# TCP-Adaptive Reno for Improving Efficiency-Friendliness Tradeoffs of TCP Congestion Control Algorithm

Hideyuki Shimonishi, Takayuki Hama, and Tutomu Murase

***Abstract*—it has been recognized that TCP throughput deteriorates in networks with large bandwidth-delay-product and non-negligible packet losses. A number of protocols, such as High Speed TCP, Scalable TCP and TCP-Westwood have been proposed to address this problem. However, their lack of friendliness to existing protocols has hampered their wide deployment in public networks. In this paper, we propose TCP-AReno (Adaptive Reno) to ensure friendliness to TCP-Reno, as well as efficiency in high-speed networks. A key feature of TCP-AReno is that it dynamically adjusts the TCP response function based on congestion estimation via RTT measurement. It increases congestion window faster and decreases the window less significantly than TCP-Reno when it recognizes no congestion. As the congestion level increases, it tunes the response function so that it behaves like TCP-Reno. Simulation and experimental results show that TCP-AReno maintains friendliness to TCP-Reno in networks with varying configurations, while it achieves much higher throughput than TCP-Reno.**

## I. Introduction

Since TCP has been designed for networks where a packet loss is recognized as a congestion signal, it is well known that TCP suffers from degradation in fast and long-distance networks with non-negligible random losses. To improve performance of TCP in such networks, a number of new TCP variants, including High Speed TCP (HS-TCP) [1], Scalable TCP [2], and TCP-Westwood (TCP-W) [3,4,6], have been proposed. These protocols follow an AIMD (Additive Increase and Multiple Decrease) behavior of TCP-Reno but have more aggressive response functions. For example, when a High Speed TCP flow finds a packet loss, it decreases its congestion window less significantly than TCP-Reno, and otherwise it increases the congestion window faster than TCP-Reno, and thus, it could severely damage coexisting TCP-Reno flows. Their potential unfriendliness to TCP-Reno could be one of the reasons that have hampered their wide deployment in public networks.

In this paper, we present TCP-AReno (Adaptive Reno) [5] and some simulation and experimental performance evaluations. TCP-AReno adaptively tunes TCP-Reno's response function in such a way that it improves efficiency in high-speed networks as well as friendliness to TCP-Reno. TCP-AReno is based on TCP-Westwood-BBE (Buffer and Bandwidth Estimation) [4], which is proposed to enhance friendliness of TCP-Westwood even in networks with varying buffer capacities, flows with varying RTTs, with/without AQMs. It estimates congestion level via RTT to determine whether a packet loss is due to congestion or not. Basically, any multi-bit congestion indicators can be used for this purpose if they detect congestion only when coexisting TCP-Reno flows detect congestion, i.e. only when a packet loss is likely to happen. We employed RTT here because it is proven to be a good congestion estimator [7] and does not require any network supports. If a loss is not recognized as a congestion event, the congestion window is reduced according to TCPW's eligible bandwidth estimation. Otherwise, the loss is associated to congestion and it adjusts the window reduction so that the reduction matches TCP-Reno's behavior, i.e., halving the congestion window. Our Internet measurements have shown that this loss discrimination helps to improve throughputs without losing friendliness to TCP-Reno [9].

TCP-AReno introduces a fast window expansion mechanism to quickly increase congestion window whenever it finds network underutilization. It dynamically adjusts the TCP response function based on the congestion measurement introduced by TCPW-BBE, whereas HS-TCP adjusts the function based on congestion window size. TCP-AReno increases the congestion window much faster than TCP-Reno when it detects no congestion, i.e. when RTT is close to its minimum value, and thus achieves higher efficiency than TCP-Reno. When a TCP-AReno flow competes with TCP-Reno flows, it detects increased RTT and thus increased congestion level. In this case, it recognizes that packet losses are likely to happen and adopts a response function that matches TCP-Reno's behavior, i.e. increasing the congestion window by 1MSS/RTT.

In this paper, we describe TCP-AReno in section 2. Then, simulation and experimental evaluations are given in Section 3 and 4, respectively, followed by conclusion in Section 5.

## II. TCP-Adaptive Reno (TCP-AReno)

### A. Congestion Measurement

A key feature of TCP-AReno is that it dynamically adjusts

The authors are with System Platforms Laboratory, NEC Corporation, 1753 Shimonumabe, Nakahara-ku, Kawasaki-shi, 211-8555 (corresponding author to provide e-mail: h-shimonishi@cd.jp.nec.com).

the TCP response function based on congestion level estimation introduced by TCPW-BBE.

TCP-AReno determines the range of RTT between $RTT_{min}$ and $RTT_{cong}$, the value that RTT is expected to take when a packet is lost due to congestion. Given that packet losses are mainly due to congestion, $RTT_{cong}$ is estimated using RTT

$$RTT_{cong}^{j} = (1-a)RTT_{cong}^{j-1} + aRTT^{j}$$

values measured right before the losses. Namely, $RTT_{cong}$ is updated upon each packet loss event as follows;
where index $j$ expresses $j$-th packet loss event and $a$ is an exponential smoothing factor. When current RTT is close to $RTT_{min}$, a TCP-AReno sender recognizes that the network is underutilized, and when the RTT is close to $RTT_{cong}$, it

$$c = \min\left(\frac{RTT - RTT_{\min}}{RTT_{cong} - RTT_{\min}},\ 1\right)$$

recognizes the network is being congested. Thus, congestion level $c$ $(0 \le c \le 1)$ is defined as follows:

### B. Congestion Window Increase

In TCP-AReno, congestion window $W$ is managed in two parts; a base part, $W_{base}$, and a fast probing part, $W_{probe}$, as shown Fig. 1. The base part maintains a congestion window size equivalent to TCP-Reno and the probing part is introduced to quickly fill the bottleneck link. The base part is always increased like TCP-Reno, i.e. 1MSS/RTT, while the increase of the probing part, $W_{inc}$, is dynamically adjusted as illustrated in Fig. 2.

$$W_{base} = W_{base} + 1MSS / W,$$
$$W_{probe} = max( W_{probe} + W_{inc} / W, 0 )$$

(1) When the network is recognized as underutilized, TCP-AReno sets $W_{inc}$ close to its maximum value $W_{inc}^{max}$. To scale up to high-speed networks, $W_{inc}^{max}$ should be adequately sized according to the bottleneck link capacity, pipe size, congestion window size, and so on. While we are now studying how to size $W_{inc}^{max}$ through our experiments, we found that linearly scaling $W_{inc}^{max}$ according to the estimated bottleneck link capacity $B$ better works in most of the cases; thus,

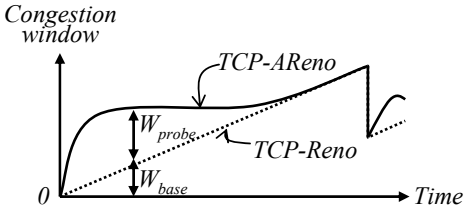


Fig. 1: Congestion window increase of TCP-AReno during congestion avoidance

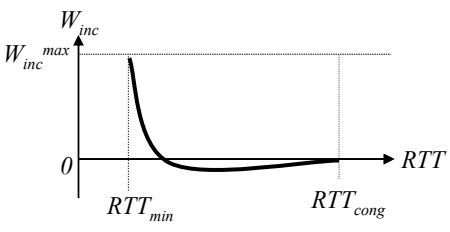


Fig. 2: Congestion window increase per RTT ($W_{probe}$ part)

$$W_{inc}^{max} = B / M * MSS.$$

where $M$ is a scaling factor we set at $10Mbps$ based on our Internet measurements so far. $B$ is estimated through time sequence of ACKed sequence numbers like TCPW.

(2) As the congestion level $c$ increases, TCP-AReno quickly decreases $W_{inc}$, even less than 0, so that congestion window of coexisting TCP-Reno flow catches up to that of TCP-AReno flow.

(3) When $c$ approaches 1 and a packet loss is likely to happen, $W_{inc}$ is gradually increased again so that it approaches to 0. In this situation, the base part takes a main role and TCP-AReno behaves like TCP-Reno.

To realize above described U-curve for $W_{inc}$, we calculate it as a combination of an exponential function for (2) and a linier function for (3). Therefore, $W_{inc}$ is given by the following function of $c$;

$$W_{inc}(c) = W_{inc}^{max}/e^{\alpha c} + \beta c + \gamma$$

where $\alpha$ determines the range when the network is recognized as underutilized. While $\alpha$ should be small enough to allow errors in RTT measurement, experiments using our Linux implementation show that $\alpha=10$ is adequate. For better friendliness to TCP-Reno, $W_{probe}$ should have converged to 0 when a packet loss is like to happen. Thus, we obtain parameters $\beta$ and $\gamma$ as follows:

$$\beta = 2W_{inc}^{max}(1/\alpha - (1/\alpha+1)/e^{\alpha})$$
$$\gamma = 1 - 2W_{inc}^{max}(1/\alpha - (1/\alpha+1/2)/e^{\alpha}).$$

### C. Congestion Window Reduction

TCP-AReno adjusts congestion window reduction based on the congestion measurement. Like TCPW-BBE, the congestion window is halved when the network is congested and a packet loss is likely to happen, while the reduction is mitigated when the network is underutilized, as shown in Fig. 3. Thus, the congestion window is reduced as follows:

$$W_{base} = W*W_{dec} = W / (1+c), \quad W_{probe} = 0.$$

This figure also illustrates the congestion window reduction of TCP-Reno and original TCPW (TCPW+[6]). As it is indicated, TCPW halves the congestion window when RTT is twice $RTT_{min}$ and thus the buffer size at the bottleneck router is equal to the pipe size. That is why friendliness of TCPW deteriorates when buffer size or RTT varies, and friendliness of TCP-BBE and TCP-AReno maintains against these variations.

## III. Simulation Results

### A. Evaluation Setups

In this section, we show some of our simulation results using

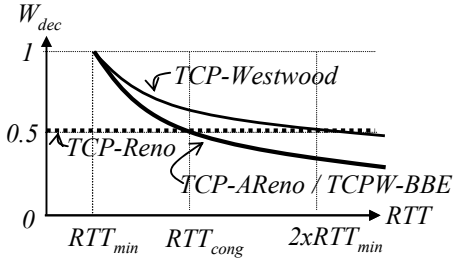


Fig. 3: Congestion window decrease at a packet loss

ns2 simulator [8]. We first consider a simple dumb-bell topology (1), shown in Fig. 4, where a High Speed TCP (HS-TCP) or TCP-AReno flow competes with a TCP-Reno flow. We also consider a more complex topology (2), shown in Fig. 5, where multiple bottleneck links are cascaded. Bandwidth and propagation delay of the links are illustrated in these figures. In Fig. 4, router buffer capacity is set equal to the bandwidth delay product and the buffer employs drop-tail discarding. In Fig. 5, the buffer employs RED (Random Early Detection), and the minimum and maximum thresholds are set at 10% and 30% of the bandwidth delay product for the longest flow.

### *B. Friendliness in high-speed environment*

Figure 6 shows throughputs of two competing TCP flow in topology (1); one uses TCP-Reno and the other uses either HS-TCP, TCP-AReno, or TCP-Reno. Bottleneck link capacity ranges from 10Mbps to 1Gbps and random packet loss rate at the bottleneck link is set at $10^{-6}$. This figure shows that a TCP-Reno flow can only utilize the bottleneck link bandwidth up to 150Mbps because of the random packet loss. When a HS-TCP flow and a TCP-Reno flow competes at the bottleneck link, although the HS-TCP flow can almost fully utilizes the link capacity up to 1Gbps, at the same time, it severely deteriorates TCP-Reno throughput. For example, when the link capacity is 200Mbps, the TCP-Reno flow can use only 10% of its eligible bandwidth. On the other hand, TCP-AReno can fairly share the bottleneck link bandwidth with TCP-Reno. These two flows equally share the bandwidth when the link capacity is less than 300Mbps. With larger link capacity, a TCP-Reno flow can utilize only 150Mbps of the bandwidth due to random packet losses but the TCP-AReno flow can use up rest of the bandwidth.

### *C. Friendliness in lossy environment*

Figure 7 shows throughputs of two competing TCP flows with varying random packet loss rates on a 100Mbps bottleneck link. If the loss rate is smaller, two TCP-Reno flow can use up the link capacity and fairly share the bandwidth, but they underutilize the bottleneck link as the loss rate increases.

When one of the TCP-Reno flow is replaced by a HS-TCP flow, which improves bandwidth utilization when the loss rate is less than 0.01%, the HS-TCP flow severely deteriorates throughput of the TCP-Reno flow. When a TCP-Reno flow and a TCP-AReno flow coexist, TCP-AReno flow can fairly share the bottleneck link. This figure also indicates that TCP-AReno is more robust to random packet losses than HS-TCP, for example, TCP-AReno can obtain 7 times larger throughput

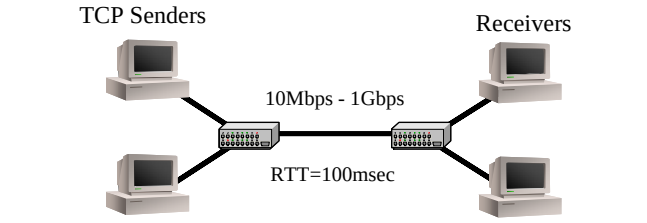


Fig. 4: Network topology (1)

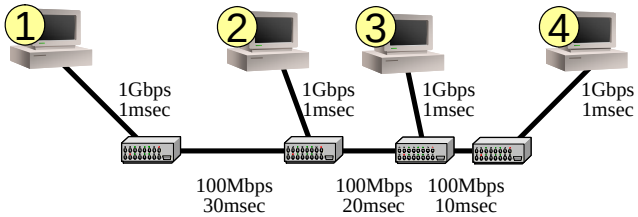


Fig. 5: Network topology (2)

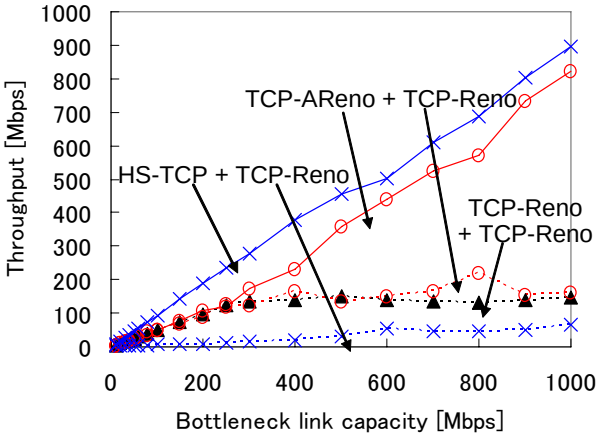


Fig. 6: Throughput of coexisting 2 flows (Large capacity case; loss rate = $10^{-6}$)

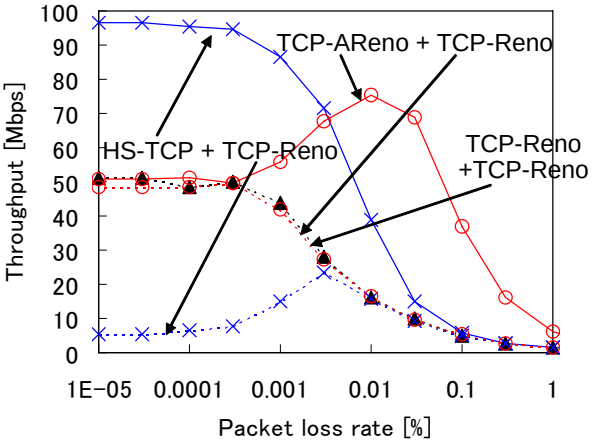


Fig. 7: Throughput of coexisting 2 flows (Large packet loss case; capacity = 100Mbps)

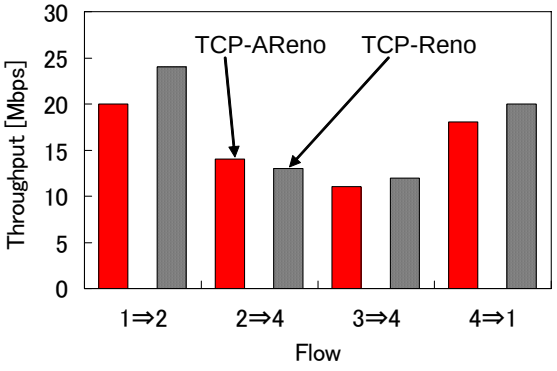


Fig. 8: Throughput of coexisting 8 flows (Topology 2)

without affecting TCP-Reno throughput when the loss rate is 0.1%.

### D. *Fairness among individual flows*

Throughputs of TCP-AReno flows with different RTTs and different hop counts are evaluated using network topology (2). Reverse flows traversing the routers in the reverse direction are included to take into account the important effects of such reverse traffic.

In Fig. 8, individual throughputs of coexisting 4 TCP-Reno flows and 4 TCP-AReno flows are shown. On the forward path, the bottleneck link bandwidth is almost fully utilized by 6 forward flows, and a TCP-AReno and a TCP-Reno flow on the same path obtain almost the same bandwidth. On the reverse path, however, although TCP-AReno flow obtains almost the same bandwidth as TCP-Reno flow, the link utilization is only around 40%. As is the same with TCP-Vegas or FAST, since they use RTT as a congestion indicator, TCP-AReno also can not efficiently utilize the residual capacity when its reverse path is congested. However, it should be emphasized that TCP-AReno is still at least as efficient as TCP-Reno, while TCP-Vegas and FAST could be less eficient than TCP-Reno in this situation.

## IV. Experimental Results

### A. *Evaluation Setups*

Some of the experimental results are shown in this section. In our laboratory and Internet experiments, we implemented TCP-AReno on a "layer-2" TCP proxy that relays TCP flows from TCP-Reno to TCP-AReno, as shown in Fig. 9. The proxy is placed as a gateway for long-distance link, so that long-distance TCP throughputs are improved without modifying TCP hosts that have no idea whether their flows are going that far, or just within a local network. For a traffic source, we use Iperf [10] to generate continuous TCP data flow.

### B. *Laboratory measurements*

Figure 10 and 11 show congestion window behaviors of a TCP-AReno flow and a TCP-Reno flow in the experimental laboratory network with $10^{-5}$ random packet losses and 300Mbps bottleneck link capacity at the network emulator. In Fig. 10, congestion window of a lone TCP-AReno flow and a lone TCP-Reno flow are compared, while in Fig. 11, their congestion windows are plotted when they coexists.

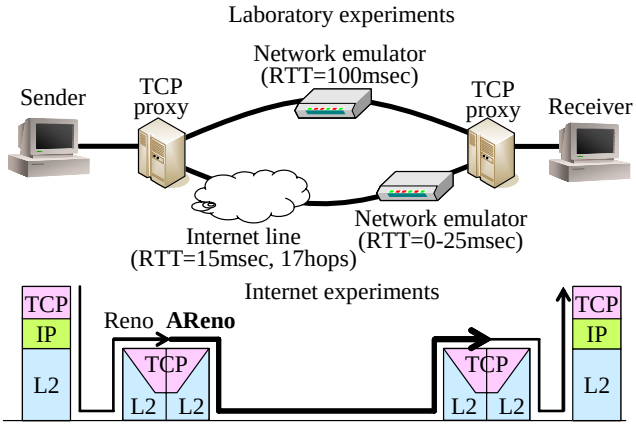


Fig. 9: Network topology and "Layer-2" TCP proxy

As shown in Fig. 10, packet losses occur even when congestion window is far less than the bandwidth delay product, and thus average throughput of a TCP-Reno flow is just 18.3Mbps. On the other hand, average throughput of a TCP-AReno flow is as high as 232.1Mbps, 12.6 times improvement. It is confirmed that the TCP-AReno flow quickly increases its congestion window when the bottleneck link is not fully utilized. And when its congestion window grows large enough, its increase mitigated to avoid unnecessary congestion. The spikes in the congestion window size indicate a window inflation mechanism, which is a unique behavior of BSD-based TCP stack, meaning a packet loss occurring at this point. By observing the spikes in Fig. 10, one can notice that, if a packet loss occurs when the link is underutilized, TCP-AReno recognizes it as a non-congestion event and its congestion window reduction is minimized. Otherwise, the loss is recognized as a congestion event and the congestion window is halved, which is an important behavior for friendliness to TCP-Reno.

When a TCP-Reno flow and a TCP-AReno flow coexist, as

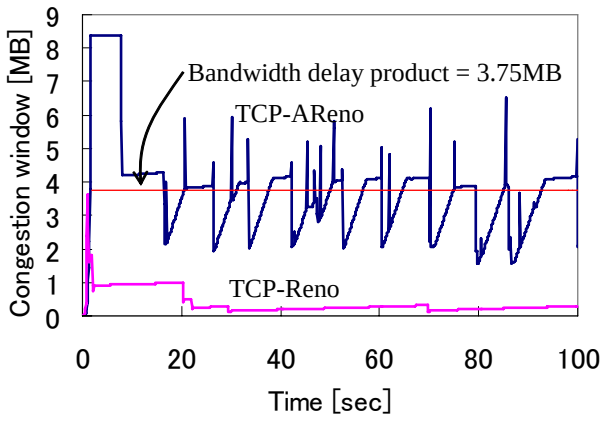


Fig. 10: Congestion window behavior; separate experiment (Laboratory measurement, 100Mbps, $10^{-5}$ packet loss)

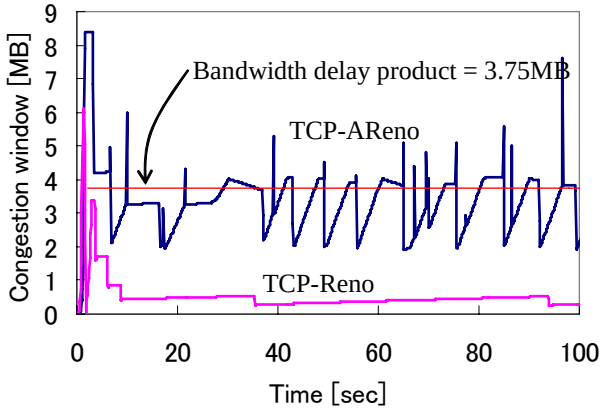


Fig. 11: Congestion window behavior; coexisting flows (Laboratory measurement, 100Mbps, $10^{-5}$ packet loss)

shown in Fig.7, the TCP-AReno flow also achives high throughput againt random packet losses without disturbing coexisting TCP-Reno flow. Average throughputs of the TCP-AReno and TCP-Reno flows are 227.4Mbps and 18.5Mbps, respectively, which are almost the same as they solely exist.

In Fig. 12, throughput of a lone flow is shown for different bottleneck capacities at the network emulator. In this case, RTT is set at 20msec. This figure shows the effectiveness of TCP-AReno in high-speed and lossy environment. Especially with high random packet loss rate of $10^{-4}$, TCP-AReno flow obtains ten times more throughput than TCP-Reno.

### C. *Internet measurements*

We had a series of Internet measurements in an environment shown in Fig. 9, where the bottleneck is a 100Mbps link to FTTH Internet access. No additional packet losses are introduced at the network emulator.

In Fig. 13, relative throughput of a relayed TCP-AReno flow and an end-to-end direct TCP-Reno flow is plotted. When no additional delay is added at the network emulator, both TCP-AReno and TCP-Reno achieve similar throughput. When 25msec round trip delay is added, packet losses at the Internet path can cause larger effects on the throughput. Since TCP-AReno is much robust against ransom packet losses, TCP-AReno flow has 2.4 times larger throughput than TCP-Reno flow.

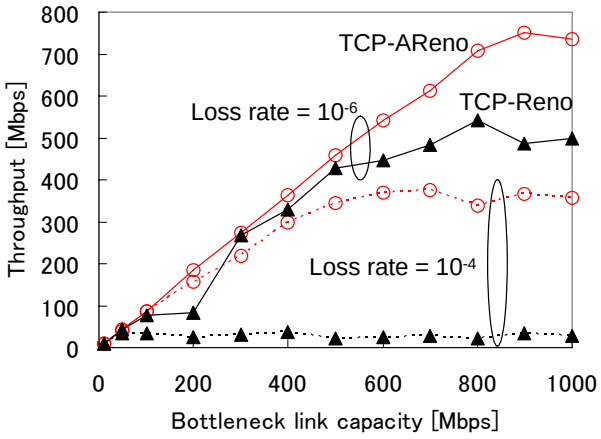


Fig. 12: Throughput of a lone flow (Laboratory measurement, 20msec)

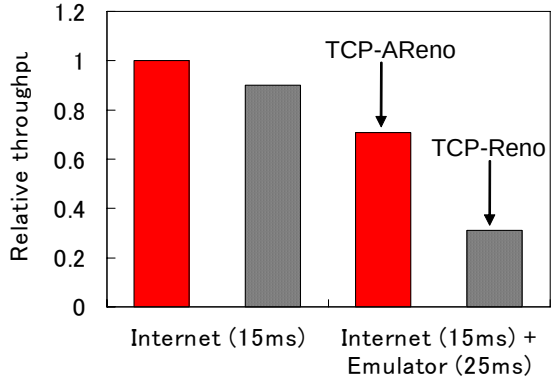


Fig. 13: Throughput of a lone flow (Internet measurement, 15msec/40msec)

## V. Conclusion

In this paper, we presented TCP Adaptive Reno, or TCP-AReno, which provides an efficient and TCP-Reno-friendly congestion control. By incorporating a congestion estimation scheme through RTT measurement, TCP-AReno dynamically adjusts the TCP response function appropriately.

Our simulation and experimental results show that TCP-AReno is friendly to coexisting TCP-Reno flows over a wide range of link capacities and random packet loss rates, while achieving higher efficiency than TCP-Reno or even High-Speed TCP. We also note that the computational cost of TCP-AReno is comparable to that of HS-TCP.

As a future work, we are planning more extensive Internet measurements over long-distance high-speed lines and will tune the response functions. Fairness among flows having variety of RTTs, as well as the effects of reverse traffic, would be one of the important issues.

## Acknowledgment

The authors would like to thank Prof. Gerla and Prof. Sanadidi, and Mr. Marcondes, University of California, Los Angeles, for their collaboration in proposing and evaluating TCPW-BBE, as well as for their comments regarding TCP-AReno. The author also would like to thank Prof. Murata, Prof. Hasegawa, and Mr. Mori for their help in setting up the Internet measurement environment.

## References

[1] S. Floyd, “High Speed TCP for large congestion windows”, RFC 3649, 2003.
[2] T. Kelly, “Scalable TCP: Improving performance in high-speed wide area networks”, In Proc. Of PFDnet03, 2003.
[3] R. Wang, M. Valla, M. Y. Sanadidi, and M. Gerla, "Adaptive Bandwidth Share Estimation in TCP Westwood", In Proc. of Globecom 2002.
[4] H. Shimonishi, M. Y. Sanadidi, and M. Gerla, “Improving Efficiency-Friendliness Tradeoffs of TCP in Wired-Wireless Combined Networks”, In Proc. of ICC2005, 2005.
[5] H. Shimonishi, T. Hama, and T. Murase, “Improving Efficiency-Friendliness Tradeoffs of TCP Congestion Control Algorithm”, In Proc. of Globecom2005, 2005.
[6] R. Ferorelli, L. A. Grieco, S. Mascolo, G. Piscitelli, and P. Camarda, “Live Internet Measurement Using Westwood+ TCP ongestion Control”, In Proc. Of FLOBECOM2002, 2002
[7] C. Jin, D. X. Wei, S. H. Low, “FAST TCP: motivation, architecture, algorithms, performance”, in Proc of IEEE INFOCOM2004, 2004.
[8] The network simulator– ns-2, http://www.isi.edu/nsnam/ns
[9] C. Marcondes, M. Y. Sanadidi, M. Gerla, H. Shimonishi. T. Hama, and T. Murase, “Inline Path Characteristic Estimation to Improve TCP Performance in High Bandwidth-Delay Networks”, In Proc. of PDLFnet2006, 2006
[10] Iperf, http://dast.nlanr.net/Projects/Iperf/