end path, whereas PathQuick2 estimates the available bandwidth and the effective UDP throughput. Furthermore, [21][22][23][24] first estimates the one metric and then the another metric. Namely, the two estimations are temporally serialized with different probing packets, and thus requiring a longer estimation duration and more probing load than a single metric estimation. In contrast, PathQuick2 estimates the two metrics at the same time with a single packet train.

D. TCP Throughput Estimation and Other Related Method

Various methods to estimate TCP throughput have been proposed such as equation-based methods [27][28], history-based methods [29][30][31][32] and the rest [33]. However, they are not useful for real-time communication applications since most of them use UDP rather than TCP.

RTCP [34] may be complementary to PathQuick2, and they may be used together. However, RTCP can only be used *during* video transmission since a RTCP receiver feedbacks the statistics of a path such as packet losses and delay jitter to a RTCP sender only after the beginning of video transmission. Namely, it cannot be used *just before* video transmission. On the other hand, PathQuick2 can be used not only *during* but also *just before* video transmission. For example, PathQuick2 can be used to determine the initial video sending rate, while RTCP cannot be used.

IV. Proposal of PathQuick2

A. Requirements for Estimation of the Available Bandwidth and the Effective UDP Throughput

Given the problems with conventional methods, we identified two requirements that must be satisfied to enable estimation of the available bandwidth and the effective UDP throughput for real-time communication.

(1) **Short estimation duration:** In spite of the need to estimate the two different metrics, the estimation

duration must be short to ensure the real-time responsiveness.

(2) **Light probing load:** In spite of the need to estimate the two different metrics, the probing load must be minimized to avoid undesirable congestion.

B. Overview of Our Previous Method, PathQuick

Before stepping forward to explain PathQuick2, we describe the overview of our previous quick available bandwidth estimation method, PathQuick [6]. In PathQuick, a sender transmits a UDP packet train to a receiver. The receiver then estimates the available bandwidth and reports the estimated result to the sender. PathQuick is based on the probe rate model (PRM) [7]. PRM is also employed by existing available bandwidth estimation methods such as pathChirp and Pathload [19]. PRM is based on the observation that (a) if the probing rate of a packet train at a sender is less than the available bandwidth, the probing packets will face no queuing delay at routers, so the time interval for each probing packet observed at a receiver will be the same as at the sender. On the other hand, (b) if the probing rate exceeds the available bandwidth, the packets will be queued at some router, increasing the time intervals observed at the receiver. The available bandwidth can be estimated by observing the probing rate at which there is a transition from (a) to (b).

1) Design of Packet Train Structure

We designed the packet train structure of PathQuick with the following features. In order to make the whole transmission duration of a packet train short, the time interval for each packet within the packet train must be short. To this end, we designed the packet train so that each packet is placed at an equal time interval (see Fig. 1-(1)). Also, in order to probe over a wide range of rates with a single packet train, the per-packet probing rate must be changed within the single packet train. To this end, we designed the structure so that each packet size linearly increases from the previous one as the packet sequence proceeds (see Fig. 1-(2)).

Let us consider a packet train consisting of N probing packets. Each packet within the packet train is placed at equal time interval T_{quick} at the sender (Fig. 1-(1)). The whole transmission duration of a packet train (i.e., the packet train length) is

$$T_{train}^{(quick)} = T_{quick} \cdot (N-1) = T_{quick} \cdot N - T_{quick} \, . \qquad (1)$$

Thus, packet train length $T_{train}^{(quick)}$ is a linear function of the number of probing packets N. This $O(N)$ nature enables PathQuick to keep the packet train length short.

The packet size of each probing packet is

$$P_i = P_1 + (i-1) \cdot \Delta P = \Delta P \cdot i + (P_1 - \Delta P), \qquad (2)$$

where $i = 1, 2, \ldots, N$ and the constant value ΔP is the increase amount of the packet size (Fig. 1-(2)). Thus, each packet size P_i is a linear function of i, since P_1 is a constant value.

The per-packet probing rate at the i-th packet – i.e., the momentary probing rate of the packet train – is

$$R_i = \frac{P_i}{T_{quick}} = \frac{\Delta P}{T_{quick}} i + \frac{P_1 - \Delta P}{T_{quick}} \, . \qquad (3)$$

Thus, each per-packet probing rate R_i is also a linear function of i. Therefore, PathQuick can increase the per-packet probing rate within a single packet train, and thereby can probe over a wide range of rates using a single packet train.

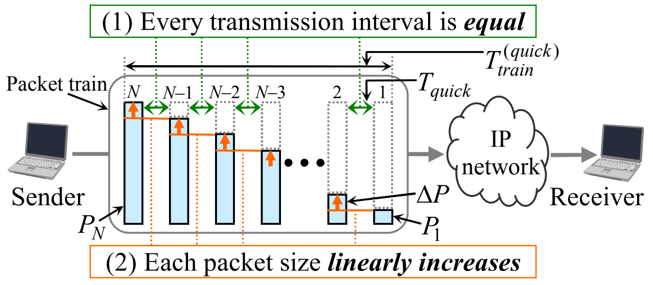

Figure 1. Design of packet train structure.

2) PRM-based Available Bandwidth Estimation

Let us define the time interval between the i-th and $(i-1)$-th packet observed at the receiver as T_i^{rcv}, and the sender transmission time of the i-th packet as t_i^{snd}.

Assuming CBR cross-traffic, the receiver analyzes the observed time intervals based on the PRM principle to estimate the available bandwidth as follows:

$$\begin{array}{ll} \text{(a)}\ T_i^{rcv} = T_{quick}, & \text{if } R_i \le B[t_1^{snd}, t_N^{snd}] \\ \text{(b)}\ T_{i-1}^{rcv} < T_i^{rcv}, & \text{otherwise,} \end{array} \qquad (4)$$

where $B[t_1^{snd}, t_N^{snd}]$ is the actual available bandwidth between times t_1^{snd} and t_N^{snd}.

In PathQuick, a per-packet probing rate $R_k = P_k / T_{quick}$, where the k-th packet is the packet at which the observed time intervals at receiver T_i^{rcv} begin increasing, becomes the estimated available bandwidth. Fig. 2 illustrates the meaning of Eq. (4), $T_2^{rcv} = T_3^{rcv} =, \ldots, = T_{k-2}^{rcv} = T_{k-1}^{rcv} = T_{quick}$ in (a) and $T_{quick} < T_k^{rcv} < T_{k+1}^{rcv} <, \ldots, < T_N^{rcv}$ in (b). That is, the k-th packet is the transition point of PRM, and the per-packet probing rate of the k-th packet becomes the estimated available bandwidth.

The above assumption of CBR cross-traffic, however, does not always hold true in real networks. Intermittent and bursty cross-traffic causes the queuing delay to fluctuate, so the observed time intervals may not increase monotonically. Hence, simply using Eq. (4) would lead to erroneous estimation results. To avoid this, we employ a statistical technique which is described in detail in [6].

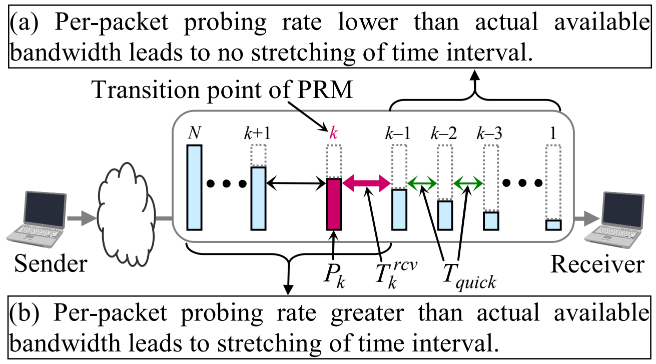
(a) Per-packet probing rate lower than actual available bandwidth leads to no stretching of time interval.
Transition point of PRM
N
k+1
k
k–1
k–2
k–3
1
Sender
Receiver
P_k
T_k^{rcv}
T_{quick}
(b) Per-packet probing rate greater than actual available bandwidth leads to stretching of time interval.

Figure 2. PRM-based available bandwidth estimation.

C. *Proposal of the Probe Slope Model and PathQuick2*

In this paper, we propose a method called PathQuick2 that can quickly and simultaneously estimate both of the available bandwidth and the effective UDP throughput of an end-to-end path by using a single packet train. Namely, PathQuick2 has a "kill two birds with one stone" characteristic. To this end, we develop the *probe slope model* (PSM), which extends and includes the concept of probe rate model (PRM). PathQuick2 is based on PSM and satisfies both of the requirements described in Section IV-A.

In PRM, the temporal change is taken notice to detect the transition point as illustrated in Fig. 2. In contrast, the receiving rate change is taken notice in PSM. Assuming CBR cross-traffic, Fig. 3 illustrates the concept of PSM. Fig. 3 depicts the *slope* of a series of the per-packet receiving rates that is (i) steep at first but (ii) gradually changes to be gentle and (iii) finally comes to be horizontal. PSM is based on the observation that (i) if the per-packet sending rate is less than the available bandwidth, a series of the per-packet receiving rates will increase linearly. Then, (ii) if the per-packet sending rate begins to exceed the available bandwidth, the slope of the per-packet receiving rate series begins to be gentle. After this, the UDP probing packets begins to push the cross-traffic aside. Since the acceleration of pushing aside declines gradually during the competition, the slope continues to be gentle. Finally, (iii) if the per-packet sending rate begins to exceed the effective UDP throughput, the tight link is momentarily saturated, so the slope becomes practically horizontal.

Namely, the first transition point from (i) to (ii) in PSM corresponds to the single transition point of PRM. The per-packet receiving rate at the first transition point in PSM becomes the estimated available bandwidth. The effective UDP throughput can be estimated by observing the receiving rate at the second transition point from (ii) to (iii) in PSM. PathChirp, Pathload, PathQuick and other PRM-based methods focus on only the first transition point in PSM. In contrast, PathQuick2 is the first method which focuses on both of two transition points in PSM.

A PathQuick2 sender is unchanged from a PathQuick sender, whereas a PathQuick2 receiver is modified based on PSM. Consequently, the PathQuick2's packet train structure is the same as the PathQuick's one. In [6], we have validated that PathQuick's estimation duration is sufficiently short and its probing load is light enough for real-time communication. The whole packet train length of PathQuick2 does not suffer any stretch than PathQuick, while PathQuick2 has additional capability that it can estimate the effective UDP throughput from a single packet train. Consequently, PathQuick2 satisfies the first requirement. Furthermore, the total packet size of the single packet train of PathQuick2 does not suffer any increase. Hence, PathQuick2 also satisfies the second requirement.

Our PSM complements and extends the large body of work on network measurement. For example, if we apply the concept of PSM to pathChirp, the modified version of pathChirp will enable us to estimate the two metrics simultaneously from a single packet train.

Note that we confirm no packet loss occurs due to the PathQuick2 measurement unless highly heavy congestion condition, since the excess of available bandwidth is momentarily and the impact is absorbed by router queues.

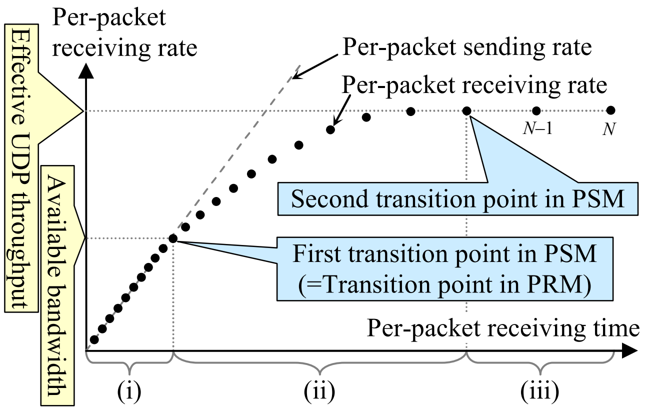
Per-packet receiving rate
Effective UDP throughput
Available bandwidth
Per-packet sending rate
Per-packet receiving rate
N–1
N
Second transition point in PSM
First transition point in PSM (=Transition point in PRM)
Per-packet receiving time
(i)
(ii)
(iii)

Figure 3. Concept of the probe slope model (PSM).

D. *Recursive Effective UDP Throughput Estimation Algorithm*

The assumption of CBR cross-traffic in Section IV-C does not always hold true in real networks due to intermittent and bursty cross-traffic. Fig. 4 depicts the uneven per-packet receiving rate of a typical packet train in real networks. From the point of microscopic view, due to the bursty cross-traffic effect, the slope of the per-packet receiving rate series may never be regarded as horizontal at last, or may be regarded as horizontal at very early stage by mistake.

In order to mitigate the bursty cross-traffic effect, we develop the recursive effective UDP throughput estimation algorithm. In the algorithm, we analyze the slope from the point of macroscopic view. Basic idea is that, we divide a long section of a packet train into two short sections. If the ratio of averaged receiving rate of the long section to that of the latter short section is greater than a predefined value, we continue the divide and compare operations, in a recursive manner, until the ratio becomes lower than the predefined value (see Fig. 4).

More specifically, Fig. 5 shows the pseudo code of the algorithm. Let us define the receiving time of i-th packet as t_i^{rcv}, the total packet size from the first packet to i-th packet as s_i, the packet sequence number of the long section's first packet as $start$ and the packet sequence number of the short section's first packet as mid. The long section length at j-th recursive iteration is $T_j^{long} = t_N^{rcv} - t_{start}^{rcv}$, and The short section length at j-th recursive iteration is $T_j^{short} = t_N^{rcv} - t_{mid}^{rcv}$ where $j = 1,2,\cdots$. The averaged receiving rate of the long section at j-th recursive iteration is $R_j^{long} = (s_N - s_{start}) / T_j^{long}$, and the averaged receiving rate of the short section at j-th recursive iteration is $R_j^{short} = (s_N - s_{mid}) / T_j^{short}$. At each recursive iteration, mid is updated to $mid = \lfloor (start + N + 1)/\alpha \rfloor$ where the division factor α is the ratio of number of packets between the long section and the short one. Also $start$ is updated to $start = mid$. The division factor α controls the number of packets of a short section. For example, if $\alpha = 2$ and $N = 24$, the first long section (covered by the lowest horizontal blue line) has 24 packets, and the first short section (covered by the second lowest horizontal blue line) has 24 / 2 = 12 packets. Also, the values of mid in the first, second and third recursive iteration are $\lfloor (1+24+1)/2 \rfloor = 13$, $\lfloor (13+24+1)/2 \rfloor = 19$ and $\lfloor (19+24+1)/2 \rfloor = 22$, respectively. The recursive algorithm stops if $R_j^{short} < (1+\varepsilon) R_j^{long}$ where ε is the nearness threshold, and the estimated effective UDP throughput becomes $\left(R_j^{short} + R_j^{long}\right)/2$.

We are currently studying other ways to mitigate the bursty cross-traffic effect. One of them is to use the exponentially weighted moving average (EWMA). Another way is to use the Kalman filter.

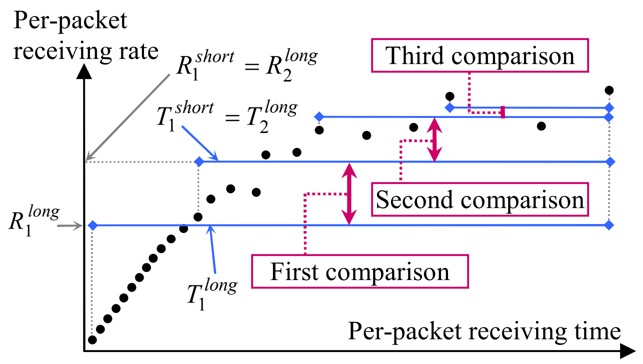
Per-packet receiving rate
$R_1^{short} = R_2^{long}$
Third comparison
$T_1^{short} = T_2^{long}$
Second comparison
R_1^{long}
First comparison
T_1^{long}
Per-packet receiving time

Figure 4. Recursive effective UDP throughput estimation algorithm.

```
function estimate_effective_udp_throughput () {
        $start = 1$
        for ( $j = 1$; $N - start \geq 1$; $j{+}{+}$ ) {
                $mid = \lfloor (start + N + 1)/\alpha \rfloor$
                $T_j^{long} = t_N^{rcv} - t_{start}^{rcv}$
                $T_j^{short} = t_N^{rcv} - t_{mid}^{rcv}$
                $R_j^{long} = (s_N - s_{start}) / T_j^{long}$
                $R_j^{short} = (s_N - s_{mid}) / T_j^{short}$
                if ( $R_j^{short} < (1+\varepsilon) R_j^{long}$ ) {
                        return $\left(R_j^{short} + R_j^{long}\right)/2$
                }
                $start = mid$
        }
}
```

Figure 5. Pseudo code of the recursive estimation algorithm.

E. Implementation

We implemented a prototype PathQuick2 system as ActiveX components that can run on Windows OS (see Fig. 6). Thus, the PathQuick2 system can be easily integrated into real-time communication applications.

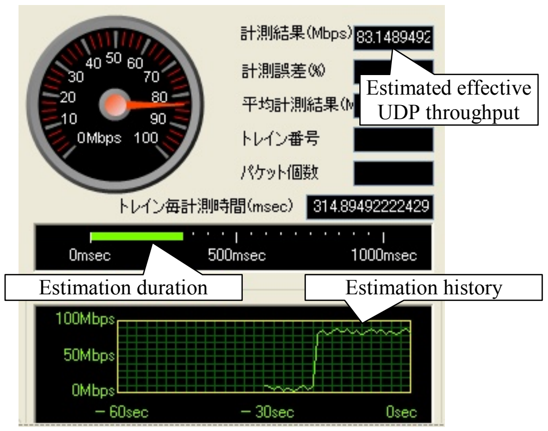
計測結果(Mbps) 83.1489492
計測誤差(%)
平均計測結果(M
トレイン番号
パケット個数
トレイン毎計測時間(msec) 314.89492222429

Figure 6. Snapshot of the prototype PathQuick2 system.

V. EVALUATION

We evaluated PathQuick2 regarding various aspects; estimation duration, probing load and estimation accuracy.

A. Parameter Choice for Quantitative Evaluation

Before going into details of the evaluation, we will explain how we choose the values of several parameters used for the quantitative evaluation.

1) Maximum Probable Bandwidth $B_{\max}$

In this evaluation, we suppose a video conferencing as a real-time communication application since the possible variation range of video sending rate is wider than video chat. A maximum video sending rate of a current commercial product is 6 Mbps [10], and maximum video sending rates of commercial products continues to increase in recent years. With an eye to the future, we doubled the rate, and thus we choose 12 Mbps as the maximum video sending rate. Hence, we choose 12 Mbps as the maximum probable bandwidth of a probing packet train $B_{\max}$.

To realize the 12-Mbps target of maximum probable bandwidth, we set the equal time interval $T_{quick} = 1$ ms, the packet size of first packet $P_1 = 1$ byte and the increase amount of the packet size $\Delta P = 12$ bytes. We set the packet size of last packet $P_N = 1{,}489$ bytes. This means a packet train of PathQuick2 consists of $N = 1 + (1{,}489 - 1)/12 = 125$ packets. So, $B_{\max} = P_N / T_{quick} = 8 \times 1{,}489 / 0.001 = 11{,}912$ kbps. This practically covers the 12-Mbps target.

2) RTT

Since the average one-way delay (OWD) in current Japanese Internet use among 13 major Japanese cities has been reported to be 26 ms in Table 1 of [35], we set the end-to-end round trip time (RTT) to 52 ms in our evaluation.

B. Experimental Setup

We did simulations using an ns-2 network simulator [36] to obtain the evaluation results in a controllable and repeatable manner. Fig. 7 shows the *dumbbell* topology, with a single bottleneck link, of the simulations. Since we intend to explore PathQuick2 under the condition that the 12-Mbps maximum video sending rate is moderately close to the physical capacity of the bottleneck link, e.g., approximately

80%, we choose 16 Mbps as the physical capacity of the bottleneck link. We set the RTT to 52 ms. The physical capacity and OWD of each link are also shown in Fig. 7. We conducted two types of simulation using different cross-traffic over the same topology. The cross-traffic in the first simulation is multiple UDP flows with Poisson packet arrivals, whereas the second one is multiple TCP flows. In order to measure actual effective UDP throughput (i.e., the right answer to the PathQuick2's estimates) during running of the cross-traffic, we emulated Iperf, a direct UDP throughput measurement tool. The sending rate of the emulated Iperf is set to 12 Mbps with CBR in all the simulations since the maximum probable bandwidth of PathQuick2 is 12 Mbps. Since the measurement time reported for Iperf is 10.0 s [14], and Iperf transmits UDP packets with 1,470 bytes, we emulate the behavior accordingly. The timing of the estimation with PathQuick2 and the measurement with Iperf are separated to avoid interference between each other's outputs.

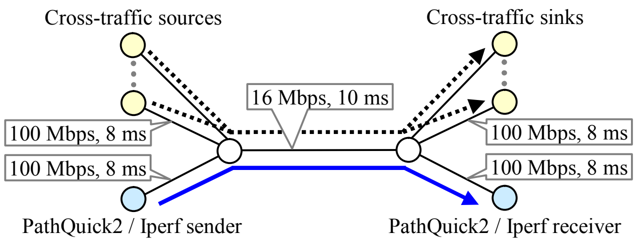

Figure 7. Topology with a single bottleneck link.

C. Estimation Duration

We measured the estimation duration of PathQuick2. It is the sum of the packet train length $T_{train}^{(quick)}$, transmission (or serialization) delay, queuing delay at routers, and RTT. According to Section V-A, $T_{train}^{(quick)}$ is calculated to $T_{train}^{(quick)} = T_{quick} \cdot (N-1) = 1 \cdot (125-1) = 124$ ms and the RTT is 52 ms. We observed the average estimation duration of PathQuick2 over the two types of simulation is 182 ms. This duration is shorter than the 400-ms OWD requirement of two-way video communication recommended by ITU-T [37].

D. Probing Load

We measured the total packet size of a single packet train (i.e., intrusiveness). PathQuick2's total packet size of a single packet train is 1+13+, … , +1,489 = 93.1 kB. In contrast in Iperf, its total packet size of 10.0-s measurement with the 12-Mbps sending rate is as much as 10.0×12 / 8 = 15 MB. Thus, the probing load of PathQuick2 is 15 MB / 93.1 kB = 161.1 times as light as Iperf.

E. Estimation Accuracy with Poisson UDP cross-traffic

We evaluated the estimation accuracy of PathQuick2 with Poisson UDP cross-traffic with a 1,000-byte packet size. Note that the Internet traffic has been shown to be well represented by the Poisson model at small time scales such as less than 1 s [38] or 5 s [39], and the estimation duration of PathQuick2 is less than 1 s as shown in Section V-C. The averaged cross-traffic load aggregated at the entrance of bottleneck link was varied from 4 to 16 Mbps, and thus the available bandwidth was varied from $16-4=12$ Mbps to $16-16=0$ Mbps. We mainly focus on the estimation accuracy of the effective UDP throughput rather than the available bandwidth, since we have already validated the latter in [6]. We use simulations to better understand the role of the several PathQuick2 parameters. We varied the division factor α and the nearness threshold ε in the recursive effective UDP throughput estimation algorithm described in Section IV-D, while keeping other parameters constant as described in Section V-A.

First, we assess the impact of the division factor α. A larger α leads to the less number of packets of short sections. Fig. 8 demonstrates the effect of the division factor α. Actual effective UDP throughput measured by the emulated Iperf is plotted with black circles. We observed that $\alpha = 2.0$ leads to somewhat wide range estimation error, and $\alpha = 2.4$ leads somewhat underestimate in particular when there is enough available bandwidth (the right end in Fig. 8). So, we choose $\alpha = 2.2$.

Next, we assess the impact of the nearness threshold ε. This parameter affects the convergence condition of the recursive algorithm. Fig. 9 demonstrates the effect of the nearness threshold ε. A smaller ε leads to tighter convergence condition, and we observed that increasing the estimation accuracy. Based on the results, we choose $\varepsilon = 0.05$.

We extract the estimation results with only the case of $\alpha = 2.2$ and $\varepsilon = 0.05$ in Fig. 10. We observed PathQuick2's estimates are conservative, i.e., on the safe side from the point of view of real-time communication. Its maximum estimation error is approximately 2 Mbps. Considering that its estimation duration is sufficiently short and also its probing load is 161.1 times as light as Iperf, we believe that the estimation accuracy of PathQuick2 is reasonable in practical use for real-time communication.

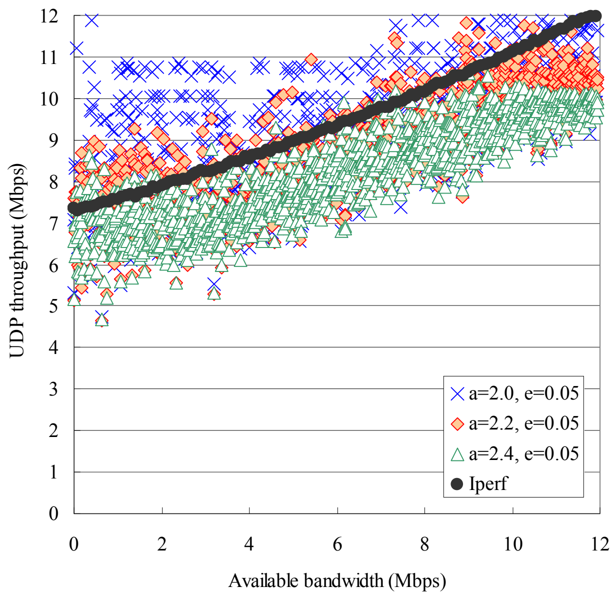

Figure 8. Estimation accuracy and the variation of division factor α.

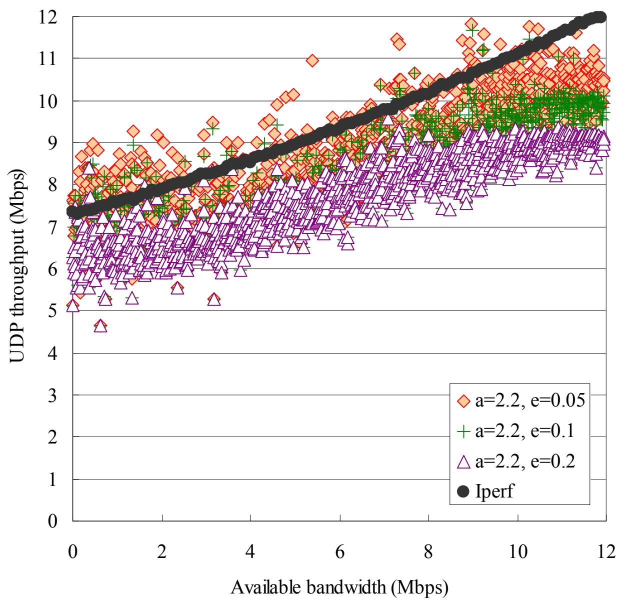

Figure 9. Estimation accuracy and the variation of nearness threshold ε .

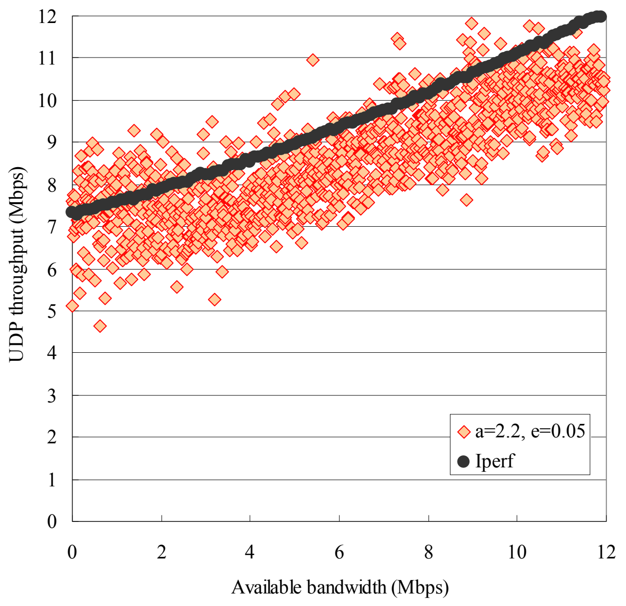

Figure 10. Estimation accuracy where $\alpha = 2.2$ and $\varepsilon = 0.05$.

F. *Estimation Accuracy with TCP cross-traffic*

We evaluated the estimation accuracy of PathQuick2 with multiple TCP cross-traffic. Akamai has reported [40] that the global average TCP throughput is approximately 2 Mbps over the current Internet. Since the bottleneck capacity is 16 Mbps, we set the maximum number of TCP flows to 16 / 2 = 8 flows. In this simulation, we varied the number of TCP flows from 4 to 8 flows. We use $\alpha = 2.2$ and $\varepsilon = 0.05$.

Fig. 11 shows the estimation results. The 10.0-s measurement results with the emulated Iperf are quite nearly 12 Mbps, consistently. As described in Section II, this is due to the responsive and elastic nature of TCP and the unresponsive nature of UDP. During the 10.0-s measurement, the emulated Iperf rapidly take the TCP's bandwidth away up to 12 Mbps. We observed PathQuick2's estimates are also consistent; its average estimation errors are approximately from 2 to 3 Mbps underestimate. Although the maximum estimation error, approximately 5 Mbps, is larger than the case of Poisson UDP cross-traffic, PathQuick2's estimates are also still on the safe side.

Note that TCP flows aggressively consume the available bandwidth in general. Indeed, we confirmed that the available bandwidth is always almost zero throughout the simulations, even when there is neither PathQuick2 nor emulated Iperf traffic. In terms of the absence of available bandwidth, the conditions may be similar to the left end of Fig. 10. Although PathQuick2's estimates range from 5 to 9 Mbps in the left end of Fig. 10, the ones range from 7 to 12 Mbps in Fig. 11, i.e., higher range. This implies PathQuick2 can chase the effective UDP throughput without dependent on the transport protocol of cross-traffic.

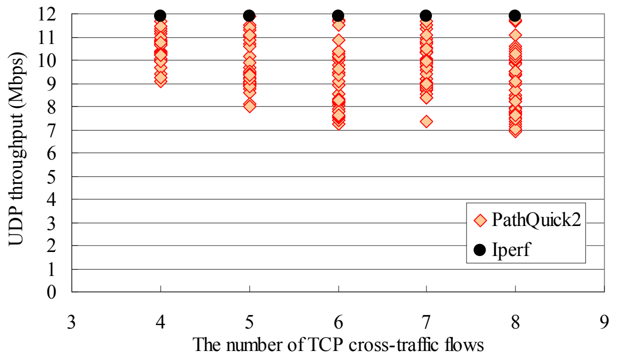

Figure 11. Estimation accuracy vs. multiple TCP cross-traffic flows.

VI. Discussion

The greedy behavior of real-time communication applications described in Section I leads to unfairness between UDP and TCP, and may cause the congestion collapse from undelivered packets [8]. To avoid this, TCP-friendly rate control (TFRC) [41] is proposed for real-time communication. A TFRC-based real-time communication application can control the video sending rate and maintain the inter-protocol's fairness. Since the TFRC only uses the information of packet size, RTT and packet loss rate for its rate control, the knowledge of the available bandwidth and the effective UDP throughput potentially improve further its rate control mechanism. We are currently developing an adaptive TFRC-based rate control method for real-time communication which combines PathQuick2 and the adaptive rate control method [42] that is developed in our research group, and preliminary results are encouraging.

VII. Conclusion and Future Work

PathQuick2 is a method that can quickly and simultaneously estimate both of the available bandwidth and the effective UDP throughput from a single packet train. Through an evaluation, we confirmed that PathQuick2 completes the estimation in 182 ms. We also confirmed that its probing load is about 90 kB which is more than 160 times as light as a direct measurement of effective UDP throughput.

In future work, we will exhaustively analyze the relationship among the estimation accuracy, the division factor α and the nearness threshold ε to further improve the estimation accuracy. We also plan to conduct large-scale experiments over the Internet with the prototype system to validate PathQuick2 in real networks.

REFERENCES

[1] L. D. Cicco, S. Mascolo and V. Palmisano, "Skype video responsiveness to bandwidth variations," *ACM NOSSDAV*, pp. 81–86, 2008.

[2] L. Gharai, C. Perkins and A. Mankin, "Scaling video conferencing through spatial tiling," *ACM NOSSDAV*, pp. 137–143, 2001.

[3] O. Boyaci, A. G. Forte and H. Schulzrinne, "Performance of video-chat applications under congestion," *IEEE ISM*, pp. 213–218, 2009.

[4] R. Prasad, C. Dovrolis, M. Murray and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, Vol. 17, Issue 6, pp. 27–35, 2003.

[5] A. Akella, S. Seshan and A. Shaikh, "An empirical evaluation of wide area Internet bottlenecks," *ACM IMC*, pp. 101–114, 2003.

[6] T. Oshiba and K. Nakajima, "Quick end-to-end available bandwidth estimation for QoS of real-time multimedia communication," *IEEE ISCC*, pp. 162–167, 2010.

[7] L. Lao, C. Dovrolis and M. Y. Sanadidi, "The probe gap model can underestimate the available bandwidth of multihop paths," *ACM SIGCOMM CCR*, Vol. 36, Issue 5, pp. 29–34, 2006.

[8] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking (TON)*, Vol. 7, No. 4, pp. 458–472, 1999.

[9] G. Jin and B. Tierney, "Netest: a tool to measure the maximum burst size, available bandwidth and achievable throughput," *IEEE ITRE*, pp. 578–582, 2003.

[10] Polycom, Inc., "Polycom HDX 9000 series FAQ," [online] http://www.ivci.com/pdf/polycom-hdx-series-faq.pdf

[11] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *ACM SIGCOMM*, pp. 295–308, 2002.

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

[13] J. Strauss, D. Katabi and F. Kaashoek, "A measurement study of available bandwidth estimation tools," *ACM IMC*, pp. 39–44, 2003.

[14] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov and K. Claffy, "Comparison of public end-to-end bandwidth estimation tools on high-speed links," *PAM Workshop*, pp. 306–320, 2005.

[15] D. Croce, T. En-Najjary, G. Urvoy-Keller and E. W. Biersack, "Fast available bandwidth sampling for ADSL links: rethinking the estimation for larger-scale measurements," *PAM Conference*, pp. 67–76, 2009.

[16] Q. Liu and J. Hwang, "End-to-end available bandwidth estimation and time measurement adjustment for multimedia QoS," *IEEE ICME*, Vol. 3, pp. 373–376, 2003.

[17] S. S. Wang and H. F. Hsiao, "Fast end-to-end available bandwidth estimation for real-time multimedia networking," *IEEE MMSP*, pp. 415–418, 2006.

[18] Iperf, [online] http://iperf.sourceforge.net/

[19] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," *ACM IMC*, pp. 272–277, 2004.

[20] G. Jin, "Algorithms and requirements for measuring network bandwidth," *LBNL Report*, LBNL-48330, 2002.

[21] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE JSAC*, Vol. 21, No. 6, pp. 879–894, 2003.

[22] S. Sargento and R. Valadas, "Accurate estimation of capacities and cross-traffic of all links in a path using ICMP timestamps," *Telecommunications Systems*, Vol. 33, No. 1, pp. 89–115, 2006.

[23] C. L. T. Man, G. Hasegawa and M. Murata, "A merged inline measurement method for capacity and available bandwidth," *PAM Workshop*, pp. 341–344, 2005.

[24] K. M. Salehin and R. Rojas-Cessa, "Combined methodology for measurement of available bandwidth and link capacity in wired packet networks," *IET Communications*, Vol. 4, Issue 2, pp. 240–252, 2010.

[25] B. Melander, M. Bjorkman and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," *IEEE GLOBECOM*, pp. 415–420, 2000.

[26] L. Cong, G. Lu, Y. Chen, B. Deng and X. Li, "pathWave: combined estimation of network link capacity and available bandwidth using statistical signal processing," *IEEE ICON*, pp. 1–6, 2008.

[27] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," *ACM SIGCOMM*, pp. 303–314, 1998.

[28] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM SIGCOMM CCR*, Vol. 27, Issue 3, pp. 67–82, 1997.

[29] Q. He, C. Dovrolis and M. Ammar, "On the predictability of large transfer TCP throughput," *ACM SIGCOMM*, pp. 145–156, 2005.

[30] M. Swany and R. Wolski, "Multivariate resource performance forecasting in the network weather service," *ACM/IEEE Supercomputing*, pp. 1–10, 2002.

[31] M. Mirza, J. Sommers, P. Barford and X. Zhu, "A machine learning approach to TCP throughput prediction," *ACM SIGMETRICS*, Vol. 35, Issue 1, pp. 97–108, 2007.

[32] T. Goto, A. Tagami, T. Hasegawa and S. Ano, "TCP throughput estimation by lightweight variable packet size probing in CDMA2000 1x EV-DO network," *IEEE SAINT*, pp. 1–8, 2009.

[33] A. Tirumala, L. Cottrell and T. Dunigan, "Measuring end-to-end bandwidth with Iperf using Web100," *PAM Workshop*, 2003.

[34] H. Schulzrinne et al., "RTP: a transport protocol for real-time applications," IETF RFC 3550, 2003.

[35] K. Yoshida et al., "Inferring POP-level ISP topology through end-to-end delay measurement," *PAM Conference*, pp. 35–44, 2009.

[36] Network simulator ns-2, [online] http://www.isi.edu/nsnam/ns/

[37] ITU-T recommendation G.1010, "End-user multimedia QoS categories," 2001.

[38] T. Karagiannis, M. Molle, M. Faloutsos and A. Broido, "A nonstationary Poisson view of Internet traffic," *IEEE INFOCOM*, Vol. 3, pp. 1558–1569, 2004.

[39] H. Gupta, A. Mahanti and V. J. Ribeiro, "Revisiting coexistence of Poissonity and self-similarity in Internet traffic," *IEEE MASCOTS*, pp. 1–10, 2009.

[40] Akamai Technologies, Inc., "The state of the Internet, 2nd quarter, 2009," Vol. 2, No. 2, 2009, [online] http://www.akamai.com/html/about/press/releases/2009/press_100109.html

[41] S. Floyd, M. Handley, J. Pahdye and J. Widmer, "TCP friendly rate control (TFRC): protocol specification," IETF RFC 5348, 2008.

[42] H. Yoshida, K. Nogami and K. Satoda, "Proposal and evaluation of joint rate control for stored video streaming," *IEEE CQR*, pp. 1–6, 2010.