A Study on Round-trip Time Estimation from Unidirectional Packet Traces Using Different TCP Congestion Control Algorithms

Toshihiko Kato, Xiaofan Yan, Ryo Yamamoto, and Satoshi Ohzahata
Graduate School of Informatics and Engineering
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. Our previous paper showed a result that adopts the Lomb periodogram method to various TCP traces using different congestion control algorithms. In this paper, we show the RTT estimation using the autocorrelation based method and the Lomb periodogram method from unidirectional TCP traces, collected through Ethernet or wireless LAN, using different congestion control algorithms, i.e., TCP Reno, CUBIC TCP, TCP Vegas, and TCP Veno. As a result, the autocorrelation based method could not estimate RTT correctly, the Lomb periodogram method provided reasonable estimation, and the results by the Lomb periodogram method are not accurate enough for subtle analysis, such as congestion window estimation.**

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

I. INTRODUCTION

This paper is an extension of our previous paper [1], which is presented in an IARIA conference.

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 [2]. 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 Δcwnd).
(5) From the mapping between cwnd and Δcwnd, 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 [3]. 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 [4]-[7]. 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 [4]-[6]. This requires bidirectional packet traces and our first paper used it. Next is a method based on the autocorrelation [5][6]. 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 [6][7]. 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 spectral analysis is performed by the Lomb periodogram [8].

In our previous paper [1], we picked up the third method for estimating RTT from unidirectional traces including different TCP congestion control algorithms, which we used for inferring congestion control algorithms [2][3]. In this paper, we add the results of RTT estimation by use of the second method, the autocorrelation based method, and discuss the results in more detail.

The rest of this paper is organized as follows. Section II explains the problems we suffered from in our previous work on estimating congestion window sizes from unidirectional packet traces [3]. Section III explains the conventional RTT estimation methods in detail. Section IV gives the results of RTT estimation for different TCP congestion control algorithms, using the autocorrelation based method and the Lomb periodogram method, and compare the results. In the end, Section V concludes this paper.

II. Related Work Problems on Congestion Window Size Estimation from Unidirectional Traces

A. *Problems on congestion window size estimation from unidirectional traces*

In our previous papers [2][3], 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 (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 [9] and TCP Vegas [10]. In the analysis a from bidirectional trace, cwnd and ⊿cwnd (both in bytes) 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 ⊿sentData (both in bytes), 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 $\left(\sqrt[3]{cnwd}\right)^2$ with decreasing and increasing parts [2]. 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 to the Vegas algorithm, ⊿cwnd takes 1,460 bytes (one segment size), 0, or -1,460 bytes independently of cwnd values, which is represented by the blue dots [2]. But, in the result for unidirectional analysis, the ⊿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

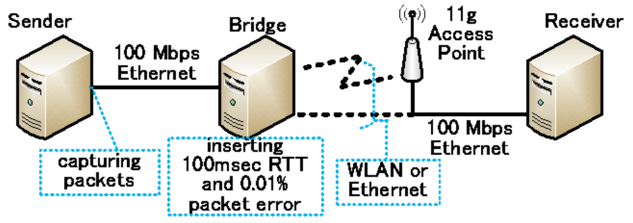


Figure 1. Experiment configuration.

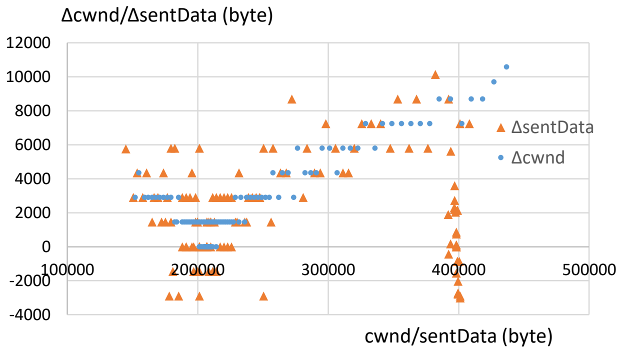


Figure 2. Result for CUBIC TCP [2][3].

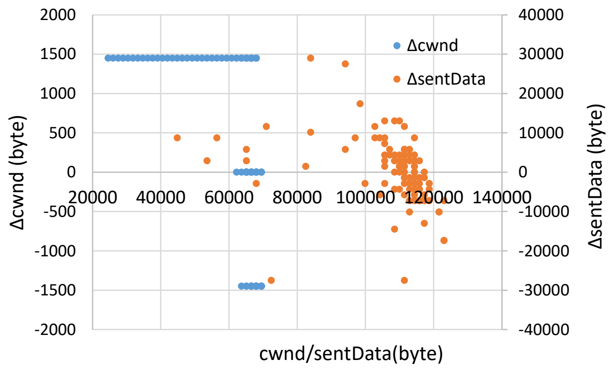


Figure 3. Result for TCP Vegas [2][3].

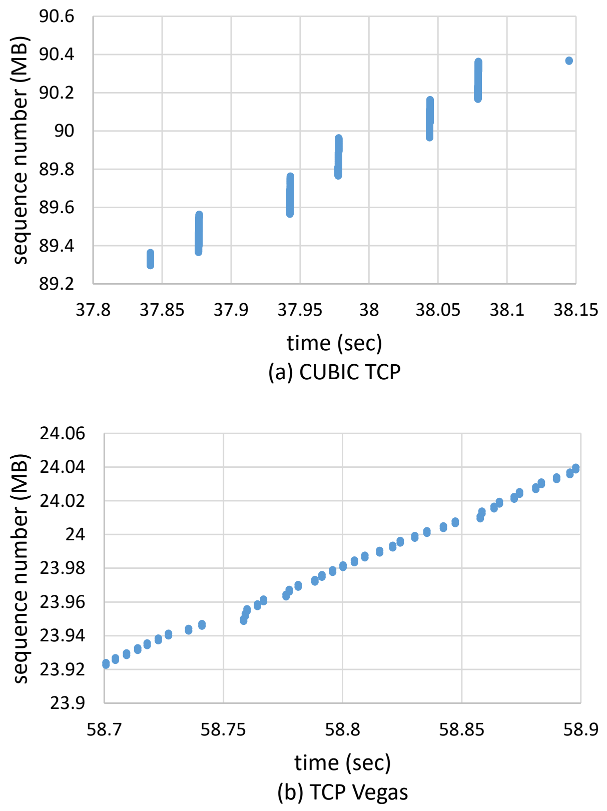


Figure 4. Sequence number vs. time.

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.

III. RTT Estimation Methods

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 Lomb periodogram method.

A. 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) t3 – t1 is a RTT for this moment. In order to detect data segment (3), the TCP time stamp option is used.

B. Autocorrelation based method

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

$$A(l) = \frac{1}{T/_{\Delta t} - l} \sum_{j=0}^{T/_{\Delta t} - l} P[j] \cdot P[j+l]. \qquad (1)$$

for lags $l = 0 \cdots T/_{\Delta t} - 1$. 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(a).

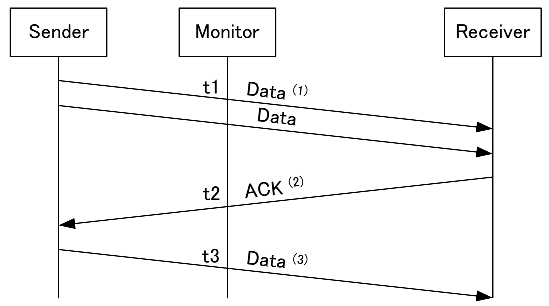


Figure 5. Data-to-ACK-to-Data method.

C. Lomb periodogram method

The last method is one based on the spectral analysis, in which a sequence of data segments are handled as a pulse time sequence, the frequency characteristic of this time sequence is analyzed, and the inverse of first harmonic is taken as RTT. Traditional spectral analysis, such as Fast Fourier Transform (FFT) assume that time domain data are regularly sampled [11]. 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 [8].

In the RTT estimation based on the Lomb periodogram, time sequence $\{t_i\}\ (i = 1, \cdots)$ 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 $t_{k-N+1}, \cdots t_k$ in the following way $(k > N)$ [7].

- 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
$$\omega_i = 2\pi f_k^{min} + i\Delta_\omega \ \ (i = 0 \cdots 2N-1),$$
where $\Delta_\omega = 2\pi \frac{f_k^{max} - f_k^{min}}{2N}$.
- The power spectrum at angular frequency ω_i is defined as
$$P_k^N(\omega_i) = \frac{1}{2\sigma_k^2} \left\{ \frac{\left[\sum_{j=0}^{N-1}(h_{k-j} - \bar{h}_k) cos\,\omega_i(t_{k-j} - \tau_k)\right]^2}{\sum_{j=0}^{N-1} cos^2\omega_i(t_{k-j} - \tau_k)} + \frac{\left[\sum_{j=0}^{N-1}(h_{k-j} - \bar{h}_k) sin\,\omega_i(t_{k-j} - \tau_k)\right]^2}{\sum_{j=0}^{N-1} sin^2\omega_i(t_{k-j} - \tau_k)} \right\} \qquad (2)$$
where $\bar{h}_k$ and σ_k^2 are the mean and variance of *N* samples of h_k:
$$\bar{h}_k = \frac{1}{N} \sum_{j=0}^{N-1} h_{k-j} \qquad (3)$$
$$\sigma_k^2 = \frac{1}{N-1} \sum_{j=0}^{N-1} h_{k-j}^2 - \frac{N}{N-1} \bar{h}_k{}^2, \qquad (4)$$
and where τ_k is the solution of:
$$tan(2\omega_i \tau_k) = \frac{\sum_{j=0}^{N-1} sin2\omega_i t_{k-j}}{\sum_{j=0}^{N-1} cos2\omega_i t_{k-j}}. \qquad (5)$$

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 RTT Estimation for 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 [2][3]. As described in Section II, 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 based on the Data-to-ACK-to-Data method.

A. Result for traces including TCP Reno

(1) Overview

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 no retransmissions are detected for 7068 data segments.

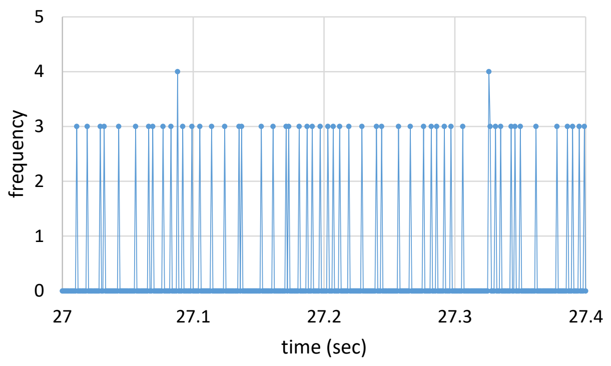
(2) Results of autocorrelation based method

In the autocorrelation based method, we adopted 1msec as the unit time (Δt). This means that we can estimate RTT in the order of mili seconds, which will be enough for discussing the estimation capability of this method. We adopted 400 msec as the measurement interval (T). By use of these values, the autocorrelation can be calculated with changing the time lag from 0 to 399. We picked up the autocorrelation in the range of $l = 1 \cdots 200$.

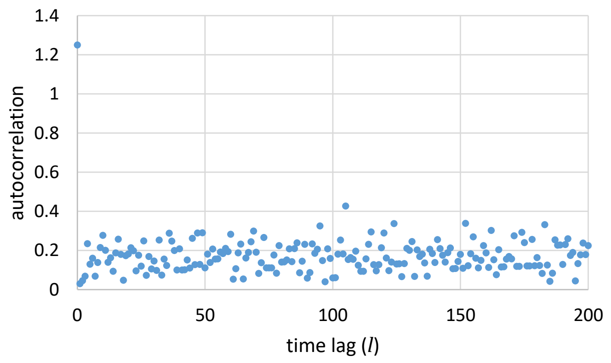
Figure 6 shows the results of the applying autocorrelation based method to the TCP Reno traced mentioned above. Figure 6(a) shows the histogram result between 27.0 sec and 27.4 sec. Three or four data segments are transferred within 1 msec interval. The intervals with data segments are repeated in a duration of a few msec and 20 msec. Figure 6(b) shows the result of autocorrelation for the data shown in Figure 6(a). The range of time lag is 0 through 200, but we use the range of $l = 1 \cdots 200$. From this result, we can select 105 as the lag value which generates the largest autocorrelation. So, we estimated that the RTT at time 27.0 sec is 105 msec. Similarly, we estimated RTT for every 400 msec from the obtained packet trace. Figure 6(c) shows the results. This figure also shows the actual RTT (indicated as RTT in the figure), the RTT estimated by the Data-to-ACK-to-Data method by use of bidirectional information contained in the original packet trace. The actual RTT is stable at 100 msec, but the estimated RTT changes between 0 msec and 200 msec although 60% of the results are close to 100 msec.

(3) Results of Lomb periodogram method

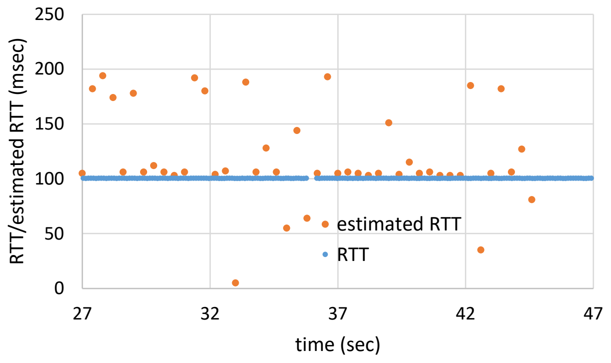
In order to apply the Lomb periodogram method, we need to decide the value of *N*. We used *N* = 500 in calculating the Lomb periodogram. Figure 7 shows a result of RTT estimation from the Reno trace. Figure 7(a) is the result for



(a) histogram of data segments between 27 sec and 27.4 sec



(b) autocorrelation for interval [27.0, 27.4]



(c) estimated RTT and actual RTT

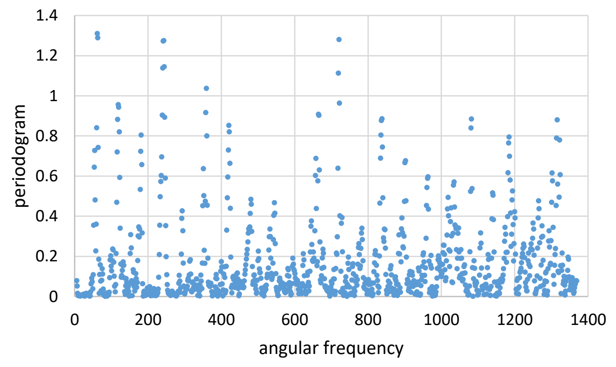
Figure 6. RTT estimation from Reno trace by autocorrelation.

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 7(b) zooms up the low angular frequency part of Figure 7(a). It shows that there are harmonized frequencies such that there are large periodogram values at some frequencies which are integral multiple of a specific frequency (fundamental frequency f_0). In Figure 7(b), angular frequencies 61.4343, 118.7525, and 181.5296 are those frequencies. From this result, we can conclude that $2\pi f_0 = 61.4343$. So, we obtain $f_0 = 9.77755$ and RTT $= {}^{1}/_{f_0} = 0.102275$sec.

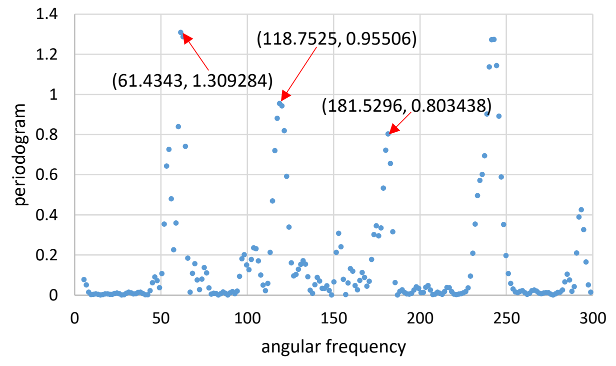
We conducted similar calculations for multiple points of time in the trace and obtained the estimated RTT as shown in

Figure 7(c). This figure also gives actual RTT values obtained from the relationship 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. When this result is compared with the result by the autocorrelation based method, that by the Lomb periodogram method is better than that by the autocorrelation based method.

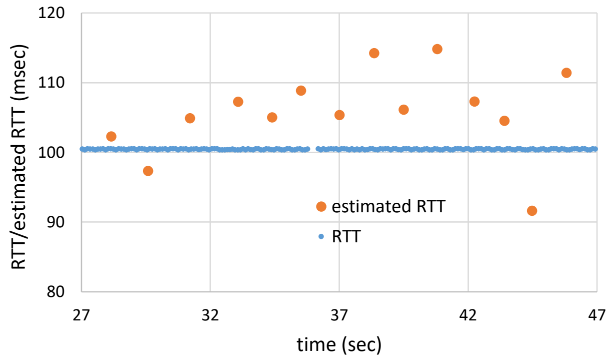
The reason that the actual RTT is stable is that this experiment is conducted through only Ethernet and that there are no large delay variations. However, the RTT estimation by use of neither the autocorrelation nor the Lomb periodogram can reflect this situation.



(a) periodogram at time 27.03713 sec



(b) zooming up low angular frequency part



(c) estimated RTT and actual RTT

Figure 7. RTT estimation from Reno trace by Lomb periodogram.

B. *Result for traces including CUBIC TCP*

(1) *Overview*

As described in Section II, CUBIC TCP defines *cwnd* as a cubic function of elapsed time T since the last congestion event [9]. Specifically, it defines cwnd by (6).

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

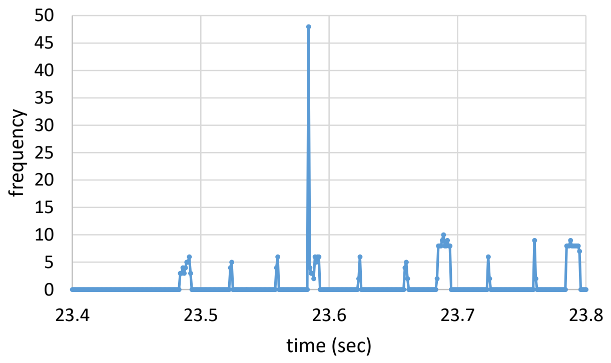
Here, C is a predefined constant, β is the decrease parameter, and $cwnd_{max}$ is the value of $cwnd$ just before the loss detection in the last congestion event. Comparing with TCP Reno, cwnd increases faster in CUBIC TCP.

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.

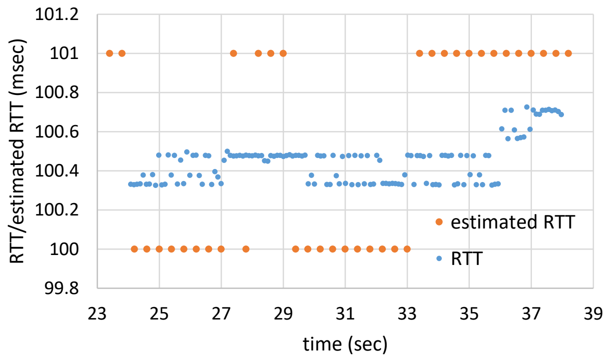
(2) *Results of autocorrelation based method*

For the autocorrelation based method, we used the same parameters as those for TCP Reno. That is, 1msec is as Δt, 400 msec is as T, and the autocorrelation is evaluated in the range of l is from 1 to 200.

The results are shown in Figure 8. Figure 8(a) shows the histogram result between 23.4 sec and 23.8 sec. Compared with the case of TCP Reno given in Figure 6(a), data segments are transmitted in a group in the case of CUBIC TCP. A



(a) histogram of data segments between 23.4 sec and 23.8 sec



(b) estimated RTT and actual RTT

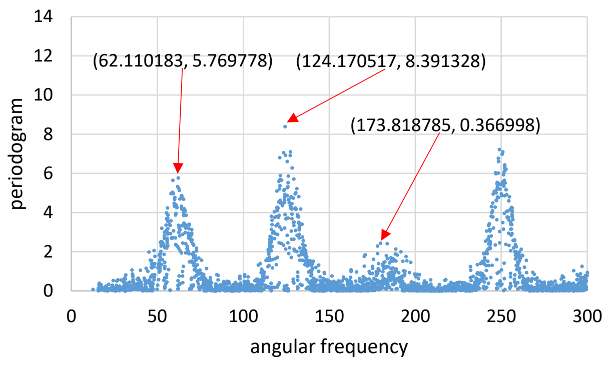
Figure 8. RTT estimation from CUBIC trace by autocorrelation.

portion where data segments are sent over multiple time slots is repeated every one hundred mili second, and so it is considered that the RTT can be easily estimated by the autocorrelation. Figure 8(b) shows the estimated RTT, together with the actual RTT. The estimated RTT takes either 100 msec or 101 msec, and the actual RTT takes values between them. Since the granularity of the estimated RTT is 1 msec, it can be said that the autocorrelation based method provides good estimation for CUBIC TCP.

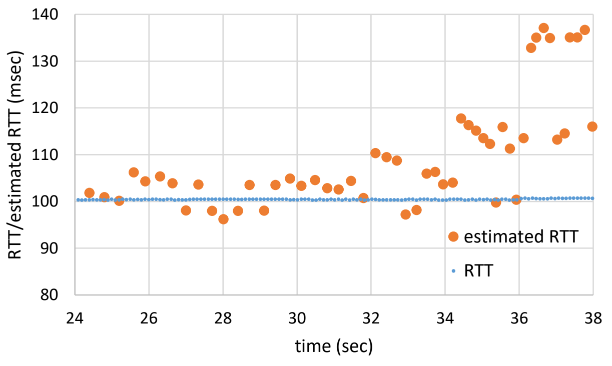
(3) Results of Lomb periodogram method

By applying the Lomb periodogram similarly with the case of Reno, we obtained estimated RTT as shown in Figure 8. Figure 9(a) shows the result for periodogram at time 23.987461 sec. In this figure, there are peaks of periodogram at angular frequencies of 62.110183, 124.170517, and so on. So, we estimated that the fundamental angular frequency is 62.110183, and calculated the estimated RTT at this timing.

Figure 9(b) shows the estimated RTT together with the 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.



(a) periodogram at time 23.987461 sec



(b) estimated RTT and actual RTT

Figure 9. RTT estimation from CUBIC trace by Lomb periodogram.

In the case of CUBIC TCP, the autocorrelation based method could provide more precise RTT estimation than the Lomb periodogram method. The reason is considered to be that the burstiness of the logged packet sequence is high in this case.

C. Result for traces including TCP Vegas

(1) Overview

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) [10].

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

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

$$\varDelta cwnd = \begin{cases} 1 & (BufferSize < A) \\ 0 & (A \leqq BufferSize \leqq B) \\ -1 & (BufferSize > B) \end{cases} \qquad (8)$$

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.

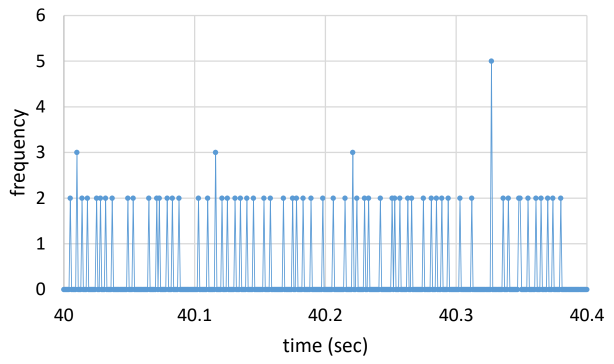
(2) Results of autocorrelation based method

Figure 10 shows the results of the RTT estimation by use of the same parameters used in the cases of TCP Reno and CUBIC TCP. Figure 10(a) shows the histogram result between 40.0 sec and 40.4 sec. In this case, the frequency is almost two packets and the timing when some packets are transmitted is scattered. That is, the packet transmission is not bursty for the Vegas packet trace used in this experiment. Figure 10(b) shows the estimated RTT and the actual RTT. As supposed from the result of the histogram, the estimated RTT is distributed between 0 mse and 200 msec. Since 200 msec is the upper bound in the estimation, it is said that the estimation here is not done well but the estimated RTT is randomly distributed.

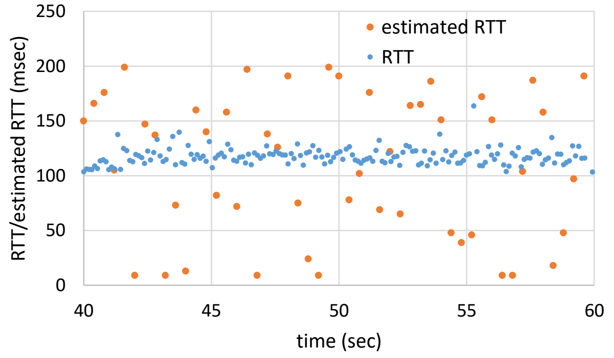
(3) Results of Lomb periodogram method

Figure 11 shows the result of the RTT estimation based on the Lomb periodogram method. Figure 11(a) is the periodogram at time 40.082778 sec. From this result, we obtain that the fundamental angular frequency is 58.288021 and the estimated RTT at this point is 107.795 msec.

Figure 11(b) shows the estimated RTT and actual RTT obtained for the part of the Vegas packet trace mentioned above. 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 Figure 4(b) and Figure 10(a), the timing of capturing data segments is almost uniformly distributed in this case. As a

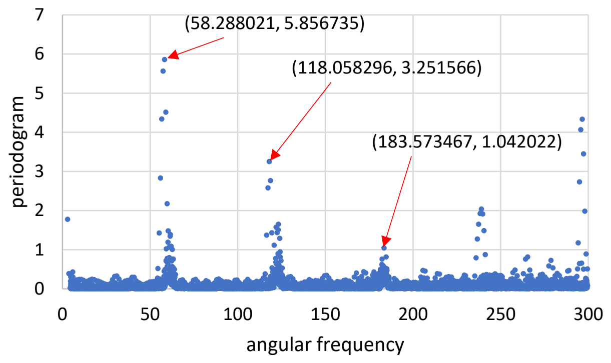


(a) histogram of data segments between 40.0 sec and 40.4 sec

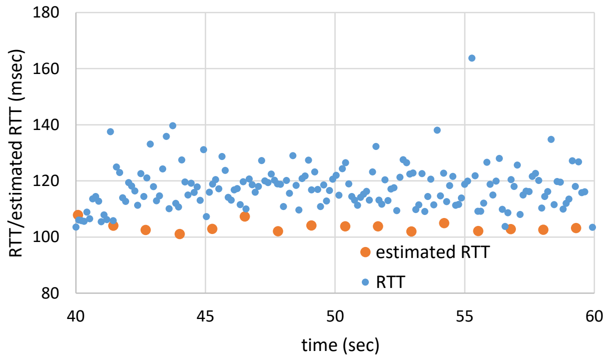


(b) estimated RTT and actual RTT

Figure 10. RTT estimation from Vegas trace by autocorrelation.



(a) periodogram at time 40.082778 sec



(b) estimated RTT and actual RTT

Figure 11. RTT estimation from Vegas trace by Lomb periodogram.

result, it is considered that the Lomb periodogram method cannot detect the actual RTT. Compared with the result of the autocorrelation based method, however, the Lomb periodogram method provides much better estimation in this case.

D. Result for traces including TCP Veno

(1) Overview

TCP Veno (Vegas and ReNO) [12] 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.

(2) Results of autocorrelation based method

Figure 12 shows the results of the RTT estimation by the autocorrelation for TCP Veno. Figure 12(a) is the histogram result between 38.0 sec and 38.4 sec. Similarly with the result for TCP Vegas shown in Figure 10(a), the data transmissions are scattered, that is, not bursty. The frequency of time slots with data transmission is two or three, and the intervals between those time slots are less than 20 msec. As a result, the estimated RTT values shown in Figure 12(b) are largely different from the actual RTT values. Especially, in the time frame later than 41.4 sec, the estimated RTT is 1 msec. In the Veno packet trace used here, the autocorrelation based method was very poor in the RTT estimation.

(3) Results of Lomb periodogram method

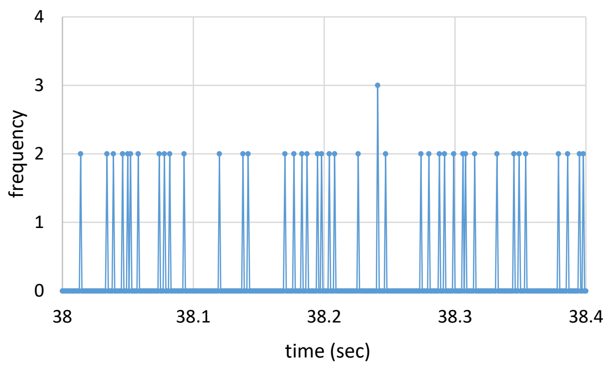
Figure 13 shows the results of the RTT estimation by the Lomb periodogram for TCP Veno. Figure 13(a) is the periodogram at time 38.090911 sec. This result indicates that the fundamental angular frequency is 63.281391 and that the estimated RTT at this point is 99.239 msec.

Figure 13(b) shows the estimated RTT and the actual RTT for TCP Veno. Similarly with the results for TCP Vegas, the estimated RTT values are stable around 100 msec, but the actual RTT has a distribution between 100 msec and 130 msec. In this sense, the Lomb periodogram method cannot estimate RTT in a strict sense, but it provides a reasonable estimation compared with the autocorrelation based method.

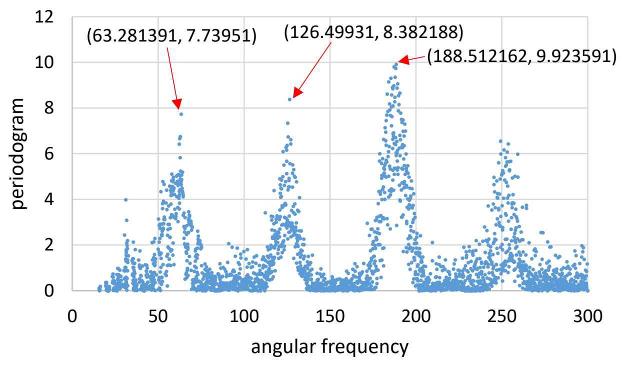
E. Discussions

Through the experiments described above, we obtained the following discussions.

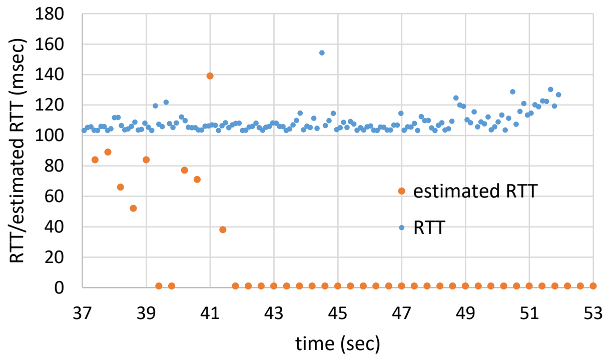
- First of all, the autocorrelation based method could not estimate RTT correctly in many cases. In our experiment, three cases out of four did not work well. In the case of



(a) histogram of data segments between 38.0 sec and 38.4 sec

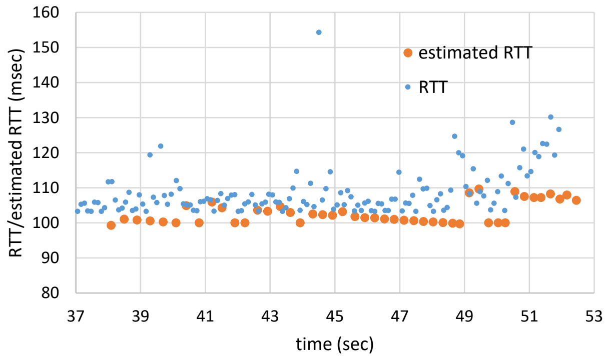


(a) periodogram at time 38.090911 sec



(b) estimated RTT and actual RTT

Figure 12. RTT estimation from Veno trace by autocorrelation.



(b) estimated RTT and actual RTT

Figure 13. RTT estimation from VENO trace by Lomb periodogram.

CUBIC TCP, this method could estimate RTT accurately. The reason is considered that data segments in unidirectional packet traces are transmitted in a burst and that the actual RTT is stable. In other cases, i.e., for TCP Reno, TCP Vegas, and TCP Veno, the data transmission is performed relatively in a uniform way, and this situation makes the RTT estimation difficult by the autocorrelation based method.

- Secondly, compared with the autocorrelation method, the Lomb periodogram method was possible to estimate 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 [3]. Moreover, although the experiments adopted here added a fix delay, actual TCP communications suffer from variable delay like Bufferbloat [13]. So, the accurate estimation will be more difficult in real environments.

- The third point is that the estimation by the Lomb periodogram method 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.

V. Conclusions

This paper described the results of applying the autocorrelation based method and the Lomb periodogram method to estimating RTT from unidirectional packet traces including TCP segments with different congestion control algorithms, TCP Reno, CUBIC TCP, TCP Vegas, and TCP Veno. The performance evaluation gave the following results.

First of all, the autocorrelation based method provided poor performance in the RTT estimation. Only CUBIC TCP in our experiment worked well, and the estimation for other three congestion control algorithms was not successful.

Secondly, the Lomb periodogram method was possible to estimate more accurate RTT values than the autocorrelation based method. However, the estimated RTT values were not accurate enough for the applications requiring precise RTT estimation, such as our method to infer the TCP congestion algorithms [3].

Lastly, we confirmed a tendency that the estimation by the Lomb periodogram is affected by the network configuration, such as with Ethernet or with WLAN. In the Ethernet configuration, the actual RTT is stable but the estimated RTT is fluctuated. In the WLAN configuration, the result is opposite.

In conclusion, it will be considered that the accurate RTT estimation will be difficult from unidirectional packet traces, although the rough estimation will be feasible by the Lomb periodogram method.

References

[1] T. Kato, X. Yan, R. Yamamoto, and S. Ohzahata, "Applying Lomb Periodogram to Round-trip Time Estimation from Unidirectional Packet Traces with Different TCP Congestion Controls," IARIA ICIMP 2018, pp. 1-6, Jul. 2018.

[2] 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.

[3] T. Kato, L. Yongxialee, R. Yamamoto, and S. Ohzahata, "How to Characterize TCP Congestion Control Algorithms from Unidirectional Packet Traces," IARIA ICIMP 2016, pp. 23-28, May 2016.

[4] 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.

[5] B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times," Passive and Active Nework Measurement, PAM 2005, LNCS, vol. 3431, pp. 121-134.

[6] R. Lance and I. Frommer, "Round-Trip Time Inference Via Pasive Monitoring," ACM SIGMETRICS Perf. Eval. Rev., vol. 33, issue 3, pp. 32-38, Dec. 2005.

[7] D. Carra et al., "Passive Online RTT Estimation for Flow-Aware Routers Using One-Way Traffic," NETWORKING 2010 LNCS6091, pp. 109-121, 2010.

[8] J. Scargle, "Statistical aspects of spacial analysis of unevently spaced data," J. Astrophysics, vol 263, pp. 835-853, Dec. 1982.

[9] 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.

[10] 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.

[11] S. Kay and S. Marple, "Spectrum analysis; A modern perspective," Proc. of the IEEE, vol. 69, issue 11, pp. 1380-1419, Nov. 1981.

[12] 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.

[13] S. Strowes, "Passively Measuring TCP Round-trip Times," ACM Queue, vol. 11, issue 8, pp. 1-12, Aug. 2013.