

Applying Lomb Periodogram to Round-trip Time Estimation from Unidirectional Packet Traces with Different TCP Congestion Controls

Toshihiko Kato, Xiaofan Yan, Ryo Yamamoto, and Satoshi Ohzahata

University of Electro-Communications

Tokyo, Japan

e-mail: kato@is.uec.ac.jp, yanxiaofan@net.is.uec.ac.jp, ryo_yamamoto@is.uec.ac.jp, ohzahata@is.uec.ac.jp

Abstract—Network operators often attempt to analyze traffic in the middle of their networks for various purposes. In such traffic analysis, the estimation of Round-Trip Time (RTT) is indispensable. Primarily, the RTT estimation is performed by consulting the relationship between a request and its response, such as a data segment and the associated ACK segment. However, in the middle of Internet, it is common that a network operator monitors traffic only in one direction. In such a case, an operator is required to estimate RTT from unidirectional packet traces. So far, several methods have been proposed for RTT estimation from unidirectional traces. In this paper, we adopt the Lomb periodogram method and apply it to various TCP traces, collected through Ethernet or wireless LAN, with different congestion control algorithms. As a result, the method can estimate RTT roughly, but the results are not accurate enough for subtle analysis, such as congestion window estimation.

Keywords- Unidirectional Packet trace; Round-trip Time; Lomb Periodogram; Congestion Control.

I. INTRODUCTION

Traffic analysis in the middle of Internet is an important issue for network operators. It can be applied the traffic classification, the traffic demand forecasting, and the malicious traffic detection. In the previous paper, we proposed a method to infer TCP congestion control algorithm from passively collected packet traces [1]. It adopts the following approaches.

- (1) Focus on a specific TCP flow using source/destination IP addresses and ports.
- (2) From the mapping between data segments and acknowledgment (ACK) segments, estimating Round-Trip Time (RTT) of the focused flow.
- (3) Estimate a congestion window size (cwnd) from the data size transferred during one RTT.
- (4) Obtain a sequence of cwnd values, and calculate a sequence of cwnd difference between adjacent cwnd values (we call Δ cwnd).
- (5) From the mapping between cwnd and Δ cwnd, infer a congestion control algorithm for the TCP flow.

This method requires a bidirectional trace to obtain both data and ACK segments.

In actual networks, however, it is often possible that only unidirectional traces are collected in the middle of networks. In this case, the above method cannot be applied. So, in another previous paper, we tried to modify the above method to infer TCP congestion control algorithms from unidirectional traces [2]. In the modified method, a fixed time

duration is used instead of RTT, and data size transferred during this duration was handled as cwnd. As a result, congestion control algorithms were estimated in some cases, but not in other cases. This is because our method depends largely on RTT value.

On the other hand, the estimation of RTT from traces has been actively studied and there are several proposals [3]-[6]. The RTT estimation methods proposed so far are classified into three categories. One is a method called Data-to-ACK-to-Data, which measures time between a data segment and the data segment sent just after the first data segment is ACKed [3]-[5]. This requires bidirectional packet traces and our first paper used it. Next is a method based on the autocorrelation [4][5]. This method counts the number of data segments in a short interval, and makes an array of counts indexed by the normalized interval. Then, it calculates the autocorrelation over the array and takes the maximum as a RTT. This method can be applied to unidirectional packet traces. The third one is use of spectral analysis [5][6]. A sequence of data segments are handled as a pulse function of time, which takes 1 when there is a data segment. Then, the frequency characteristic of this function is analyzed and the inverse of first harmonic is taken as RTT. Since the interval of data is irregular, the special analysis is performed by the Lomb periodogram [7].

For the purpose of precise RTT estimation from unidirectional traces, we adopt the third method because it can work for various type of traffic [5]. In this paper, we apply the Lomb periodogram to the RTT estimation from unidirectional packet traces, which we used in our previous papers, including different TCP congestion control algorithms, and discuss the results in detail. The rest of this paper is organized as follows. Section II explains the related work, including the problems we suffered from in our second previous paper [2] and the conventional RTT estimation methods. Section III describes a detailed scheme to estimate RTT using the Lomb periodogram. Section IV gives the results of RTT estimation for different TCP congestion control algorithms. In the end, Section V concludes this paper.

II. RELATED WORK

A. Problems on congestion window size estimation from unidirectional traces

In our previous papers [1][2], we collected packet traces in the configuration shown in Figure 1. A TCP data sender is connected with a bridge through 100 Mbps Ethernet. The bridge inserts 100 msec RTT (50 msec delay for each direction) and 0.01% packet losses. The bridge is connected with a TCP data receiver through IEEE 11g wireless LAN

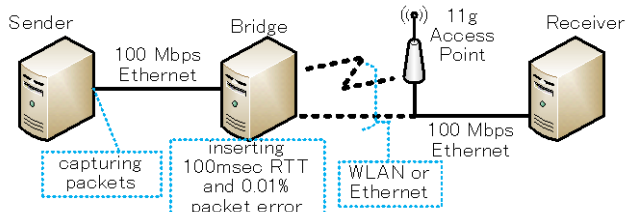


Figure 1. Experiment configuration.

(WLAN) or 100 Mbps Ethernet. The packet trace is collected at the TCP sender side. The collected traces include bidirectional ones, and in the unidirectional analysis, we picked up only data segments from the TCP sender to the TCP receiver.

Figures 2 and 3 show the results for CUBIC TCP [8] and TCP Vegas [9]. In the analysis a from bidirectional trace, $cwnd$ and $\Delta cwnd$ are estimated in the way described in Section I, and their relationship is given in the figures (by blue dots). In the analysis from a unidirectional trace, we assumed that RTT is 100 msec. The data size transferred during 100 msec and its difference are called $sentData$ and $\Delta sentData$, respectively, and shown in the figures by orange dots. In the case of CUBIC TCP, both results show the similar graph, which is a function in the form of $(\sqrt[3]{cwnd})^2$ with decreasing and increasing parts [1]. This result means that the unidirectional analysis works well. In the case of TCP Vegas, however, the results for bidirectional analysis and unidirectional analysis are significantly different. According

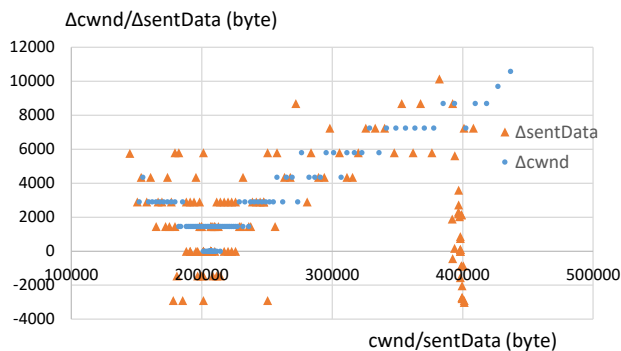


Figure 2. Result for CUBIC TCP [1][2].

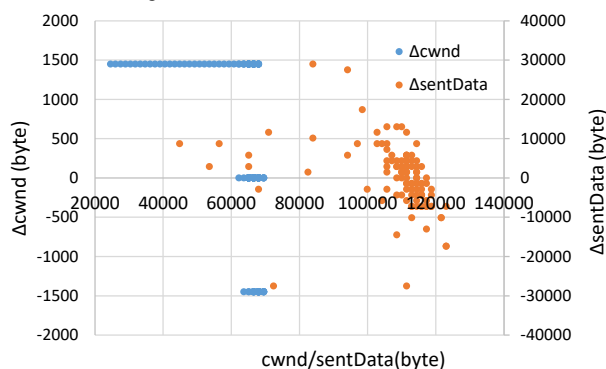


Figure 3. Result for TCP Vegas [1][2].

to the Vegas algorithm, $\Delta cwnd$ takes 1,460 bytes (one segment size), 0, or -1,460 bytes independently of $cwnd$ values, which is represented by the blue dots [1]. But, in the result for unidirectional analysis, the $\Delta sentData$ values indicated by the orange dots are unstable. So, the unidirectional analysis does not work well.

In our experiment, the trace for CUBIC TCP is collected in the configuration that uses Ethernet between the bridge and the TCP receiver, and that for TCP Vegas is collected by use of WLAN. This is one of the reasons. Figure 4 shows examples of the time variation of TCP sequence number for CUBIC TCP and TCP Vegas. In the case of CUBIC TCP, data segments are transferred in groups and there are idle time periods without any data transmissions. Therefore, in the unidirectional analysis, a sequence of data segments sent within a congestion window can be traced by use of 100 msec, which is a RTT determined tentatively. But, in the case of TCP Vegas, data segments are transmitted contiguously, and therefore, if RTT is not estimated correctly, a $sentData$ value does not match the real $cwnd$ value.

There considerations mean that the RTT estimation is critical for inferring TCP congestion control algorithms.

B. Related work on RTT estimation

As described in Section I, the RTT estimation methods are classified into three categories; the Data-to-ACK-to-Data method, the autocorrelation based method, and the spectral analysis method.

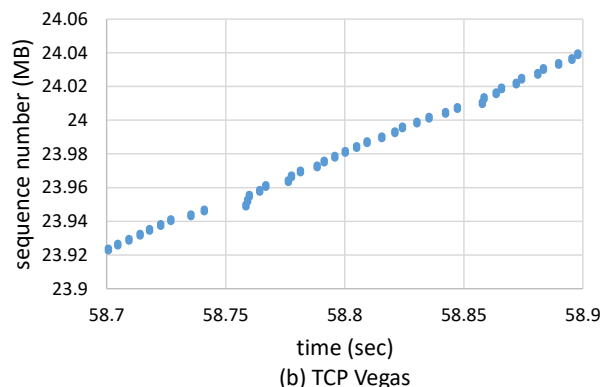
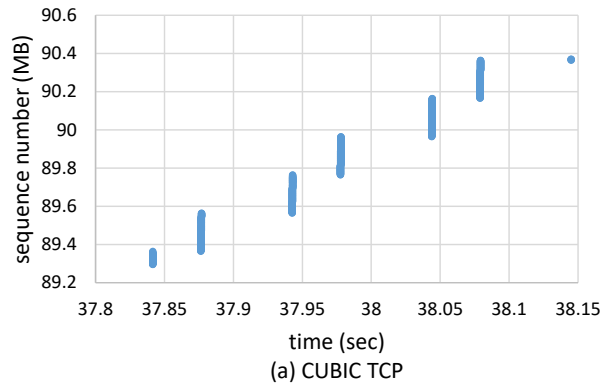


Figure 4. Sequence number vs. time.

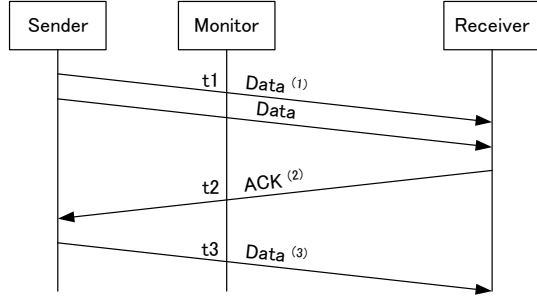


Figure 5. Data-to-ACK-to-Data method.

The Data-to-ACK-to-Data method is illustrated in Figure 5. Since there is some transmission delay between a TCP data sender and a monitor capturing packet traces, the following procedure is used to estimate RTT between sender and receiver. (1) A monitor focuses on a data segment, and remembers the time (t1). (2) A monitor catches the ACK segment that acknowledges the data segment. (3) A monitor detects the data segment sent by the sender just after the ACK segment in (2), and remember the time (t2). (4) $t2 - t1$ is a RTT for this moment. In order to detect data segment (3), the TCP time stamp option is used.

In the autocorrelation based method, the RTT estimation is performed once per measurement interval T . An array $P[n]$ maintaining the count of data segments is prepared using unit time Δt , where n is ranging from 0 to $T/\Delta t - 1$. If a data segment is detected at an interval $[start\ time + m \cdot \Delta t, start\ time + (m + 1) \cdot \Delta t)$, one is added to $P[m]$. For all the data segments from $start\ time$ to $start\ time + T$, the array $P[n]$ is arranged. After that, the autocorrelation function is defined as

$$A(l) = \frac{1}{T/\Delta t - l} \sum_{j=1}^{T/\Delta t - l} P[j] \cdot P[j + l]. \quad (1)$$

for lags $l = 0 \dots T/\Delta t - 1$. RTT is computed as $\max(A)$. This method can be applied to the unidirectional analysis, and will work well for the cases that data segments are distributed unevenly in a trace, such as the case of CUBIC TCP in Figure 4. However, for an evenly distributed trace, such as the case of TCP Vegas in Figure 4, it is concerned that RTT cannot be estimated correctly.

The spectral analysis method will be the most promising for RTT estimation among the three methods. Traditional spectral analysis, such as Fast Fourier Transform (FFT) assume that time domain data are regularly sampled [10]. However, in the RTT estimation, the time domain data is packet inter-arrival time of a specific flow. This data is sampled at each data packet capturing. This means that the time domain data in this case is irregularly sampled. In the case of the spectral analysis for irregularly sampled data, the Lomb periodogram is commonly used [6]. The details are shown in the next section.

III. RTT ESTIMATION USING LOMB PERIDGRAM

In the RTT estimation based on the Lomb periodogram, time sequence $\{t_i\}$ ($i = 1, \dots$) is considered as an input, where t_i corresponds to one data segment capturing time. At a specific time t_k , the frequency characteristic of this time

sequence is calculated using N time samples t_{k-N+1}, \dots, t_k in the following way ($N > k$) [6].

- The minimum and maximum frequencies of the range for power spectrum are defined as

$$f_k^{min} = \frac{1}{t_k - t_{k-N+1}} \text{ and } f_k^{max} = \frac{N}{2} f_k^{min}.$$

Accordingly, the power spectrum is calculated for angular frequency

$$\omega_i = 2\pi f_k^{min} + i\Delta_\omega \quad (i = 0, \dots, 2N - 1),$$

where $\Delta_\omega = 2\pi \frac{f_k^{max} - f_k^{min}}{2N}$.

- The power spectrum at angular frequency ω_i is defined as

$$P_k^N(\omega_i) = \frac{1}{2\sigma_k^2} \left\{ \frac{[\sum_{j=0}^{N-1} (h_{k-j} - \bar{h}_k) \cos \omega_i (t_{k-j} - \tau_k)]^2}{\sum_{j=0}^{N-1} \cos^2 \omega_i (t_{k-j} - \tau_k)} + \frac{[\sum_{j=0}^{N-1} (h_{k-j} - \bar{h}_k) \sin \omega_i (t_{k-j} - \tau_k)]^2}{\sum_{j=0}^{N-1} \sin^2 \omega_i (t_{k-j} - \tau_k)} \right\} \quad (2)$$

where \bar{h}_k and σ_k^2 are the mean and variance of N samples of h_k :

$$\bar{h}_k = \frac{1}{N} \sum_{j=0}^{N-1} h_{k-j} \quad (3)$$

$$\sigma_k^2 = \frac{1}{N-1} \sum_{j=0}^{N-1} h_{k-j}^2 - \frac{N}{N-1} \bar{h}_k^2, \quad (4)$$

and where is the solution of:

$$\tan(2\omega_i \tau_k) = \frac{\sum_{j=0}^{N-1} \sin 2\omega_i t_{k-j}}{\sum_{j=0}^{N-1} \cos 2\omega_i t_{k-j}}. \quad (5)$$

From the $2N - 1$ power spectrum values specified in an $\omega - P(\omega)$ plane, local maximum values are calculated. Among the frequencies generating local maximum power spectrum values, the fundamental frequency f_0 is estimated under the condition that other frequencies generating local maximum values are multiples of f_0 . At last, $T = 1/f_0$ is the estimated RTT.

IV. RESULTS OF APPLYING PERIDGRAM TO VARIOUS CONGESTION CONTROL ALGORITHMS

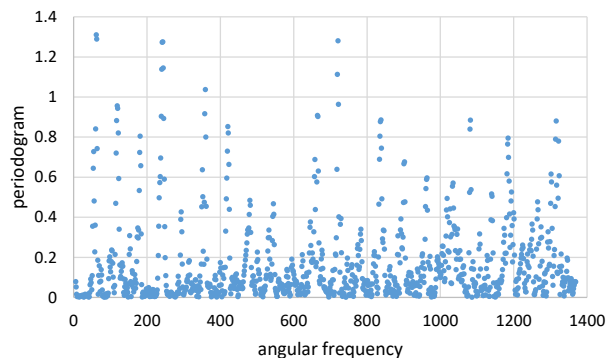
This section describes the results of RTT estimation for various types of TCP traces with different congestion control algorithms. We use the packet traces used in our previous papers [1][2]. As described in Section II.A, these traces are collected at the sender side in the configuration shown in Figure 1. Since packet losses are inserted at the bridge, we picked up a part of packet traces where no packet losses are detected, that is, where the sequence number of TCP segments keeps increasing. The traces themselves have bidirectional packet information and only the capturing time of data segments is extracted to build unidirectional traces. Together with the extraction, the real RTT is calculated from the mapping between data segments and ACK segments.

A. Result for traces including TCP Reno

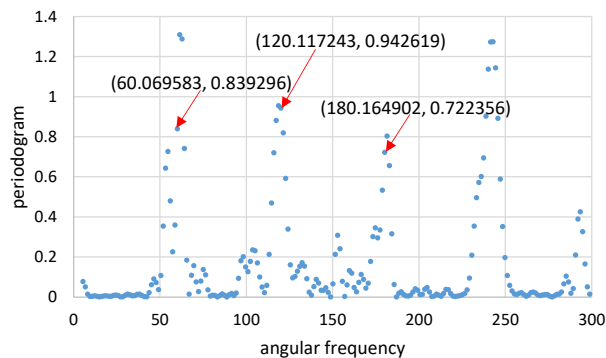
TCP Reno is a classic congestion control method which adopts an additive increase and multiplicative decrease (AIMD) algorithm. Here, $cwnd$ is increased each time the TCP sender receives an ACK segment acknowledging new data. The increase is $\frac{1}{cwnd}$ segments during the congestion avoidance phase, and as a result, $cwnd$ is expected to be increased by one segment during one RTT.

The Reno packet trace we used here is collected in the network configuration with Ethernet (see Figure 1), and we picked up a part from 27.010458 sec. to 45.99513 sec. in the trace, where there no retransmissions are detected for 7068 data segments. We used $N = 500$ in calculating the Lomb periodogram.

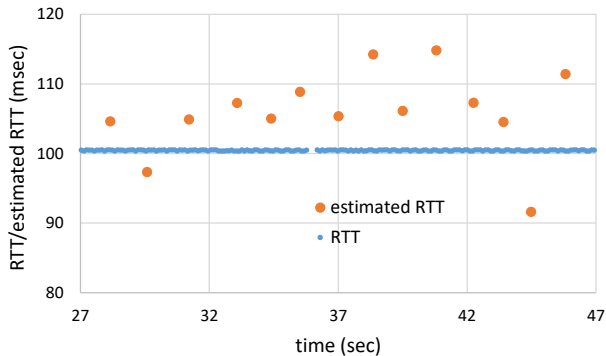
Figure 6 shows a result of RTT estimation from the Reno trace. Figure 6(a) is the result for periodogram at time 28.156143 sec. The horizontal axis is an angular frequency and the vertical axis is a periodogram. This figure shows there are several peaks periodically. Figure 6(b) zooms up the low angular frequency part of Figure 6(a). It shows that there are harmonized frequencies such that there are large periodogram values at some frequencies which are integral multiple of a



(a) periodogram at time 27.03713 sec.



(b) zooming up low angular frequency part



(c) estimated RTT and actual RTT

Figure 6. RTT estimation from Reno trace.

specific frequency (fundamental frequency f_0). In Figure 6(b), angular frequencies 60.069583, 120.117243, and 180.164902 are those frequencies. From this result, we can conclude that $2\pi f_0 = 60.069583$. So, we obtain $f_0 = 9.56037123$ and $RTT = 1/f_0 = 0.10459845$ sec.

We conducted similar calculations for multiple points of time in the trace and obtained the estimated RTT as shown in Figure 6(c). This figure also gives actual RTT values obtained from data and ACK segments in the original trace information. This result says that, although the actual RTT is extremely stable at 100 msec, the estimated RTT includes some errors in the order of 10 msec. The reason that the actual RTT is stable is that this experiment is conducted through only Ethernet and that there are no large delay variations. However, the RTT estimation by use of the Lomb periodogram cannot reflect this situation.

B. Result for traces including CUBIC TCP

As described in Section II.A, CUBIC TCP defines $cwnd$ as a cubic function of elapsed time T since the last congestion event [8]. Specifically, it defines $cwnd$ by (6).

$$cwnd = C \left(T - \sqrt[3]{\beta \cdot \frac{cwnd_{max}}{C}} \right)^3 + cwnd_{max} \quad (6)$$

Here, C is a predefined constant, β is the decrease parameter, and $cwnd_{max}$ is the value of $cwnd$ just before the loss detection in the last congestion event. Comparing with TCP Reno, $cwnd$ increases faster in CUBIC TCP.

We estimated RTT from the unidirectional packet trace including only data segments with CUBIC TCP. The trace is collected in the configuration using only Ethernet. We picked up a part in the trace from 23.483123 sec. to 38.348383 sec. for the RTT estimation. By applying the Lomb periodogram similarly with the case of Reno, we obtained estimated RTT as shown in Figure 7. This figure also gives actual RTT values.

The results show that the actual RTT is stable at 100 msec. and, on the other hand, the estimated RTT changes a lot between 90 msec. and 140 msec. The fluctuation is larger for CUBIC than TCP Reno. Especially, the difference between the estimated RTT and the actual RTT becomes large when the time is between 36 sec. and 38 sec. During this period, the $cwnd$ value itself becomes large and the large $cwnd$ value may give some bad influence to the RTT estimation.

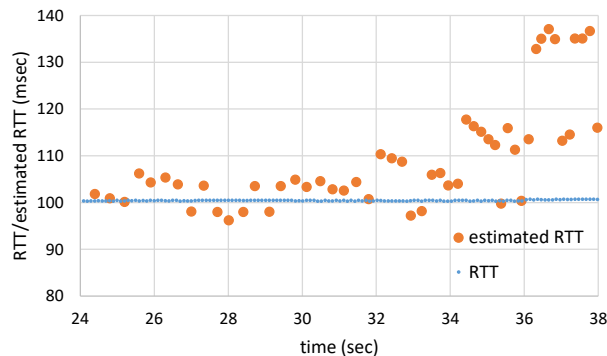


Figure 7. RTT estimation from CUBIC trace.

C. Result for traces including TCP Vegas

TCP Vegas estimates the bottleneck buffer size using the current values of $cwnd$ and RTT, and the minimal RTT for the TCP connection, according to (7) [9].

$$BufferSize = cwnd \times \frac{RTT - RTT_{min}}{RTT} \quad (7)$$

At every RTT interval, Vegas uses this $BufferSize$ to control $cwnd$ in the congestion avoidance phase in the following way.

$$\Delta cwnd = \begin{cases} 1 & (BufferSize < A) \\ 0 & (A \leq BufferSize \leq B) \\ -1 & (BufferSize > B) \end{cases} \quad (8)$$

Here, $A = 2$ and $B = 4$ (in unit of segment) are used in the Linux operating system.

We estimated RTT from the unidirectional packet trace including only data segments with TCP Vegas. In this case, in contrast with the above cases, the trace is collected in the configuration using WLAN. We picked up a part in the trace from 37.988347 sec. to 59.699611 sec. for the RTT estimation. By applying the Lomb periodogram to this time sequence, we obtained estimated RTT as shown in Figure 8, with actual RTT values.

In this case, the estimated RTT is stable around 100 msec, and on the other hand, the actual RTT values are scattered between 100 msec. and 140 msec. That is, although the actual RTT is changing, the RTT estimated by the Lomb periodogram does not follow the fluctuation. As we indicated in Section II.A, the timing of capturing data segments is almost uniformly distributed in this case. As a result, it is considered that the Lomb periodogram method cannot detect the actual RTT.

D. Result for traces including TCP Veno

TCP Veno (Vegas and ReNO) is an example of hybrid type congestion control method, considering packet losses and delay. It uses the $BufferSize$ in (7) to adjust the growth of $cwnd$ in the congestion avoidance phase as follows. If $BufferSize > B$ (B is the Vegas parameter B), $cwnd$ grows by $1/cwnd$ for every other new ACK segment, and otherwise, it grows in the same manner with TCP Reno. That is, when the congestion status is heavy, i.e., the bottleneck buffer size is large, the increasing rate of $cwnd$ is halved.

We estimated RTT from the unidirectional Veno trace captured in the WLAN configuration in Figure 1. We picked up a part in the trace from 37.684643 sec. to 52.653736 sec. including 23,360 data segments. By applying the Lomb periodogram to this sequence, we obtained estimated RTT as in Figure 9, which also gives the actual RTT.

Similarly with the case of TCP Vegas, the estimated RTT is rather stable around 100 msec., which is different from the actual RTT spreading in the range between 100 msec. and 130 msec.

V. CONCLUSIONS

This paper described the results of applying the Lomb periodogram method to estimating RTT from unidirectional

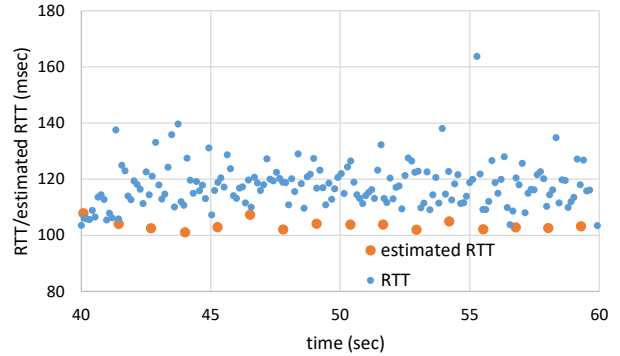


Figure 8. RTT estimation from Vegas trace.

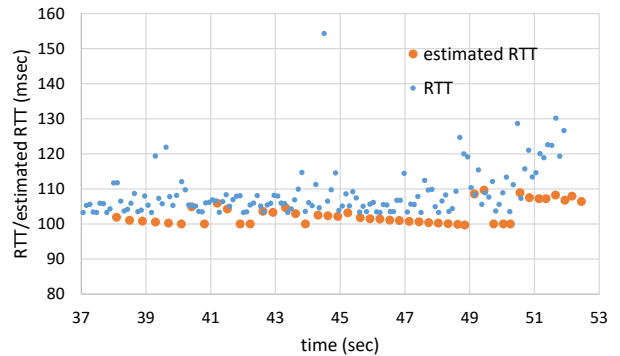


Figure 9. RTT estimation from VENO trace.

packet traces including TCP segments with different congestion control algorithms, TCP Reno, CUBIC TCP, TCP Vegas, and TCP Veno. Among them, the packet traces for TCP Reno and CUBIC TCP are collected in the network configuration using only Ethernet, and those for TCP Vegas and TCP Veno are from WLAN configuration. The performance evaluation gave the following results.

First of all, the Lomb periodogram method was possible to estimate an approximate RTT values from unidirectional packet traces. Strictly speaking, however, the estimated RTT values have some errors and they are not tolerable for the approaches that require accurate RTT estimation, such as our method to infer the TCP congestion algorithms from unidirectional packet traces [2]. Moreover, although the experiments adopted here added a fix delay, actual TCP communications suffer from variable delay like Bufferbloat [12]. So, the accurate estimation will be more difficult in real environments.

The second point is that the estimation is affected largely by the network configuration, such as with Ethernet or with WLAN. It is also affected somehow by the congestion control used in packet traces. In our experiment, the traces of TCP Reno and CUBIC TCP were collected in an Ethernet configuration. In this case, the actual RTT was stable and the estimated RTT was fluctuated. In the CUBIC TCP trace, where the congestion control is more aggressive, the errors of the estimated RTT increased. On the other hand, the traces of TCP Vegas and TCP Veno were collected in a WLAN configuration. In this case, while the actual RTT was

fluctuated, the Lomb periodogram method could not estimate this fluctuation and the estimated RTT was stable.

Since it is difficult to estimate RTT correctly from unidirectional packet traces, we need to develop a new method to infer TCP congestion control algorithms from unidirectional traces.

REFERENCES

- [1] T. Kato, A. Oda, C. Wu, and S. Ohzahata, "Comparing TCP Congestion Control Algorithms Based on Passively Collected Packet Traces," IARIA ICSNC 2015, pp. 135-141, Nov. 2015.
- [2] T. Kato, L. Yongxiale, R. Yamamoto, and S. Ohzahata, "How to Characterize TCP Congestion Control Algorithms from Unidirectional Packet Traces," IARIA ICIMP 2016, pp. 23-28, May 2016.
- [3] H. Jiang and C. Dovrolis, "Passive Estimation of TCP Round-Trip Times," ACM SIGCOMM Comp. Commun. Rev. vol. 32, issue 3, pp. 75-88, Jul. 2002.
- [4] B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times," Passive and Active Network Measurement, PAM 2005, LNCS, vol. 3431, pp. 121-134.
- [5] R. Lance and I. Frommer, "Round-Trip Time Inference Via Passive Monitoring," ACM SIGMETRICS Perf. Eval. Rev., vol. 33, issue 3, pp. 32-38, Dec. 2005.
- [6] D. Carra et al., "Passive Online RTT Estimation for Flow-Aware Routers Using One-Way Traffic," NETWORKING 2010 LNCS6091, pp. 109-121, 2010.
- [7] J. Scargle, "Statistical aspects of spacial analysis of unevenly spaced data," J. Astrophysics, vol 263, pp. 835-853, Dec. 1982.
- [8] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," ACM SIGOPS Op. Syst. Review, vol. 42, issue 5, pp. 64-74, Jul. 2008.
- [9] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," IEEE J. Sel. Areas Commun., vol. 13, no. 8, pp. 1465-1480, Oct. 1995.
- [10] S. Kay and S. Marple, "Spectrum analysis; A modern perspective," Proc. of the IEEE, vol. 69, issue 11, pp. 1380-1419, Nov. 1981.
- [11] C Fu and C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," IEEE J. Sel. Areas Commun., vol. 21, no. 2, Feb. 2003.
- [12] S. Strowes, "Passively Measuring TCP Round-trip Times," ACM Queue, vol. 11, issue 8, pp. 1-12, Aug. 2013.