Considering Spurious Timeout in Proxy for Improving TCP Performance in Wireless Networks

YuChul Kim
Telecommunication R&D Center,
Samsung Electronics,Co., Ltd
yuchul.kim@samsung.com

DongHo Cho
Communication and Information Systems Lab.
Korea Advanced Institute of Science and Technology
dhcho@ee.kaist.ac.kr

***Abstract*— In this paper, we introduce a new proxy that effectively prevents unnecessary retransmissions from flowing over a wireless link on a path with sudden delay. The proposed STD algorithm detects spurious timeout based on the comparison of the measured inter-arrival timeout of ACK during spurious timeout period with the average inter-arrival time of ACK. It responses to spurious timeout by filtering duplicate ACKs that can cause spurious fast retransmission. Simulation results show that proposed Spurious Timeout Detection (STD) algorithm performs better than, or as well as, other end-to-end mechanisms in the ranges of data rate which current wireless networks can provide.**

I. Introduction

Sudden delay is an unanticipated delay long enough to cause timeout of a TCP sender. If a sudden delay occurs, the retransmission timer of the TCP sender expires and delayed acknowledgement packets(ACK) trigger unnecessary Go-back-N retransmissions of data segments(DATA). The timeout would not have occurred if the TCP sender had waited longer. This is called spurious timeout. The response of TCP-Reno to sudden delay is shown is Fig.1.

It is no longer an uncommon event in wireless networks[1] [2]. Blocking due to high priority traffic, handover or cell reselection and high persistency of link layer ARQ are known as the main reasons of spurious timeout. In recent years, Eifel[3] and F-RTO[4] algorithms were proposed as solutions. These two schemes detect spurious timeout and respond by reversing the congestion control state. But they need server side modification because they are algorithms that enhance the TCP itself. Due to these facts, it is likely to be a long time before they are recommended for implemention in commercial TCP. Sudden delay is a problem of the current and the newly emerging wireless networks. We believe that, at least a certain amount of effort should be made to solve the problem locally. So, as a local enhancement, we introduce a new performance enhancing proxy(PEP) to try to mitigate the performance degradation of TCP experiencing spurious timeout, which operates at the boundary of the wireless networks and the Internet. This paper provides a detailed algorithm for PEP and evaluates its performance.

II. Related Works

Eifel algorithm is composed of Detection and Responding Algorithms. It solves the retransmission ambiguity problem by time stamp option. If a TCP uses time stamp option,

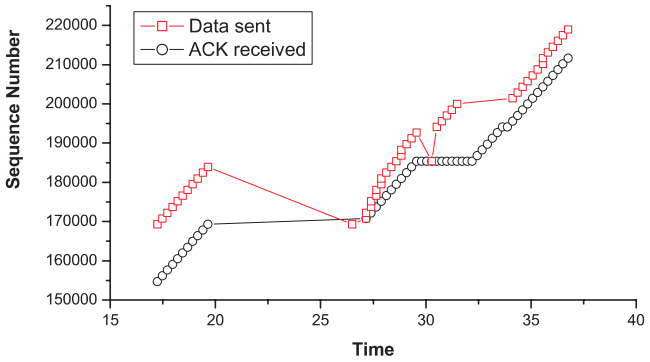

Fig. 1. Response of TCP-Reno to Spurious Timeout

received ACK has the timestamp value of data segment which triggered the ACK. Eifel algorithm can successfully discriminate spurious timeout and normal timeout. But, 12 bytes additional information is likely to be heavy overhead especially in narrow wireless link. In addition, TCP Reno with Eifel experiences performance degradation on the path with sudden delay accompanied by multiple packet losses within one window. In that case, resuming the transmission of unsent data can cause multiple timeout. When spurious timeout is detected, congestion control state is reversed. How to reverse the congestion control state is the problem of responding to spurious timeout. Responding mechanism still remains as research issues[5].

F-RTO is server side modification algorithm. It does not need any additional option field such as time stamp field. When retransmission timer expires, TCP with F-RTO sends two unsent data segments for the first acceptable ACK. It postpones solving the retransmission ambiguity problem[3] until the second ACK arrives. Because two new data were sent, if arrived ACK does not acknowledge all outstanding data but advances window, it means strong indication of spurious timeout. F-RTO algorithm achieves performance increase gracefully. It is great advantage of F-RTO that it does not need any additional cost. But, it has some drawbacks. If packet reordering or packet duplication follows the spurious timeout, it shows a similar action with regular TCP. It may retransmit a window of data. In addition, if timeout occurs during fast recovery, F-RTO causes unnecessary retransmissions like regular TCP.

0-7803-7975-6/03/$17.00 (C) 2003

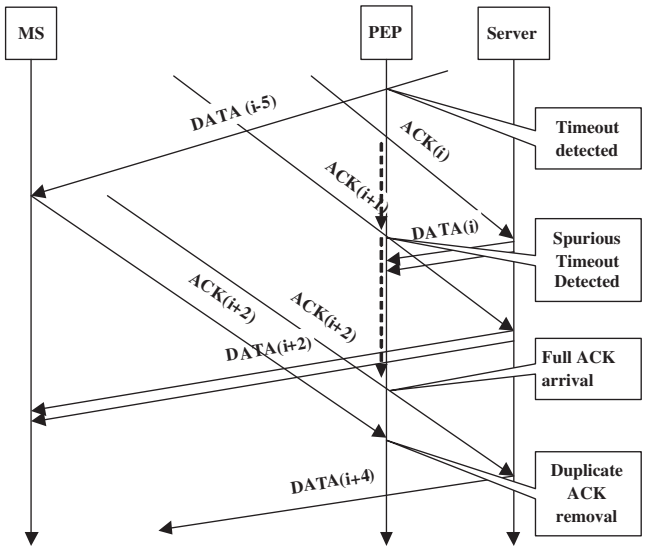

Fig. 2. Detection of spurious timeout and filtering of unnecessary retransmissions

III. STD Algorithm

Proposed algorithm is called STD algorithm. It works in intermediate node PEP between a wireless network and the Internet. We chose PEP approach because it is rapidly adoptable and can be deployed without consent of pre-existing Internet infrastructures or organizations. In this respect, PEP is very useful device that can solve the existing problem until appropriate end-to-end mechanisms are available.

PEP maintains per flow state and tracks all DATAs and ACKs that traverse PEP. Useful Information, such as data sequence numbers and acknowledge sequence numbers are stored in PEP for later use. It first detects timeout at the TCP sender and then investigates further whether the detected timeout is spurious or not. If timeout is identified as spuriousness, it responds by filtering unnecessarily retransmitted segments and removing duplicate ACKs that can cause spurious fast retransmission at the TCP sender. Following subsections explain the STD algorithm in detail.

A. Timeout Detection

PEP maintains the state variable *max_seq_end*[1]. By comparing the sequence numbers of the received DATA, PEP can categorize a received segment as new transmission or retransmission. PEP detects timeout by unanticipated retransmission. There are two kinds of retransmission: one due to fast retransmission and the other due to timeout. A data segment retransmitted by fast retransmission can be anticipated easily because it follows 3 duplicate acknowledgements. So, by this method, PEP can easily discover whether the retransmission is triggered from fast retransmission or timeout.

B. Spurious Timeout Detection

Once timeout is detected, the arrival of first delayed ACK initiates the process of investigating whether the timeout is spurious or not. Fig.2 shows the process of detecting spurious timeout. Detecting spurious timeout is equivalent to solving the problem of retransmission ambiguity[3]. When ACK(i) is received by PEP, it cannot know whether the ACK is in response to the original transmission or in response to a retransmission. So, PEP sets a timer, which expires in time T. The timer's operation is represented by a dashed arrow. It waits for next new ACK that advances a window. If the new ACK arrives before timer expires, it indicates that the timeout was spurious. Otherwise, the new ACK is treated as a response to a retransmission and therefore the STD algorithm stops investigating.

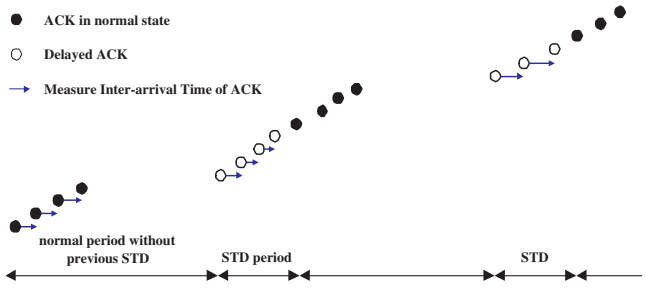

Fig. 3. Measuring inter-arrival time of ACK

When spurious timeout is detected, PEP copies the value of *max_seq_end* to variable *sp_tout_max_seq_end*. Upon the arrival of DATAs from the TCP sender, DATAs with sequence number larger than *acked_seq* but less than *sp_tout_max_seq_end* are stored temporarily in PEP. In Fig.2, DATA(i) and DATA(i+1) are buffered. These DATAs are used for local retransmission upon the arrival of delayed duplicated ACK or removed later if received ACK acknowledges them. However DATAs with sequence number lager than *sp_tout_max_seq_end* are forwarded to the wireless network as soon as possible because these segments are new, never transmitted before. DATA(i+2) and DATA(i+3) are forwarded to MS directly as shown in Fig.2.

If new ACK with sequence number larger than *acked_seq*[2] arrives, *acked_seq* is updated and all buffered DATAs with sequence number less than *acked_seq* are removed by PEP. ACK with sequence number larger than *sp_tout_max_seq_end* is called a full ACK because it acknowledges all outstanding DATAs at the moment that spurious timeout occurred. When full ACK is received, PEP performs a duplicate ACK removal as a next step. This will be explained in next subsection. If delayed ACKs are duplicate ones, it indicates that sudden delay is accompanied by one or more segment loss. PEP locally retransmits buffered DATA which is requested by duplicate ACKs if the number of received duplicate ACK reaches *dup_ack_thresh*[3].

C. Setting Timer Value T

The correctness of STD algorithm depends on the timer value T. Basically, T is determined by the average inter-arrival

[1]It contains the maximum DATA sequence number which traversed PEP.

[2]It contains the maximum ACK sequence number that traversed PEP

[3]We can not ignore the possibility of packet reordering. Default value is 3.

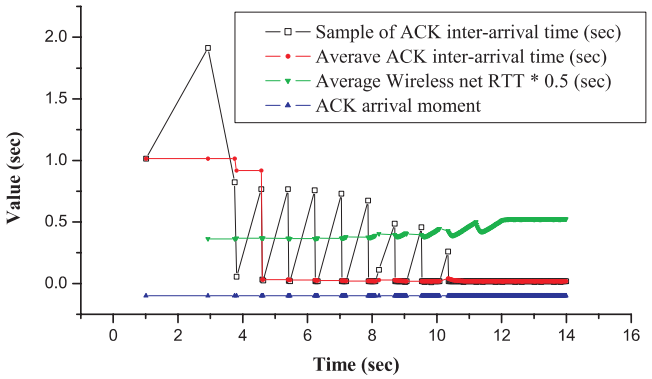

Fig. 4. Calculating ACK inter-arrival time in normal period

time of ACK multiplied by gain G[4]. Whenever PEP receives an ACK, it samples the inter-arrival time and updates the average value by N and S filters described below.

We define 2 different periods in which samples are taken. Both periods are shown in Fig.3. First period is "normal period without previous STD" before. During this period, we estimate the appropriate value of T from every sample of ACK inter-arrival time. Each sample is passed through an N-filter which is described in (1).

$$x_{av\_new} = \begin{cases} x & \text{if } x < \frac{y(n)}{S} \\ \alpha_N x_{av\_old} + \beta_N x & \text{if } \frac{x_{av\_old}}{S} \le x < X_{th} \\ x_{av\_old} & \text{if } X_{th} \le x \end{cases} \quad (1)$$

Here, Sampled ACK inter-arrival time and its average value are denoted as x and x_{av} respectively. N-filter works as follower, smoother and limiter for different ranges of sample x. Follower, $x_{av\_new} = x$, works for fast adaptation to real ACK inter-arrival value, while the limiter, $x_{av\_new} = x_{av\_old}$, excludes unwanted large sample values that exist in the initial phase of slow start period[5]. Smoother, $x_{av\_new} = \alpha_N x_{av\_old} + \beta_N x$, works as linear filter for valid sample values. X_{th} is the threshold of x and is assigned as $0.5wRTT$, where $wRTT$ is the average round trip time from PEP to a Mobile Station. Fig.4 shows the rapid adaptation of average ACK inter-arrival time to real ACK inter-arrival time. Samples of peak values are effectively excluded in updating average ACK inter-arrival time.

The second period is the "STD" period when the STD algorithm works. During this period, sampled values are updated through an S-filter in (2).

$$x_{av\_new} = \begin{cases} \alpha_S x_{av\_old} + \beta_S x & \text{if } x < x_{av\_old} \\ x & \text{if } x_{av\_old} \le x \end{cases} \quad (2)$$

In this filter, follower, $x_{av\_new} = x$, works for rapid tracking of the valid ACK inter-arrival sample. Rapid adaptation to the increase of ACK inter-arrival time results in PEP having a timer value T large enough to allow the STD algorithm to complete its operation. This operation is required when sample

[4]G is system dependent paramter. In performance evaluation, G is set to 2.

[5]In the slow start phase of TCP, the number of consecutively received ACKs increases as 1, 2, 4, 8 $\cdots$ if ACK delay is not considered. The inter arrival time between these groups of ACKs is relatively large and is treated as noisy sample and excluded. These noisy samples are shown as peak values in Fig.4.

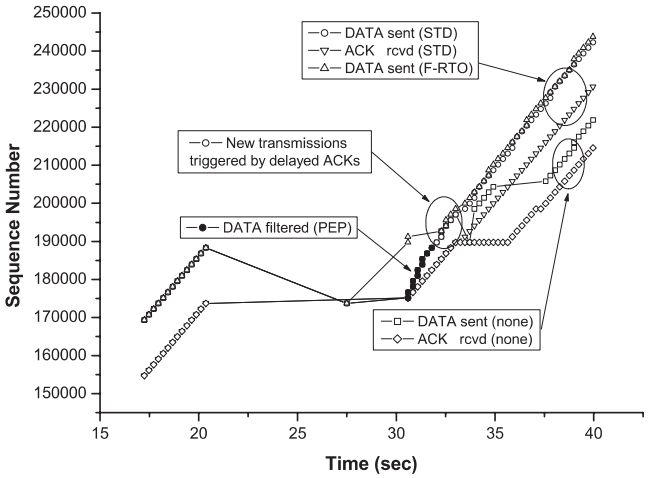

Fig. 5. Operation of STD

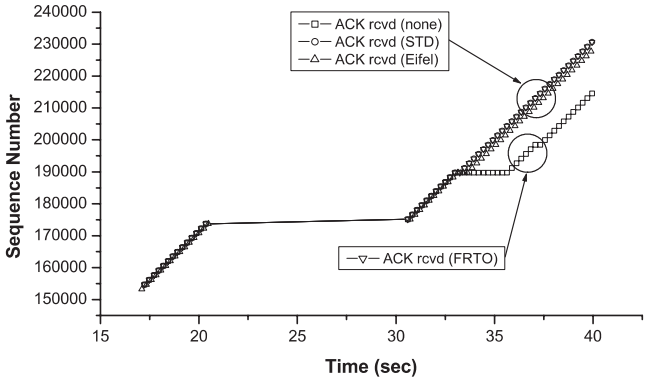

Fig. 6. Sequence number of ACK received by various TCP receiver

values between normal period and STD periods correlate poorly. In particular, when bandwidth oscillation[2] exists, samples from normal period correlate poorly with samples from the STD period. Due to this low correlation, PEP does not take samples in the normal period, which follows the STD period.

S is selected as 20. And both α_N and α_S are selected as 0.9 and accordingly $\beta_N = 1 - \alpha_N$ and $\beta_S = 1 - \alpha_S$ are 0.1 in the performance evaluation[6].

D. Duplicate ACK removal

Full ACK is the last delayed ACK. So, when PEP receives a full ACK, it moves into the final stage responding to spurious timeout. In this stage, PEP removes incoming duplicate ACKs that are triggered from retransmission timeout. If the delay is large, multiple timeouts can occur and cause more than two duplicate ACKs. More than three duplicate ACKs cause spurious fast retransmission at TCP sender. So, PEP filters this unnecessary duplicate ACKs. It can prevent the halving of congestion window and is helpful for preserving the current sending rate of TCP sender.

[6]Further research will be required if an appropriate coefficient value is to be chosen. We think the process of choosing the coefficient value of the RTT smoother in the TCP round trip timer may yield valuable insight and references.

IV. Analysis of STD

Fig.5 shows the increasing sequence numbers sent from various TCPs. Between 20sec and 30sec, long sudden delay occurs and delayed ACKs begin to arrive about 30sec. They trigger both unnecessary retransmissions and new transmissions from both regular TCP[7] and TCP with STD. Unnecessary retransmissions are noted as filled dot-dashed line in Fig.5. These segments are filtered by PEP. New transmissions are forwarded to client by PEP.

The performance improvement of the STD algorithm is determined by the size of congestion window recovered after spurious timeout. If the sending rate of the TCP sender just after receiving a full ACK is larger than the data rate of bottle necked link, TCP with STD can show equivalent performance improvement with other end-to-end algorithms. As long as the sending rate is higher than the data rate of bottle necked link, the size of congestion window does not matter. If we represent the size of congestion window after spurious timeout as $cwnd_{STD}$, the sending rate is calculated as $cwnd_{STD}/RTT$. We can write the above condition mathematically as (3).

$$\frac{cwnd_{STD}}{RTT} > bottle\_neck\_link\_data\_rate \quad (3)$$

If this condition is met in a given network, the TCP with the STD algorithm shows the similar performance improvement compared with other end-to-end schemes. This condition is easily met when bottle neck bandwidth is small. But the condition is not met over a certain threshold rate. The increase of TCP receiver's window size can cause the increase of flight size, which can result in the increase of $cwnd_{STD}$. So, the increase of the receiver window size could have condition (3) be met at even high data rate. Fig.5 shows the operation of STD when the condition (3) is met. Performance improvement is similar to that of F-RTO because the sending rate is recovered soon enough not to make pause between old transmissions before sudden delay and new transmissions after sudden delay. The sequence number of regular TCP increases with a discontinuity for the reception of duplicate ACKs. ACK sequence increases of four different schemes were compared in Fig. 6. Only regular TCP has a difficulty in increasing its receive sequence number. The duplicate ACKs from 31sec to 33sec are triggered from unnecessary retransmissions. This retransmissions trigger spurious fast retransmission and cause TCP's sending rate to be halved. In Eifel and F-RTO (See Fig.5 and Fig.6), both congestion window and slow start threshold were reduced to the half of its congestion window value[8].

V. Performance Evaluation

To evaluate STD algorithm's performance, we simulated a cellular network in which a mobile user downloads a 10Mbyte file from an Internet server. The data rate of last hop wireless link is noted as BW in Fig.7 and we considered it as bottleneck link. The proposed algorithm works in a proxy located at the border of wireless network. TCP Reno with Maximum Segment Size of 1460byte was used for both TCP sender and receiver. Sudden delay is generated based on [1]'s model. Once sudden delay occurs, Base Station stops transmitting data segment and waits for a certain time until next transmission starts. The distribution of delay duration is uniform distribution with minimum and maximum value of 3 sec and 15 sec, respectively. Inter delay time follows uniform distribution with minimum and maximum value of 20 sec and 40 sec. The proposed STD algorithm was tested in the environment with sudden delay without congestion loss. Throughput and goodput were evaluated. Throughput is defined as "amount of data transmitted divided by time required to complete transfer." Goodput is defined as "minimum amount of data required for completing data transfer divided by actual amount of data transmitted over a wireless link."

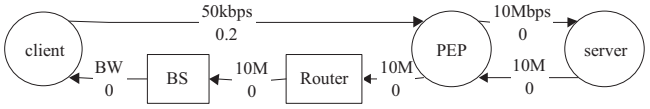

Fig. 7. Simulation network

Fig.8 shows the throughput, throughput increase ratio and goodput when receiver window size is 8k. As we have anticipated earlier, over a certain threshold rate (data rate of bottle necked link), condition (3) is not satisfied anymore. It's because the size of $cwnd_{STD}$ is determined by the flight size before sudden delay which is limited by receiver buffer size. When receiver window size is 8k, 70kbps may be the largest data rate that STD can give throughput similar to end-to-end schemes such as Eifel and F-RTO. As the last hop data rate increases, the throughput gain of STD slowly decreases and eventually will converge to 0%. But goodput gain of STD is always as high as that of F-RTO. It means that proposed algorithm can guarantee the wireless link efficiency irrespective of last hop data rate. This phenomenon comes from the nature of the STD algorithm that it always filters unnecessary retransmissions. Fig.9 shows the results when receiver buffer is increased to 64k. The trend is very similar to previous case. But maximum last hop data rate that can be supported is increased to as much as 200kbps. The threshold rates of 70kbps and 200kbps are very reasonable value because current and legacy 2G or 3G cellular technology (which still experience sudden delay problem) can provide from 9.6kbps to 50 or 60kbps per user. Even though maximum data rate of IS-2000 and WCDMA systems goes to 144kbps or 384kbps respectively, the actual bandwidth that is available to one user is relatively small. We think that about 50kbps per user is maximally achievable throughput in IS-2000 system in the view of transport layer.

The line of goodput of regular TCP is somewhat peculiar. It looks like the cross section of a valley. Two graphs have their lowest points. It is because goodput of regular TCP is affected from two opposite factors. The first factor is RTT. As last hop data rate decreases, transmission time of data segment is increased and it is reflected as the increase of RTO

[7]Regular TCP is noted as "none" in Fig.5.

[8]The responding algorithm of spurious timeout still remains as a research issue.

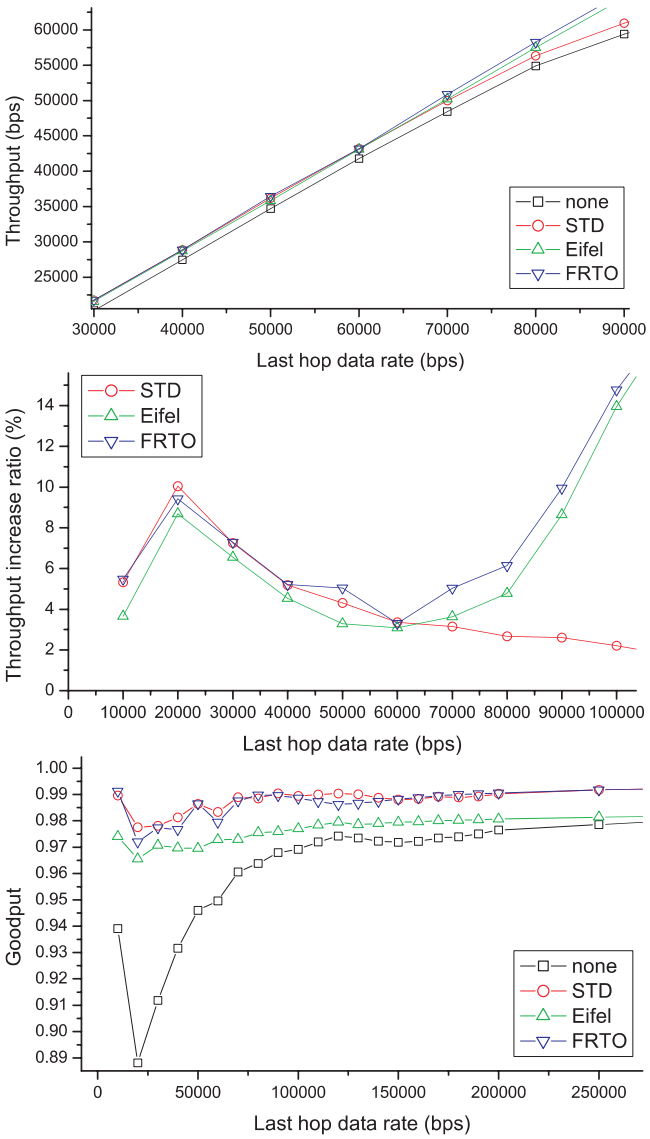

Fig. 8. Throughput, throughput increase ratio and goodput of various schemes on the path with sudden delay, when receiver window size is 8k

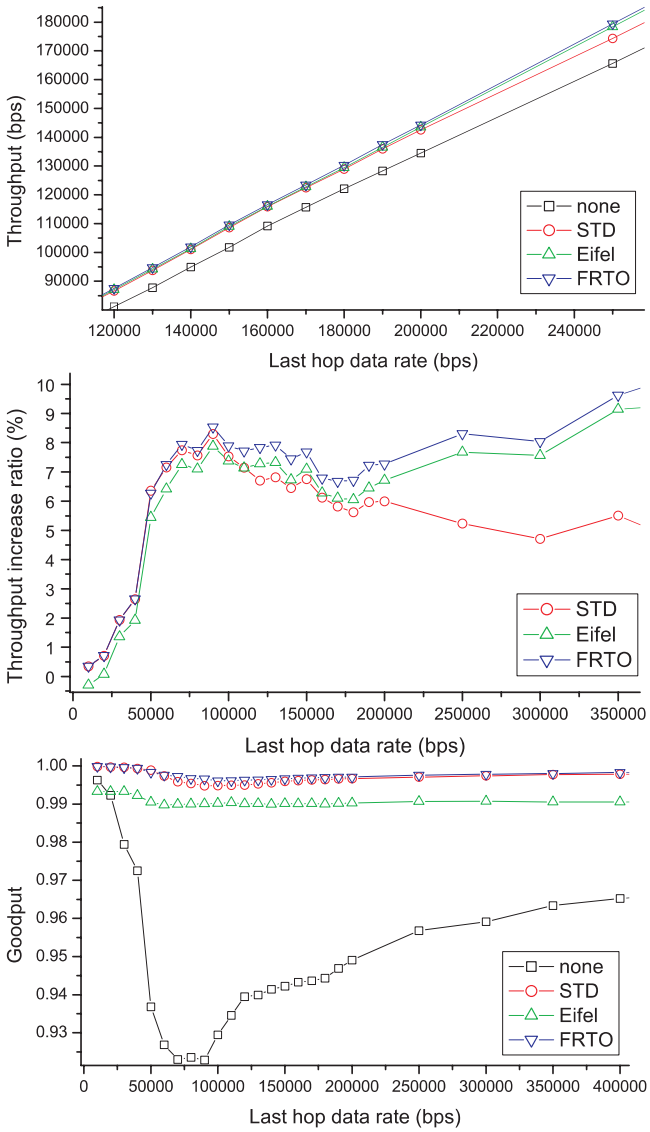

Fig. 9. Throughput, throughput increase ratio and goodput of various schemes on the path with sudden delay, when receiver window size is 64k

at the TCP sender. The large RTO makes TCP endure the sudden delay without fast expiration of timer and it leads to the decrease of possibility of spurious timeout. Hence, the decrease of last hop data rate increases goodput. The second factor is the time required for data transfer. The amount of user's data is always limited. As last hop data rate increases, time required to complete data transfer decreases. It means the number of experiencing sudden delay also decreases. And it consequently leads to the decrease of the amount of unnecessary retransmissions. The second factor implies that the increase of last hop data rate increases goodput. These two opposite factors are additively combined in the simulation and that make the two lowest points.

VI. Conclusions

We showed that proposed STD algorithm can compensate TCP performance experiencing spurious timeout by detecting and responding spurious timeout at intermediate PEP. It improved TCP throughput and goodput in the ranges of data transmission rate that current wireless communication systems could support. The proposed proxy style solution will be effective until end-to-end solution is available. As a further work, we are focusing on finding optimum filter coefficients and analyzing performance of given algorithm on the path with various lossy environments.

References

[1] A. Gurtov, R. Ludwig, "*Evaluating the Eifel algorithm for TCP in a GPRS network*," In Proceedings of European Wireless, February 2002.

[2] F. Khafizov, M. Yavuz, "*Running TCP over IS-2000, IEEE conference on communications*," April 2002.

[3] Reiner Ludwig, Randy H. Katz, "*The eifel algorithm: making TCP robust against spurious retransmissions*," ACM Computer Communication Review, vol. 30, no. 1, January 2000.

[4] Pasi Sarolahti, Markku Kojo, Kimmo Raatikainen, "*F-RTO: a new recovery algorithm for TCP retransmission timeouts*", University of Helsinki, Department of Computer Science Techical Report C-2002-07, February 2002.

[5] A. Gurtov, R. Ludwig, "Responding to Spurious Timeouts in TCP," IEEE INFOCOM, vol. 3, pp. 2312-2322, March 2003.