Vol. 42 No. 12 IPSJ Journal Dec. 2001

Recommended Paper

Detection of Congestion Signals from Relative One-Way Delay

Yoshito Tobe,† Hiroto Aida,† Yosuke Tamura†
and Hideyuki Tokuda††

In this paper, we describe an experimental method of measuring per-packet *Relative One-way Trip Time* (ROTT) of a UDP stream at a receiver on the Internet as well as its experimetal results. The method uses Incremental Skew Calculation Method (ISCM) to compensate the skew between clocks of a sender and a receiver. It is found that spikes with successive plots, which we call *spike-trains*, often appear on a time-ROTT graph. Also, congestion-related losses are found to be strongly correlated only to the spike-trains and the path status is effectively identified by such spike-trains. From several experiments, it is indicated that congestion signals can be obtained from ROTT.

1. Introduction

Recent advances in computer technology, digital signal processing applications, and packet network capacity have accelerated the proliferation of real-time, packet-based continuous media (CM) communications. Furthermore, the growth of the Internet has fueled demand for these technologies and highlighted the difficulties of handling CM in a wide area network. Once a CM flow is injected into a best-effort Internet, a rate control that avoids congestion is essential. UDP flows that carry CM do not automatically control their rates, which may make both the network and themselves inefficient. To overcome this problem, TCP-friendly rate control algorithms [8)~10),12)] have been introduced.

A problem in the rate control is the type of information whereby congestion is detected and determined. With the exception of some proposals [1),3)], TCP relies on a loss of acknowledgments (ACKs) indicating a packet loss to detect congestion. Unlike a TCP flow, a sender of a CM UDP flow does not usually receive per-packet ACKs. Instead, it is recommended that the CM UDP flow uses Real-time Transport Protocol (RTP) and its control protocol, RTCP, to control a rate based on information about losses in a certain period. However, reliance on information about losses alone may lead to an erroneous control. First, when the UDP flow competes with a TCP flow, the TCP flow reduces its rate more quickly, resulting in a few or even no packet losses in the UDP flow. In other words, reliance on the information about loss only works when a UDP receiver notifies the corresponding sender of the loss immediately and the sender reacts to the notification quickly. Second, as is well known, packets may be lost in the absence of congestion over a lossy wireless link [4)].

To explore an alternative way to detecting congestion, we investigate a way to measure one-way delay from the sender to the receiver together with losses. Since an absolute value of delay is difficult to obtain due to a skew between the clocks at the sender and the receiver, we observe variation in the one-way delay. We call such a variation *Relative One-way Trip Time* (ROTT). Extensive experimental results show that a value of ROTT often increases with a spike accompanied by successive plots, which we call *spike-trains*, on a time-ROTT graph.

We then investigate the correlation between the ROTT and losses, and find that spike-trains are only related to congestion-related losses.

The remainder of this paper is organized as follows. In Section 2, some measured results of ROTT over the Internet are shown. In Section 3, the dependence between ROTT and packet losses is analyzed. In Section 4, related works are described.

2. Path Status

2.1 Identification of the Variation in One-Way Delay

Prior to discussing how to determine the path

† Graduate School of Media and Governance, Keio University
†† Faculty of Environmental Information, Keio University

The initial version of this paper was presented at the DPS Workshop held on Dec. 2000, which was sponsored by SIGDPS. This paper was recommended to be submitted to the Journal of IPSJ by the chairman of SIGDPS.

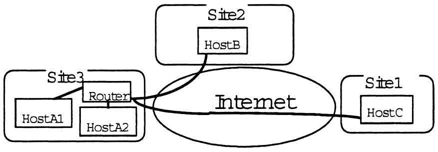

Fig. 1 Topology over the Internet.

status, we define how to deal with delay. Since absolute values of one-way delay are difficult to measure, we calculate the variation in delay, instead. First, we assume that a sender is transmitting rate-controlled data at a certain regular interval with a header including a timestamp and a sequence number. Let $t_s(i)(i = 1, 2, 3, \ldots)$ and $t_r(i)$ denote the time that i-th packet of the CM flow is sent and received, respectively. Then we define a *relative delay* $\Delta t_{rs}(i)$ as follows:

$$\Delta t_{rs}(i) \equiv t_r(i) - t_s(i) - (t_r(1) - t_s(1)).$$

Two values, $t_r(1)$ and $t_r(1)$, are used as initial references for calculation. Note that $t_s(i)$ and $t_r(i)$ are measured with the clock of the sender and the receiver, respectively. Hence $\Delta t_{rs}(i)$ monotonically increases or decreases depending on the difference between the precise tick of these two clocks. Therefore, $\Delta t_{rs}(i)$ is expressed as follows:

$$\Delta t_{rs}(i) = D_{min} + \Delta T_{rs}(i) + \alpha t(i) + \beta,$$

where $t(i)$, α, and β are the receiving time of i-th packet, the skew and the offset caused by the difference between the clocks. Here, D_{min} which is referred to as the base delay is the minimum value of true one-way trip time. However we cannot obtain the value of D_{min} unless the clocks of the sender and the receiver are completely synchronized. Instead, $\Delta T_{rs}(i)$ which is referred to as *Relative One-way Trip Time* (ROTT) can be obtained if α and β are estimated. We use Incremental Skew Calculation Method (ISCM) to calculate α and β. The details of ISCM are described in Appendix.

2.2 Basic Observation of Delay and Loss

We examine the relationship between the loss and ROTT through experiments over the Internet (**Fig. 1**). Communication paths between the following sites were used: **Site1: CESAT in Spain**, **Site2: Carnegie Mellon University (CMU) in U.S.A.**, and **Site3: Keio University in Japan**. Typical RTTs on the Site3–Site2 and Site1–Site3 paths are 140–300 ms and 700–1,500 ms. All hosts are installed to FreeBSD2.2.8R and equipped with a Pentium counter for measurement. The timestamp contained in the header uses a value converted from the Pentium counter. As a result, it does not mean the absolute time.

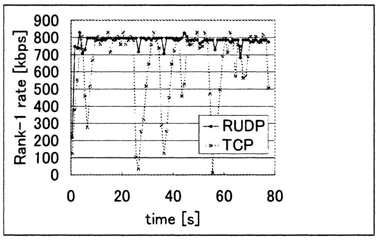

Fig. 2 1-s rate (Site3–Site2).

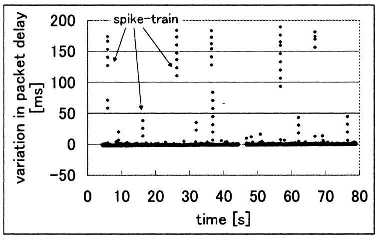

Fig. 3 Spike-trains appears in ROTT (Site3–Site2).

First we used the path between Site2 and Site3. We created one flow of TCP from Host A1 to B and another of Rate-probing UDP (RUDP) from Host A2 to B simultaneously. In RUDP, rate probing using TCP is intermittently inserted [11]. Both RUDP and TCP flows transmit 1,440-byte packets ☆. To discuss rates of flows, we use the amount of received or transmitted bytes in 1 second, which is referred to as 1-s rate or Rank-1 rate.

Figure 2 shows 1-s rates of RUDP and TCP flows measured at Host B. We will examine the cause of frequent dropping in 1-s rate of TCP by investigating ROTT. We then examine ROTT of the RUDP flow. In **Fig. 3**, we set the point at which the first rate probing finishes to the beginning of calculating ROTT. As seen in the figure, several jumped sequences of plots appear in the delay. We call such a sequence a "spike-train." This is the same phenomenon as probe compression in [2], but it is more clearly seen in a time-ROTT graph. Comparing the trend of the delay with Fig. 2, a dropping in the TCP 1-s rate occurs when a spike-train appears. This suggests that packets are likely to be dropped when a spike-train appears.

Let us further see where packet losses occur.

☆ The size of TCP or UDP payload was set to 1,440 bytes, which is a default TCP maximum segment size over Ethernet in FreeBSD 2.2.8R.

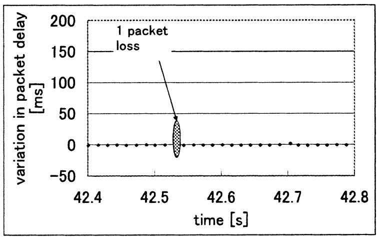

Fig. 4 One packet loss without variation (Site3–Site2).

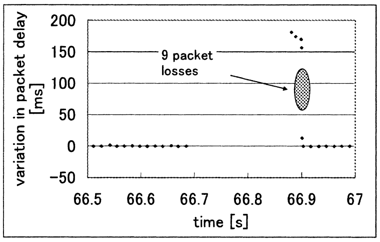

Fig. 5 Packet losses in a spike-train (Site3–Site2).

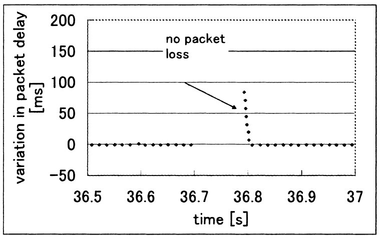

Fig. 6 No packet loss in a spike-train (Site3–Site2).

Figures 4, **5**, **6** show enlarged parts of Fig. 3. As can be seen in these figures, the delay does not always increase when a packet is lost. At the same time, a packet is not always lost when a spike-train appears. However, most packets are lost in spike-trains. We will examine the relationship further later.

We also conducted an experiment over the path between Site1 and Site3. One 1,440-byte UDP data was transmitted at a regular interval. As seen in **Figs.7** and **8**, spike-trains are also observed in the Site1–Site3 path.

3. Dependence between ROTT and Loss

This section provides experimental results of investigating the relationship between ROTT and loss.

3.1 Dependence on Transmission Rate

Based on the preliminary observations above, we further investigate the dependence between ROTT and loss, and also examine the effect of reducing a rate.

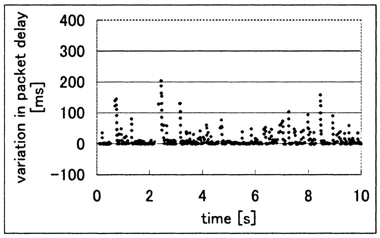

Fig. 7 Spike-trains (Site1–Site3).

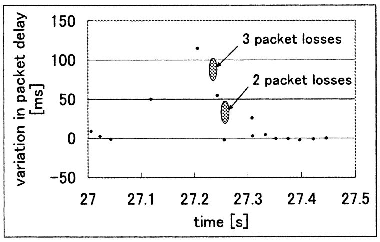

Fig. 8 Packet losses in a spike-train (Site1–Site3).

Table 1 Variables for analysis.

N_p	number of packets expected to be received
N_l	number of lost packets
N_s	number of packets in spike-train periods
N_{sl}	number of lost packets in spike-train periods
N_{bl}	number of lost packets in blackouts
ρ_{pl}	N_l/N_p
ρ_s	N_s/N_p
ρ_{sl}	$N_{sl}/(N_l - N_{bl})$
ρ_{ss}	N_{sl}/N_s

Let us define the variables for analyzing the relationship among ROTT, loss, and rate as listed in **Table 1**. We created Flows 1 and 2 transmitting 1,440-byte packets at a regular interval with different rates simultaneously on the path between Site2 and Site3. The transmission rate of Flow 2 is varied to six cases, 200 kbps, 400 kbps, 900 kbps, 1,200 kbps, 1,800 kbps, and 3,000 kbps, whereas the transmission rate of Flow 1 is fixed at 600 kbps. Ten trials with 80-s duration were conducted for six cases and the variables in Table 1 were calculated. N_p is identified by the difference between the start and end of sequence numbers. To calculate N_s, an algorithm for detecting a spike-train shown in **Fig. 9** was used. The threshold values $B_{spikestart}$ and $B_{spikeend}$ were set to 20 ms and 5 ms, respectively.

Table 2 shows the results obtained. Since an actual transmitted rate is calculated from N_p, the ratio of rates for the two flows γ is calculated by N_p(Flow 1) / N_p(Flow 2). Each case was conducted in a different time. However, a common characteristics can be identified from

==

On receipt of a packet with sequence number i:

if ((state == not-in-spike-train) AND (ROTT(i) $\geq B_{spikestart}$)) {
  state = in-spike-train;
  spike-first-sequence-number = i;
}
if ((state == in-spike-train) AND (ROTT(i) $\leq B_{spikeend}$)) {
  state = not-in-spike-train;
  number-of-spike-packets = i - spike-first-sequence-number + 1;
}

==

Fig. 9 Calculating the number of packets in a spike-train period.

Table 2 Measured statistics (Site3–Site2).

Set		N_p	N_l	N_s	N_{sl}	N_{bl}	ρ_{pl}	ρ_s	ρ_{sl}	ρ_{ss}	γ
1	Flow 1 (600 kbps)	39137	22	267	20	0	5.6×10^{-4}	6.8×10^{-3}	0.91	0.075	0.36
	Flow 2 (200 kbps)	14272	1	73	0	0	7.0×10^{-5}	5.1×10^{-3}	-	-	
2	Flow 1 (600 kbps)	39167	19	209	17	0	4.9×10^{-4}	5.3×10^{-3}	0.89	0.081	0.72
	Flow 2 (400 kbps)	28021	20	169	17	0	7.1×10^{-4}	6.0×10^{-3}	0.85	0.10	
3	Flow 1 (600 kbps)	39165	51	239	49	0	1.3×10^{-3}	6.1×10^{-3}	0.96	0.21	1.6
	Flow 2 (900 kbps)	62425	107	317	106	0	1.7×10^{-3}	5.1×10^{-3}	0.99	0.33	
4	Flow 1 (600 kbps)	39061	56	348	53	0	1.4×10^{-3}	8.9×10^{-3}	0.95	0.15	2.1
	Flow 2 (1,200 kbps)	83817	218	761	216	0	2.6×10^{-3}	9.0×10^{-3}	0.99	0.28	
5	Flow 1 (600 kbps)	39184	4	194	3	0	1.0×10^{-4}	5.0×10^{-3}	0.75	0.015	3.2
	Flow 2 (1,800 kbps)	125012	10	512	8	0	8.0×10^{-6}	4.1×10^{-3}	0.80	0.016	
6	Flow 1 (600 kbps)	39235	25	245	25	0	6.4×10^{-4}	6.2×10^{-3}	1.0	0.10	4.7
	Flow 2 (3,000 kbps)	185750	110	910	98	0	5.9×10^{-4}	4.9×10^{-3}	0.80	0.10	

Table 3 Measured results (Site1–Site3).

Trial	N_p	N_l	N_s	N_{sl}	N_{bl}	ρ_{pl}	ρ_s	ρ_{sl}	ρ_{ss}
1	1706	409	562	53	313	0.24	0.33	0.55	0.094
2	1693	143	1258	111	0	0.084	0.74	0.78	0.088
3	1699	472	741	75	243	0.28	0.44	0.80	0.101
4	1698	155	1611	149	0	0.091	0.95	0.96	0.092

the table. First, ρ_s is almost the same in the two flows. This suggests that a flow suffers from spike-trains in the identical ratio, irrespective of its transmission rate. Second, ρ_{sl} is always near 1.0; most packet losses occur in the spike-train periods. There are some exceptions, but the ratio of them is negligibly small. Finally, ρ_{ss} is approximately identical for the two flows. A possible explanation is that a flow with a larger rate may suffer from a larger number of losses but it can inject more packets in a spike-train period.

It was found that the maximum TCP-equivalent rate was always less than 800 kbps. Nevertheless, flows with a higher rate than 800 kbps, even with 3,000 kbps, do not encounter packet losses at a higher rate. This indicates that a rate reduction scheme for a RTP/UDP cannot rely on information about the loss ratio alone.

Similarly, we measured values of the variables in Table 1 for one UDP flow of 64-byte packet on the Site1–Site3 path. The transmission rate was set to 10 kbps because the maximum TCP rate was measured to be less than 30 kbps. Results of four trials with 80-s duration are shown in **Table 3**. In addition, snapshots of ROTT for trials 1 and 4 are shown in **Figs.10** and **11**, respectively.

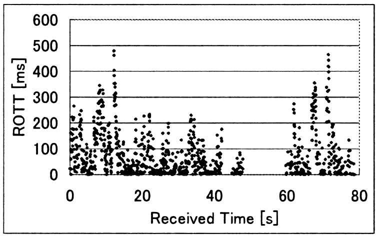

Fig. 10 ROTT (Site1–Site3, trial 1).

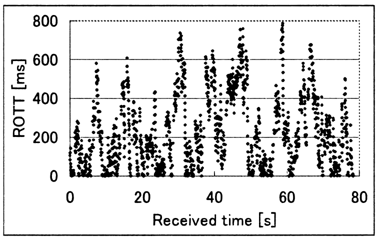

Fig. 11 ROTT (Site1–Site3, trial 4).

In trials 1 and 3, blackout losses appear. In contrast, in trials 2 and 4, there is no such blackout but most packets are in spike-train pe-

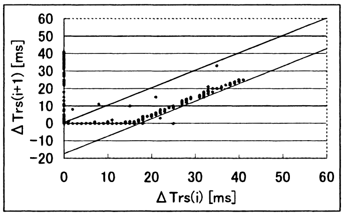

Fig. 12 Phase plot of ROTT for case 5 (Site3–Site2: Flow 1).

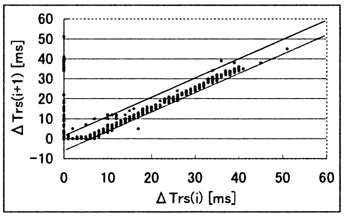

Fig. 13 Phase plot of ROTT for case 5 (Site3–Site2: Flow 2).

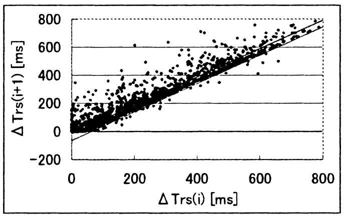

Fig. 14 Phase plot of ROTT for trial 4 (Site1–Site3).

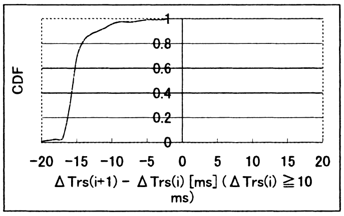

Fig. 15 CDF of $(\Delta T_{rs}(i+1) - \Delta T_{rs}(i))$ for case 5 (Site3–Site2: Flow 1).

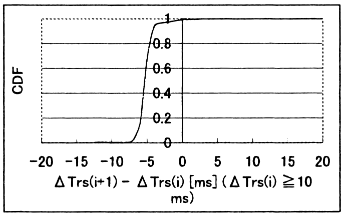

Fig. 16 CDF of $(\Delta T_{rs}(i+1) - \Delta T_{rs}(i))$ for case 5 (Site3–Site2: Flow 2).

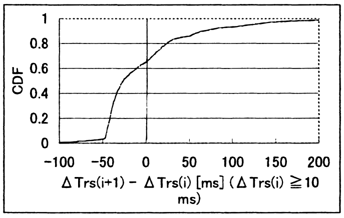

Fig. 17 CDF of $(\Delta T_{rs}(i+1) - \Delta T_{rs}(i))$ for trial 4 (Site1–Site3).

riods. Hence it may be inappropriate to transmit a CM UDP data on such an ill-conditioned path. However, it is interesting to find that ρ_{ss} is almost the same in all cases in this experiments on the path.

The differences between the Site3–Site2 and Site1–Site3 paths are clearly observed in phase plots [2]. When a path is stable with a few spike-trains, sequences of plots both beside the vertical line and along the line

$$\Delta T_{rs}(i+1) = \Delta T_{rs}(i) - \Delta H(\text{constant}) \quad \text{(a)}$$

are evident as shown in **Figs.12** and **13**. In contrast, when a path is unstable with many overlapped spike-trains, there are no such evident plot (**Fig. 14**).

According to [2], ΔH in Eq.(a) is related to an interval of transmission δ; $\Delta H = \delta - P/\mu$, where P and μ are the length of packets and the service rate at the bottleneck, respectively. Let P_p ☆ and u denote the size of packet payloads and the transmission rate, respectively. Since $\delta = P_p/u$, $\mu = P/(P_p/u - \Delta H)$. This is converted to payload-level bottleneck bandwidth μ'; $\mu' = P_p/P \times \mu = P_p/(P_p/u - \Delta H)$. Thus, a spike-train contains information about the bottleneck bandwidth. For instance, in a trial in Set-6 Site3–Site2 experiment, ΔH of Flow 2 was measured to be approximately 3.0 ms. Since $P_p = 1,440$ byte, and $u = 2,674$ kbps, μ is approximately 8.8 Mbps, which is much larger than the rate of Flow 2, 2,675 kbps, and a TCP-equivalent rate, 800 kbps. This suggests that a rate control based only on the bottleneck bandwidth is difficult when the path is relatively stable.

Rather than an absolute value of ΔH, the sign of ΔH provides more useful information. It tells whether the transmission rate exceeds the bottleneck bandwidth. When $u > P_p/(P/\mu + \Delta H)$, $\Delta H > 0$. In Figs.11 and 14, there are many plots where $\Delta H > 0$; the transmission rate should have been set to be below 10 kbps. To observe the distribution of $(\Delta T_{rs}(i+1) - \Delta T_{rs}(i))$, its cumulative distribution functions (CDFs) are shown in **Figs.15**, **16**, **17**. We focus on the area where ROTT $\geq$ 10 ms because the behaviors only when ROTT increases are con-

☆ $P = P_p$ + size of UDP and IP headers.

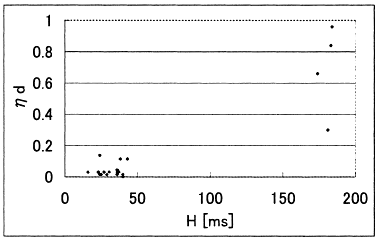

Fig. 18 H vs. η_d.

cerned with congestion. As can be seen in the graphs, most values of $(\Delta T_{rs}(i+1) - \Delta T_{rs}(i))$ are distributed in the negative area for the Site3–Site2 path, and thus the CDF provides indication of stability of the transmission rate.

3.2 TCP Rate vs. ROTT Change

We also examine how the existence of spike-trains affect a TCP 1-s rate. Again, Flows 1 and 2 are simultaneously run on the Site3–Site2 path except that Flow 2 is set to TCP. Let H and η_d denote the largest height of spike-trains in Flow 1 and the ratio of dropping in a 1-s rate for Flow 2 in one-second period, respectively. The obtained results are shown in **Fig. 18**. Interestingly, plots are grouped into two, which suggests that there are two major bottlenecks in the Site3–Site2 path. One group is assumed to correspond to repetition of a smaller number of queued packets at some router. Spike-trains in this group do not affect the TCP rate significantly although the TCP 1-s rate is degrated by up to 20%. The other group is assumed to correspond to larger queue temporarily formed at another router. This type of spike-trains affects the TCP rate by reducing it by 20–100%. It is found that the TCP 1-s rate is not substantially degraded when H is less than 50 ms.

3.3 Summary of Experiments and Discussion

Although the conducted experiments do not cover the entire behavior of the Internet, the following results are found.

- ROTT can be obtained with ISCM.
- A spike-train is strongly correlated to congestion which may reduce the throughput of TCP.
- Reducing the transmission rate does not lead to reduction of packet loss ratio. As shown in Table 1, ρ_{ss} does not always increase when a rate increases. For instance, when 600 kbps- and 3,000 kbps-UDP flows are transmitted simultaneously, ρ_{ss} for both flows is 0.1. This is because the number of sent packets in a spike-train period also increases while the number of lost packets in the same period increases.

The objective of the experiments is to provide a continuous UDP flow with informtion about congestion and a hint of reducing or raising its transmission rate. From the results of Section 3.1, it is suggested that reducing a transmission rate depending on the packet loss ratio is not a good principle as long as most points in the phase plot appear above line (a). The transmssion rate of the UDP flow is not considered to excess an available bandwidth. Instead, the UDP flow is considered to encounter temporary congestion at routers. From the result of Section 3.2, it is also suggested that short spike-train periods can be neglected for controlling a transmission rate of a UDP flow since they do not degrade TCP rate significantly. To design a congestion control algorithm for a continuous UDP flow, decision about how the temorary congestion is dealt with is important.

Although the cause of spike-trains has not been identified yet, there can be some possibilities. One is a sudden increase of other traffic at a queue of a router along the observed path. Such traffic is input into the queue and will delay packets of the observed UDP flow. The delay of the first packet that has been waited in the queue becomes large. Let the first packet be the i-th packet. Then we can say that an inter-arrival time between the $(i-1)$-th and the i-th packets becomes larger than one between the $(i-2)$-th and the $(i-1)$-th packets. We assumed that a large volume of traffic is inserted between the $(i-1)$-th and the i-th packets at the queue. Let us assume that N packets are inserted between the $(i-1)$-th and the i-th packets. In addition, let λ[packet/s], μ[packet/s], and d_i[s] $(\lambda < \mu)$ represent the constant transmission rate of the UDP flow, the evacuation rate of the queue, and the latency of the i-th packet at the queue, respectively. In a steady state with no other large traffic, the delay of a packet at the queue is $1/\mu$. Therefore $d_{i-1} = 1/\mu$. Since N packets are inserted, $d_i = 1/\mu + N/\mu$. Similarly, $d_{i+1} = 1/\mu + (N - \mu/\lambda)/\mu$, $d_{i+2} = 1/\mu + (N - 2 \times \mu/\lambda)/\mu$, ... Thus d_i becomes the largest and the delayed packets constitute a spike-train. Some packets in the spike-train may be lost at the tail of the queue.

Another possibility is periodic processing of routing informations. In general, routers exchange routing information with each other and update their own routing tables based on the information periodically. This processing can consume CPU times and may halt packet forwarding. However, these two remain only possibilities and need to be investigated further.

4. Related Work

Kim, et al.[6] proposed a scheme named LIMD/H. In LIMD/H, a sender maintains the history of losses and distinguishes between congestion-related and non-congestion-related packet losses. The distinction is reflected in change of a decrease factor of AIMD. In[1], a TCP receiver distinguishes between the two using inter-arrival times. These approaches aim at less than 1-s rate. In contrast, we aim at a control at 10-s rate that prevents a receiver from sending frequent feedback to a sender.

Additionally, there have been several efforts to introduce delay into congestion control. Jain[5] advocated congestion avoidance based on a change in RTT. This work provides a good starting point for a consideration of delay. In TCP Vegas[3], RTT is used not only to adjust timeout values, but also to estimate an expected throughput. The difference between an actual throughput and an expected one provides a base for rate control. However issues of stability and coexistence among other TCP variants are still being studied. LDA[10] uses RTT for its control, but LDA only obtains samples of RTT by exchanging RTCP messages and the sampled RTT values do not contain dynamic behaviors of the delay. Bolot, in the literature[2], studied properties of RTT on the Internet with UDP probing packets at a regular interval. This work describes several significant findings: the relationship between the interval and two consecutive RTT values, and existence of probe compression. The probe compression is similar to a spike-train; our study does not use packets only for probing and focuses on ROTT instead of RTT. This work suggests that spike-trains do not appear when the regular interval is sufficiently large. We think that, unlike in probing, a CM UDP flow that attempts to obtain as much bandwidth as possible is likely to encounter a spike-train.

5. Conclusion

This paper has described the experimental results of measuring per-packet ROTT of a UDP stream. We have measured ROTT over two paths, Japan–U.S.A. and Japan–Spain, and find that a spike-train provides a congestion signal.

It is also found that reducing the transmission rate does not effectively result in decrease of packet loss ratio. This suggests that an algorithm that decreases the transmission rate based on packet loss ratio alone is not effective under some level of rate. Instead of packet losses, ROTT is observed to be effective for extracting congestion signals. For a future work, we plan to investigate the cause of spike-trains and also to establish a congestion control scheme based on measuring ROTT.

References

1) Biaz, S. and Vaidya, N.H.: Discriminating congestion losses from wireless losses using inter-arrival times at the receiver, *IEEE Symp. ASSET'99* (1999).

2) Bolot, J.-C.: End-to-end packet delay and loss behavior in the Internet, *Proc. ACM SIGCOMM'93*, pp.289–298 (Sep. 1993).

3) Brakmo, L.S., O'Malley, S.W. and Peterson, L.L.: TCP Vegas: New techiniques for congestion detection and avoidance, *Proc. ACM SIGCOMM'94*, pp.24–35 (Sep. 1994).

4) DeSimone, A., Chuah, M.C. and Yue, O.C.: Throughput performance of transport-layer protocols over wireless LANs, *Proc. GLOBECOM'93* (Dec. 1993).

5) Jain, R.: A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks, *ACM Computer Communication Review*, Vol.19, No.5, pp.56–71 (1989).

6) Kim, T., Lu, S. and Bharghavan, V.: Improving congestion control performance through loss differentiation, *Proc. IEEE ICCCN'99* (Oct. 1999).

7) Moon, S.B., Skelly, P. and Towsley, D.: Estimation and removal of clock skew from network delay measurements, *Proc. INFOCOM'99*, pp.227–234 (Mar. 1999).

8) Padhye, J., Kurose, J., Towsley, D. and Koodli, R.: A model based TCP-friendly rate control protocol, *Proc. 9th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp.137–151 (June 1999).

9) Rejaie, R., Handley, M. and Estrin, D.: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet, *Proc. INFOCOM'99* (Mar. 1999).

10) Sisalem, D. and Schulzrinne, H.: The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme, *Proc. 8th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp.215–

226 (Jul. 1998).
11) Tobe, Y., Tamura, Y., Nishino, H. and Tokuda, H.: Rate probing based adaptation at end hosts, *Proc. IEEE Workshop on QoS Support for Real-Time Internet Applications*, pp.58–65 (June 1999).
12) Vicisano, L., Rizzo, L. and Crowcroft, J.: TCP-like congestion control for layered multicast data transfer, *Proc. INFOCOM'98* (Apr. 1998).

Appendix: ISCM

As a previous work, Moon, et al. proposed the application of linear programming to remove clock skew [7]. We call their scheme the "MST scheme." Although different notation is used in the MST scheme, the basic idea of the MST scheme can be expressed in our notation and we translate the MST scheme into our terms.

Let us reformulate the skew problem. Let $\tilde{t}_s(i)$ and $\tilde{t}_r(i)$ be as follows:

$\tilde{t}_s(i) = t_s(i) - t_s(1)$

$\tilde{t}_r(i) = t_r(i) - t_r(1)$

Then, $\Delta t_{rs}(i) = \tilde{t}_r(i) - \tilde{t}_s(i)$. Let $\Delta T_{rs}(i)$ represent the OTT without the clock skew. $\Delta T_{rs}(i)$ can be expressed as follows:

$\Delta T_{rs}(i) = \Delta t_{rs}(i) - \alpha \tilde{t}_s(i) - \beta,$

where α and β are a coefficient related to the skew and an offset value, respectively.

Let N denote the total number of samples. If we apply the MST scheme to the above reformulated representation, the problem is described as follows.

(1) **Obtain** values of α and β
(2) such that they **minimize**

$$\sum_{i=1}^{N}(\Delta t_{rs}(i) - \alpha \tilde{t}_s(i) - \beta)$$

(3) **subject to**

$\Delta t_{rs}(i) - \alpha \tilde{t}_s(i) - \beta \geq 0$

The objective of ISCM is to obtain the following $\Delta T_{rs}(i)$:

$\Delta T_{rs}(i) = \Delta t_{rs}(i) - \alpha \tilde{t}_r(i) - \beta$

$\tilde{t}_r(i)$ instead of $\tilde{t}_s(i)$ is used since it is the measured at the receiver.

We make the MST scheme the starting points of ISCM. However, we modify the scheme. First, ISCM enhances the estimate of the base delay incrementally on arrival of packets rather than calculating the skew after a certain period. Second, ISCM is simplified such that convex points that are under $(\tilde{t}_r(i), \Delta t_{rs}(i))$ plots are searched. The resulting line of base delay is a line that traverses two convex points. The two convex points are chosen such that the distance on the $\tilde{t}_r(i)$ axis becomes the largest (**Fig. 19**).

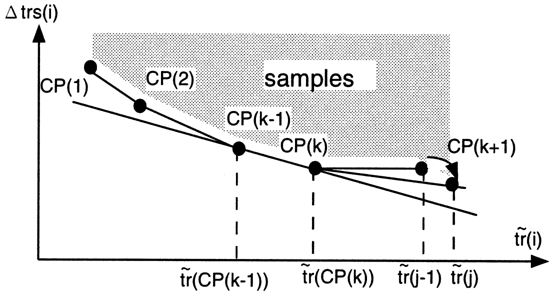

Fig. 19 Seeking and connecting convex points.

Let $CP(k)$ represent i at the k-th convex point. The following two variables are defined.

$$s(i_1, i_2) = \frac{\Delta t_{rs}(i_2) - \Delta t_{rs}(i_1)}{\tilde{t}_r(i_2) - \tilde{t}_r(i_1)}$$

$$f(i_1, i_2, t) = s(i_1, i_2)(t - \tilde{t}_r(i_1)) + \Delta t_{rs}(i_1)$$

With these variables, a pseudo-code for ISCM is shown in **Fig. 20**. Note that there is no variable for the total number of received packets.

```
========================
Initialization:
CP(1) = 1; CP(2) = 2; k = 2;

On arrival of i-th (i ≥ 3) packet:
if (t̃_r(i)−t̃_r(i−1) is not within an expected
        range)
        ignore the packet;
if (Δt_rs(i) < f(CP(k − 1), CP(k), t̃_r(i))) {
    find l such that s(CP(l − 1), CP(l)) ≤
        s(CP(l − 1), i) < s(CP(l), CP(l + 1));
        CP(l + 1) = i;
        k = l + 1;
    }
else if (CP(k) == i − 1) CP(k + 1) = i;
else if (Δt_rs(i) < f(CP(k), CP(k + 1), t̃_r(i)))
        CP(k + 1) = i;
if (t̃_r(CP(k)) < t̃_r(i) − t̃_r(CP(k))) {
        k = k + 1;
        cp[k + 1] = i;
    }
========================
```

Fig. 20 Pseudo-code for ISCM.

With this ISCM, $\Delta T_{rs}(i)$ is obtained by approximation as follows:

$\Delta T_{rs}(i) = \Delta t_{rs}(i) - f(CP(k-1), CP(k), \tilde{t}_r(i))$

The value of s eventually converges to α. However, even before it converges to a certain value, $\Delta T_{rs}(i)$ is calculated.

(Received August 20, 2001)
(Accepted October 16, 2001)

Editor's Recommendation

The authors describe an experimental method of measuring per-packet relative one-way trip time (ROTT) of a UDP stream using Incremental Skew Calculation (ISCM). They find that the spikes with successive plots (called spike-trains) often appear on a time-ROTT graph and also find that the congestion-related losses is strongly correlated to the spike-trains from several experimental results. There are still many issues in the research of Internet performance evaluation, this paper is expected to give the penetrating ideas in the research of the congestion control of Internet.

(Chairman of SIGDPS, Hiroshi Miyabe)

Yoshito Tobe received his B.E. and M.E. degrees in electrical engineering from the University of Tokyo in 1984 and 1986, and his M.S. degree in electrical and computer engineering from Carnegie Mellon University in 1992, respectively; Ph.D. degree in Meia and Governance from Keio University in 2000. He is an associate professor at graduate school of Media and Governance, Keio University since 2000. His current interests include protocols for home-area networks, traffic analysis of the Internet, and wireless mobile communications. He is a member of the ACM, the IEEE, and the IEICE.

Hiroto Aida received his B.S. degree in environmental information from Keio University in 2000. He is a master's student at graduate school of Media and Governance, Keio University. He is currently studying wireless packet scheduling and mobile communications.

Yosuke Tamura received his B.S. degree in environmental information from Keio University in 1997. He received his M.A. and Ph.D degrees in Media and Governance from Keio University in 1999 and 2001, respectively. He is a visiting researcher at Keio University, where he is currently studying flow control for internetworked data communications. He is a member of the IEEE Computer Society, and the IEICE.

Hideyuki Tokuda received his B.S. and M.S. degrees in electrical engineering from Keio University in 1975 and 1977, respectively; Ph.D. degree in computer science from the University of Waterloo in 1983. He joined the School of Computer Science at Carnegie Mellon University in 1983, and is an Adjunct Associate Professor from 1994. He joined the Faculty of Environmental Information at Keio University in 1990, and is a professor since 1996. His current interests include distributed operating system and computer networks. He is a member of the ACM, the IEEE, the IEICE, and the JSSST.