Dynamic Parameter Adjustment for Available-Bandwidth Estimation of TCP in Wired-Wireless Networks

Natsuki ITAYA and Shoji KASAHARA

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan

Abstract

With the extensive spread of the Internet and mobile terminals, the Internet access from a mobile terminal becomes more important than ever before. In the Internet, Transmission Control Protocol (TCP) is used as a standard data transport protocol, however, it is well known that TCP performance degrades in wired-wireless network environments. This is because TCP is designed originally for wired network environment. In this paper, we propose a simple TCP algorithm for wired-wireless network environment focusing on small-sized file documents transmission. In the proposed algorithm, available bandwidth of network is well estimated with time intervals between consecutive ACKs. With the estimates, the congestion control in the proposed method adapts to the state of wireless link and prevents throughput degradation. A simple filter is used for available bandwidth estimation and dynamic adjusting of filter parameter according to the state of wireless link improves the throughput of small-sized file transmission. We compare the proposed algorithm with existing TCP Reno and TCP Westwood by simulation and show that the proposed algorithm achieves better performance than existing two TCP algorithms in wired-wireless network environment.

Keywords: TCP, the Internet, Wired-Wireless Network, Web Document Transmission

1 Introduction

The Internet is indispensable in today's social life and Internet accessibility becomes more important than ever before. In addition, the demand of ubiquitous networking increasing day by day. In the near future, ubiquitous networking is environment will be realized based on currently existing technologies for the Internet, and it is natural that network access from mobile terminals becomes important for the realization of ubiquitous networking. Existing applications such as E-mail, Web browsing, Web directory service and multimedia data distribution, and future new applications will be used in a network, and it will be necessary to transmit and to receive data using a reliable data transport protocol. The protocol used in the existing Internet for reliable data transport is a Transmission Control Protocol (TCP) [32], and it is considered that TCP becomes one of the dominant protocols in the future ubiquitous networking as well. Currently, TCP being used dominantly is TCP Reno [36].

TCP Reno algorithm assumes that packet loss occurs at router buffer mainly due to network congestion, and when packet loss occurs, Reno reduces transmission rate for network congestion avoidance [2, 18, 37]. When the algorithm was proposed at the end of 1980s, only the use in wired

network environment was taken into consideration and network access from mobile terminal was not expected at all. If a mobile terminal such as mobile cellular phone, laptop PC and PDA accesses the Internet, the mobile terminal uses wireless link to access to the network. One of the different characteristics of wireless link from wired one is fading due to the degradation of radio channel. Because of fading, the bit error rate of wireless link changes rapidly in wide range. Another characteristic of wireless link is cellular handoff by the movement of the mobile terminal. When handoff occurs, the mobile terminal changes wireless base station and this results in the re-establishment of TCP connections and packet loss may occur successively. Because the network available bandwidth changes rapidly and bit error rate of wireless link varies widely, existing TCP degrades throughput performance in wired-wireless network. The reason is as follows.

In wireless link, packet loss occurs due to not only network congestion but also wireless fading and cellular handoff. In the case of packet loss due to wireless fading or cellular handoff, TCP does not need perform congestion control. However, TCP cannot recognize which causes packet loss, network congestion or degradation of wireless link due to fading or cellular handoff. TCP sender is in the edge of the Internet and TCP performs congestion control even when the packet loss causes the degradation of wireless link. Note that wireless link is likely to recover from the degradation and this implies the rapid increase of the network available bandwidth. If this rapid recovery of wireless link occurs during congestion avoidance phase, the transmission rate of TCP is not adapted to the network available bandwidth immediately and this also results in the degradation of TCP throughput.

In the near future, the next generation Internet protocol IPv6 becomes standard and every mobile terminal will be assigned IP address. In order to access to the Internet from mobile terminal seamlessly, data transmission performances such as loss and delay in wireless network must be the same as those in wired network. A number of studies have been done to improve TCP throughput performance in wired-wireless network environment [5-11, 13-15, 19, 26, 29, 33-35, 39, 41]. The approaches to improve TCP performance in wireless network can be classified into the following: (1) link-layer protocols [4, 13, 20, 23-25], (2) split-connection protocols [5, 6, 11, 16, 17, 41], (3) snoop protocols [7, 9, 33] and (4) end-to-end protocols [10, 15, 19, 26, 27, 34, 35, 39, 40]. We summarize these studies in the next section.

Although most of those works have succeeded in the enhancement of TCP performance, there still exist several issues such as the deployment and the scalability for the actual Internet. As for the deployment issue, the end-to-end protocol has advantage than other protocols. From this reason, one of the realistic approaches for improving performance is to modify existing TCP. In this paper, we consider the modification of existing TCP for improving throughput performance in wired-wireless network.

In most of related works, throughput performance has been evaluated for the transmission of infinitely large-sized file. However, the transmission of small-sized file such as Web document is more realistic for wired-wireless network environment than the large-sized file transmission. Therefore, our primary concern is how to achieve high throughput performance for the small-sized file transmission.

In this paper, we propose a TCP algorithm for wired-wireless network especially suitable for small-sized file transmission. In the proposed algorithm, a simple filter is used for available bandwidth estimation and dynamic adjustment of filter parameter according to the state of wireless link improves the throughput of small-sized file transmission. We investigate the performance of the proposed algorithm by simulation and show how effective the proposed algorithm is for the small-sized file transmission.

The paper is organized as follows. Section 2 summarizes the previous works related to the performance improvement of TCP for wired-wireless networks. Section 3 represents the design

of our proposed TCP algorithm in detail. The performance evaluation by simulation is presented in Section 4 and Section 5 concludes the paper.

2 Related Work

In this section, we summarize the previous works related to the improvement of TCP throughput for wired-wireless networks. As we stated in Section 1, the approaches to improve TCP performance can be classified into (1) link-layer protocols, (2) split-connection protocols, (3) snoop protocols, and (4) end-to-end protocols.

In the link-layer protocol, a wireless link is considered as a reliable link using link-layer techniques such as automatic repeat request (ARQ) [23] and forward error correction (FEC) [24, 25]. The link-layer protocols for CDMA [20] primarily use ARQ technique. AIRMAIL protocol [4] adopts a combination of ARQ and FEC techniques for loss recovery. These protocols have advantage that those fit naturally into the layered protocol stack of the Internet. The link-layer protocol like ARQ or FEC operates independently of transport layer protocol like TCP. These protocols can improve TCP performance in wired-wireless network better than other protocols. However, these need special support of network infrastructure such as router. It is difficult for link-layer protocol to be deployed widely due to the expensive cost for special support of infrastructure.

The split-connection protocol splits TCP connection into wireless and wired parts at base station, and hides wireless loss from wired part connection. In MTCP [41], a specialized protocol in wireless link called selective repeat protocol (SRP) or UDP is used over wireless link. However, [41] reported that MTCP obtains no significant advantage in using SRP in comparison with TCP. In [11], M-TCP is used for wireless link.

Indirect-TCP (I-TCP) [6] uses standard TCP over wireless link. However, like other split-connection protocol, using TCP over wireless link results in performance degradation because original TCP sender often experiences timeout and data transmission stalls frequently. In addition, base station maintains two TCP connections; one is wireless part connection and the other is wired one. So the overhead of base station is twice in comparison with non-split-connection protocol and the base station needs much resources such as buffer capacity and CPU power. Furthermore, split-connection protocol violates the end-to-end semantics of TCP. In addition, since base station maintains two sets of TCP status information for one end-to-end TCP connection, handoff procedures become complicated and take long time. As a result, the split-connection protocols can hardly improve TCP performance in wired-wireless network.

The snoop protocol uses *Snoop* agent at base station. The agent monitors packets of each TCP connection and buffers unACKed packets. The agent retransmits packets instead of TCP sender if the agent detects packet loss by receiving triple duplicate ACKs or by local timeout. TCP sender is hidden from packet loss in wireless link by retransmission of snoop agent. WTCP [33] is using SACK and does not use local retransmission timer for loss recovery. WTCP can recovery from loss more effective than Snoop by using SACK. The one of advantages for using this type of protocol is that it suppresses duplicate ACKs and that it locally retransmits the buffered packet, and thereby it succeeds in avoiding unnecessary congestion control at TCP sender. Snoop agent needs to maintain per-connection TCP status like split connection protocol, but the overhead is smaller than split connection protocol. Snoop protocol has the same complexity of handoff procedure as split connection protocol.

The end-to-end protocol improves TCP performance keeping end-to-end semantics. The importance of keeping the end-to-end semantics [12] can never be overemphasized. In fact, keeping this semantics guarantees the transmission of data over any kind of heterogeneous network.

Furthermore, if we use IPsec of IPv6, keeping end-to-end principle is very important because intermediate router cannot monitor TCP packet header. TCP SACK option [19, 27] enhances information of packet loss. If packet loss occurs, original TCP retransmits all packets which has higher sequence number than that of lost packet. TCP SACK option gives detailed information of received and unreceived packets to the sender host. The sender host with TCP SACK option retransmits lost packet more effective than original TCP. So TCP performance can be improved by using SACK in environment where random loss occurs. TCP SACK option is proposed by RFC 1072 [19], however, the operation of sender side TCP when SACK packet is received is not defined in detail.

Freeze-TCP [15] uses one of standard TCP mechanisms, zero window advertisement (ZWA). When cellular handoff occurs, the mobile terminal sends ZWA to TCP sender to advertise the occurrence of cellular handoff. If the sender receives ZWA, the sender aborts transmission and keeps `cwnd` and `ssthresh` not changed. After handoff process is completed, the sender resumes transmission with the same sending rate as that before handoff. Freeze-TCP can improve TCP throughput performance if cellular handoff occurs. However, it does not consider the degradation of wireless link and TCP performance degrades when random packet loss occurs over wireless link.

TCP Westwood [26] performs available bandwidth estimation for the update of `cwnd` and `ssthresh` when fast retransmission or timeout occurs. By using available bandwidth estimation for setting `cwnd` and `ssthresh`, the sender does not reduce transmission rate needlessly and the throughput performance and link utilization is greatly improved. TCP Westwood achieves higher performance than TCP Reno in wired and wireless network environments. However, available bandwidth estimation algorithm is complex and the algorithm for available bandwidth estimation cannot follow the rapid change of network condition. Furthermore, TCP Westwood does not take the transmission of small-sized file such as Web document into consideration.

3 Proposed Algorithm

In this section, we describe our proposed algorithm in detail. The algorithm consists of two parts; one is the estimation of available bandwidth and the other is the update of slow start threshold and congestion window size using the bandwidth estimate.

3.1 Available Bandwidth Estimation

The proposed algorithm estimates network available bandwidth as follows. Let t_k denote the time at which original sender host receives kth ACK and d_k the amount of kth data segment sent to destination host. Whenever ACK reaches the sender host, the sender host calculates sample available bandwidth estimation $\mathrm{BW}_{Sample}(k)$ using the following.

$$\mathrm{BW}_{Sample}(k) = \frac{d_k}{t_k - t_{k-1}}. \tag{1}$$

Because ACK interval $t_k - t_{k-1}$ fluctuates under TCP window flow control, the value of $\mathrm{BW}_{Sample}(k)$ also fluctuates and is inaccurate estimate for available bandwidth. Therefore, the proposed algorithm smoothes $\mathrm{BW}_{Estimated}(k)$ using smoothing filter like other TCP variants including TCP Westwood. The filter implemented in TCP Westwood is well designed and provides more accurate estimates than other TCP variants proposed before. However, the filter of TCP Westwood is complicated and its computation time is not negligible, that is, computation

overhead is larger than other TCP variants. Furthermore, it cannot capture the rapid change of available bandwidth well as stated in the previous section.

We focus on the well-known simple smoothing filter, exponential weighted moving average (EWMA). The EWMA filter is expressed with α ($0 \leq \alpha \leq 1$) as

$$\begin{aligned} \mathrm{BW}_{Estimated}(k) &= \mathrm{BW}_{Estimated}(k-1) \times \alpha \\ &\quad + \mathrm{BW}_{Sample}(k) \times (1-\alpha). \end{aligned} \tag{2}$$

The idea based on the EWMA filter is that the more recent samples better reflect the current status in the network. Note that large α provides the estimate greatly affected by the past estimation results. Conversely, small α provides the estimate largely reflected by the current network status. However, [21] has reported that the EWMA filter based on autoregressive and moving average (ARMA) models may not work well because a sample sequence of delays such as RTT is a mixture of nonstationary and long-range dependent processes. Actually, it is difficult to obtain accurate estimates at the rapid change of network condition if the filter uses fixed α. For example, if the filter uses small α, the estimated value captures instantaneous rapid change well and hence small α is appropriate for the network whose condition changes instantaneously and rapidly as seen in wireless handoff. On the other hand, if α is set large, the estimate is less updated and this is preferable for the case in which the network condition doesn't change so much. In actual use of TCP over wired-wireless network, small-sized file transmission such as Web documents is a typical application. When a small-sized file transmission starts, TCP needs accurate available bandwidth estimate as soon as possible. This is the same situation as cellular handoff.

In order to obtain the high performance for the small-sized file transmission over wired-wireless network, we dynamically adapt α in (2) to the current network state. By dynamically updating α, $\mathrm{BW}_{Estimated}$ is quickly adapted when TCP session starts or network condition changes rapidly, and $\mathrm{BW}_{Estimated}$ is updated moderately when the network condition is stable.

When kth ACK is received, we first update α with $\mathrm{BW}_{Estimated}(k-1)$ and $\mathrm{BW}_{Sample}(k)$ in the following manner:

$$\begin{aligned} \text{if } (\mathrm{BW}_{Sample}(k) &\geq \mathrm{BW}_{Estimated}(k-1)) \\ \alpha &= \frac{\mathrm{BW}_{Estimated}(k-1)}{\mathrm{BW}_{Sample}(k)} \end{aligned} \tag{3}$$

$$\begin{aligned} \text{elsif } (\mathrm{BW}_{Sample}(k) &< \mathrm{BW}_{Estimated}(k-1)) \\ \alpha &= \frac{\mathrm{BW}_{Sample}(k)}{\mathrm{BW}_{Estimated}(k-1)}. \end{aligned} \tag{4}$$

Finally, updated α is substituted into (2) and kth estimate of available bandwidth $\mathrm{BW}_{Estimated}(k)$ is calculated. Then $\mathrm{BW}_{Estimated}(k)$ is used for setting slow start threshold (`ssthresh`) and congestion window (`cwnd`) in TCP congestion control algorithm.

3.2 Congestion Control Algorithm

The TCP Reno also updates `ssthresh` and `cwnd` and controls transmission rate when either timeout or retransmission by fast retransmit algorithm occurs. It is well known that Reno is likely to reduce transmission rate smaller than network available bandwidth and this causes throughput degradation. In order to avoid the throughput degradation, the proposed algorithm updates `ssthresh` and `cwnd` using available bandwidth estimate value $\mathrm{BW}_{Estimated}$.

The algorithm for updating `ssthresh` and `cwnd` is as follows.

Slow start threshold:

$$\texttt{ssthresh} \;=\; \frac{\mathrm{BW}_{Estimated} \times \mathrm{RTT}_{\min}}{\text{Packet Size}}, \tag{5}$$

where $\mathrm{RTT}_{\min}$ is the minimum of round trip time (RTT).

Congestion window:

the fast retransmission case;

$$\begin{aligned} \text{if } (\texttt{cwnd} &\geq \texttt{ssthresh}) \\ \texttt{cwnd} &= \texttt{ssthresh} \qquad (6) \\ \text{elsif } (\texttt{cwnd} &< \texttt{ssthresh}) \\ \texttt{cwnd} &\Rightarrow \text{keep previous value} \qquad (7) \end{aligned}$$

the timeout case;

$$\texttt{cwnd} \;=\; 1. \tag{8}$$

When retransmission by fast retransmit algorithm occurs, the proposed algorithm updates `ssthresh` with (5) and `cwnd` with (6) or (7). This is because the fast retransmission results from light congestion and it need not decrease the transmission rate so much.

When timeout occurs, the proposed algorithm updates `ssthresh` and `cwnd` by (5) and (8), respectively. This is because timeout indicates heavy congestion and the transmission rate needs to decrease.

4 Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm by simulation using network simulator ns2 [31]. We compare the proposed algorithm with the existing two TCPs: TCP Reno and TCP Westwood. To simulate TCP Westwood, we use TCP Westwood module for ns2 [38].

In the following, we assume that TCP uses delayed ACK mechanism with 200ms delay. TCP packet size is 1400 bytes while UDP packet size is 1000 bytes and the router buffer size is 100 packets. The initial window size is 100 segments and maximum window size is set to 2000 segments.

4.1 Effectiveness of Bandwidth Estimation

In this section, we investigate the effectiveness of the proposed bandwidth estimation algorithm. Simulation network topology is shown in Fig. 1. The bandwidth of wireless link is 10 Mbps and available bandwidth is also 10 Mbps. A single TCP connection is established and the packet loss rate of wireless link is 0.001% and 10%. We plot the data at 50 ms intervals because 50 ms is roughly equal to the mean RTT.

Fig. 2 shows the bandwidth estimates of TCP Westwood and the proposed algorithm when the rate of packet loss is equal to 0.001%. From Fig. 2, we observe that our proposed algorithm is more effective for available bandwidth estimation than TCP Westwood. Both estimates drop suddenly just after packet loss occurs. Then, the estimate of TCP Westwood increases gradually while that of the proposed algorithm recovers quickly. This results from the effect of dynamical

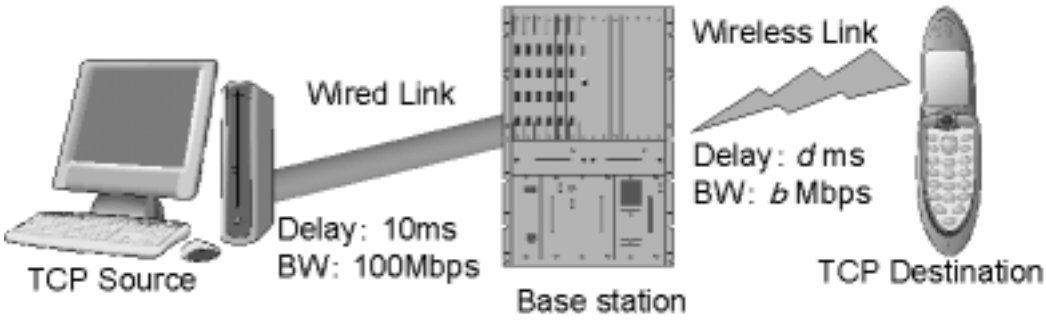


Figure 1: Network topology for wired-wireless network simulation.

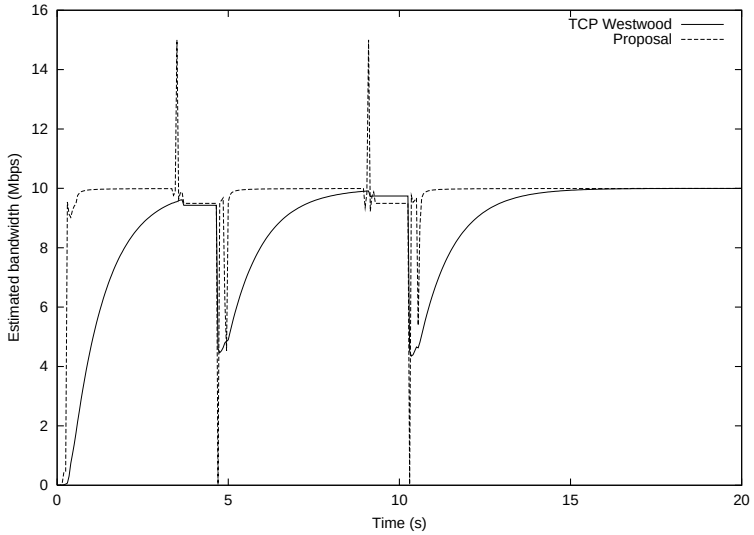
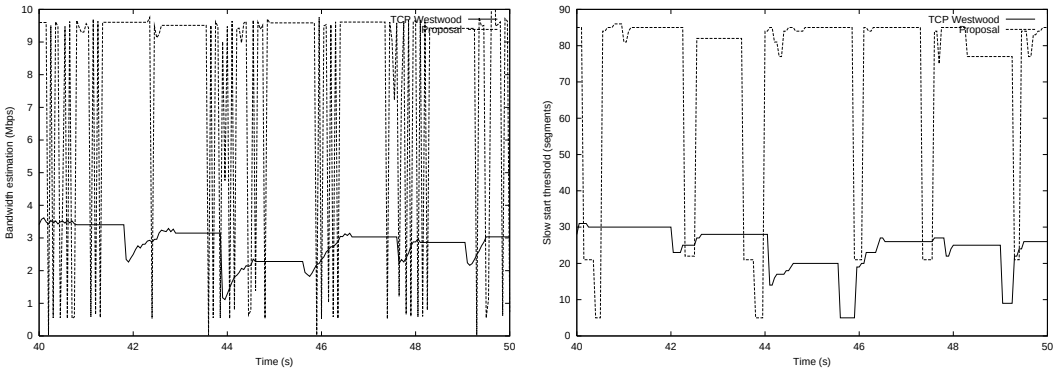


Figure 2: Effectiveness of bandwidth estimation (packet loss rate: 0.001%).



(a) Estimated bandwidth.

(b) Slow start threshold.

Figure 3: Effectiveness of bandwidth estimation (packet loss rate: 10%).

update of smoothing filter parameter α. This is a quite attractive feature of the proposed algorithm because high transmission rate is achieved just after TCP session starts. Furthermore, in small-sized file transmission, TCP session is short lived and our proposed algorithm is also effective for such short lived session.

Fig. 3 illustrates bandwidth estimates and slow start threshold with packet loss rate of wireless link equal to 10%. Fig. 3(a) shows the bandwidth estimates and we observe that the bandwidth estimate of TCP Westwood is far lower than 10 Mbps, actual available bandwidth. On the other hand, the bandwidth estimate of our proposed algorithm oscillates with high frequency after each detection of packet loss. This is a weak point of our algorithm, however, the resulting estimate is close to the actual available bandwidth. In addition, because of updating `cwnd` and `ssthresh` is less frequently than the update of bandwidth estimate, both `cwnd` and `ssthresh` hardly oscillate as shown in Fig. 3(b). Therefore, the available bandwidth estimation of our proposed algorithm is more effective than that of TCP Westwood.

4.2 Fairness Issue

The fairness of bandwidth sharing means that all connections have the same opportunity to transmit data. In this subsection, we investigate how fair-share is achieved in the proposed algorithm, TCP Reno and TCP Westwood. The simulation topology is shown in Fig. 4. There are two TCP sources in the network; one is connected to router with 100 Mbps wired link whose propagation delay is 10 msec and the other is connected to the router with the same wired link except 20 msec propagation delay. A TCP connection is established between each source and destination hosts for FTP session and each source sends infinitely large-sized file by FTP for 300 s. Fig. 5 illustrates how the sequence number increases against time for each TCP protocol. It is well known that the bandwidth used for TCP session with short RTT is likely to become large while that with long RTT tends to be small. We observe this tendency in Fig. 5. However, the discrepancy in TCP Reno is the largest while that in the proposed algorithm is the smallest. This implies that the proposed algorithm provides better fair-share than TCP Reno and TCP Westwood even when sessions with different RTT are multiplexed. The reason is that the available bandwidth estimate is close to actual available bandwidth and this prevents long RTT connection from needless reduction of `cwnd` and `ssthresh`.

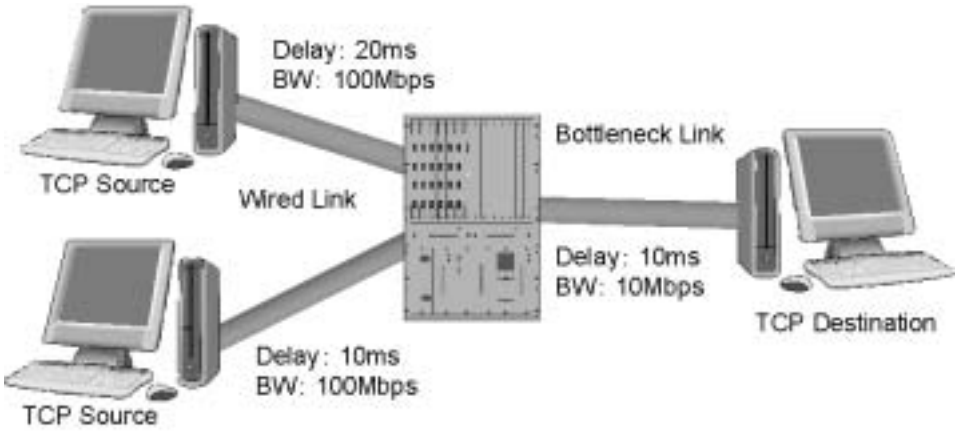


Figure 4: Network topology for fairness simulation.

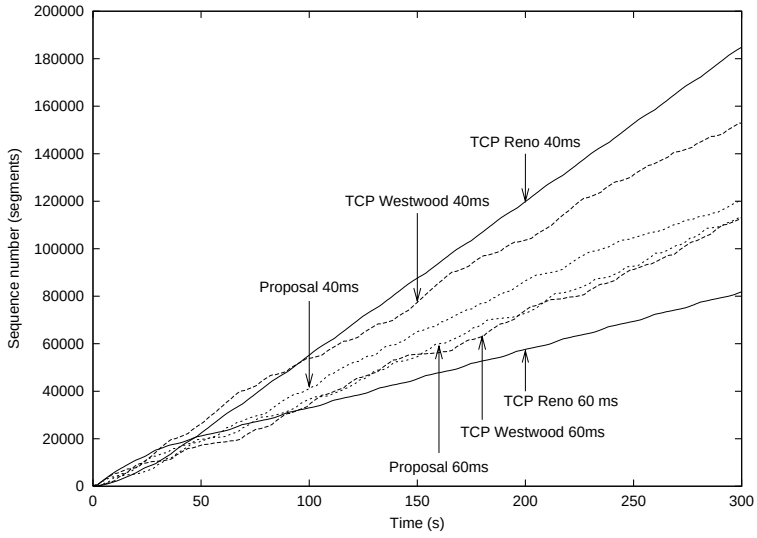


Figure 5: Sequence number vs. time for long and short RTT connections.

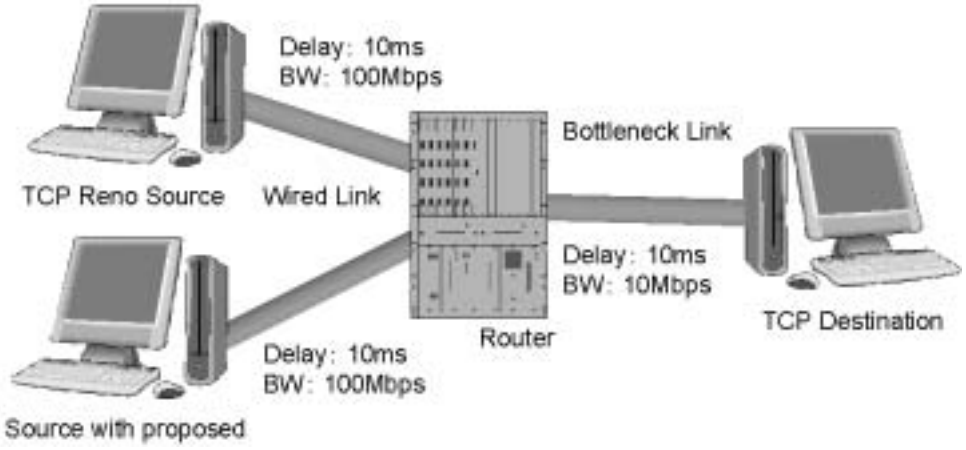


Figure 6: Network topology for friendliness simulation.

4.3 Friendliness Issue

Currently, it is important for modified TCP variant to achieve good friendliness with TCP Reno in its standardization. Here, the friendliness means fair bandwidth sharing among different data transmission protocols. We evaluate the friendliness between the proposed algorithm and TCP Reno by simulation. The network topology is shown in Fig. 6. In Fig. 6, the senders of TCP Reno and the proposed algorithm are connected to router with 100 Mbps wired link whose propagation delay is 10 ms. The link between router and destination is 10 Mbps wired link with 10 ms propagation delay. Each TCP connection is established between each source and destination hosts for FTP session and each source sends infinitely large-sized file for 300 s.

From Fig. 7, both the sequence numbers are growing in a similar manner. This implies that the proposed algorithm achieves fair-share of bottleneck bandwidth with TCP Reno and this is the advantage for standardization.

4.4 Bulk Data Transmission

In this scenario, source host sends infinitely large-sized file by FTP for 1000 s. This is the case of FTP session where CD-ROM image or multimedia data file is transmitted. We evaluate the proposed algorithm both in wired network environment and in wired-wireless network environment. The performance measure is the average of TCP throughput during 1000 s time interval.

4.4.1 Impact of bottleneck bandwidth on TCP throughput over wired network

For wired network simulation experiments, we consider a network model shown in Fig. 8. In Fig. 8, both links for TCP and UDP sources connected to router are wired ones with 100Mbps bandwidth and 10ms propagation delay. The link between router and destination host is also wired link whose bandwidth and propagation delay are variable. UDP traffic is sent from the UDP sender to the destination host as background traffic. UDP traffic is constant bit rate (CBR) traffic and the bit rate changes in the range from 0 Mbps to bottleneck bandwidth, and the interval between time epochs at which the bit rate changes is exponentially distributed. The mean length of the interval is set to 1 s or 10 s.

In Fig. 9, the horizontal axis means bottleneck bandwidth and the vertical axis represents

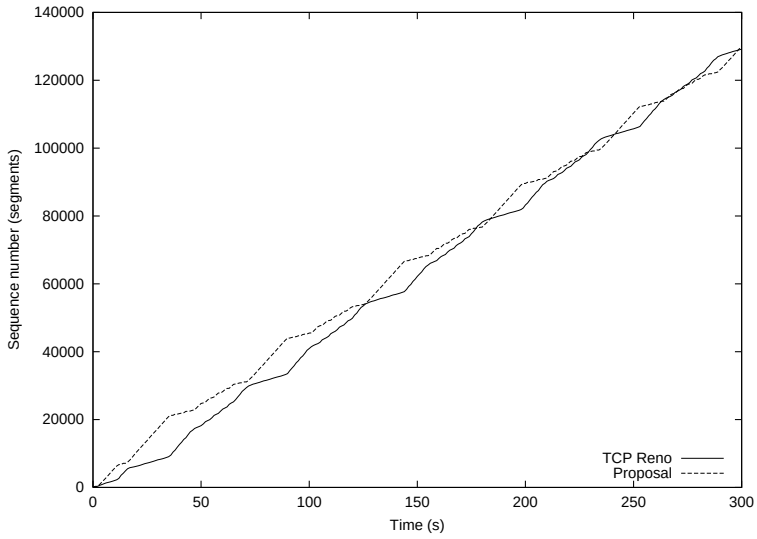


Figure 7: Sequence number vs. time for TCP Reno and proposal.

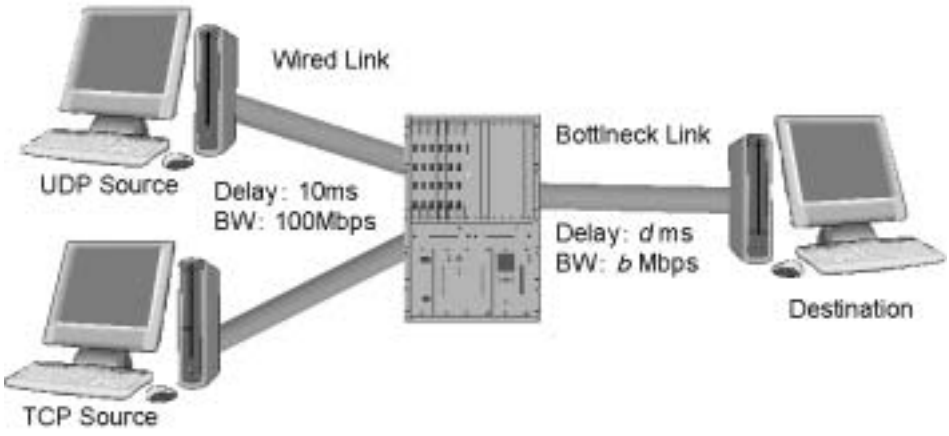


Figure 8: Network topology for wired network simulation.

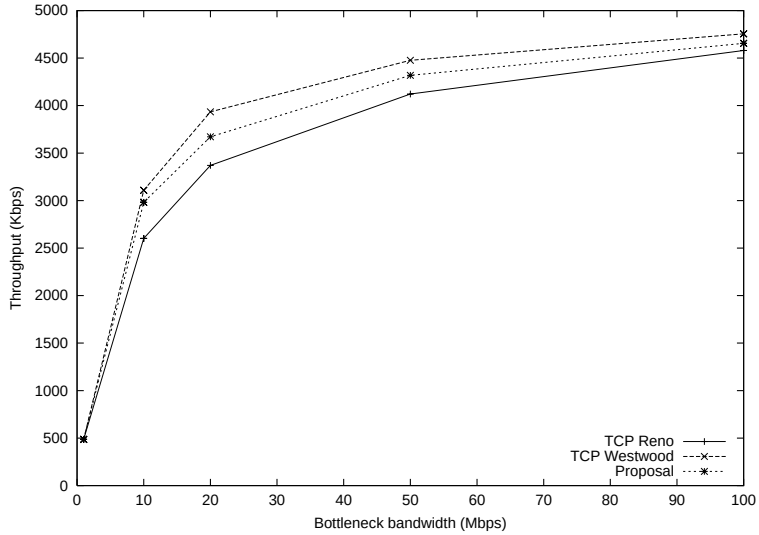


Figure 9: Throughput vs. bottleneck bandwidth (bulk data transmission in wired network model), bottleneck link delay: 100 ms, mean burst time: 10 s.

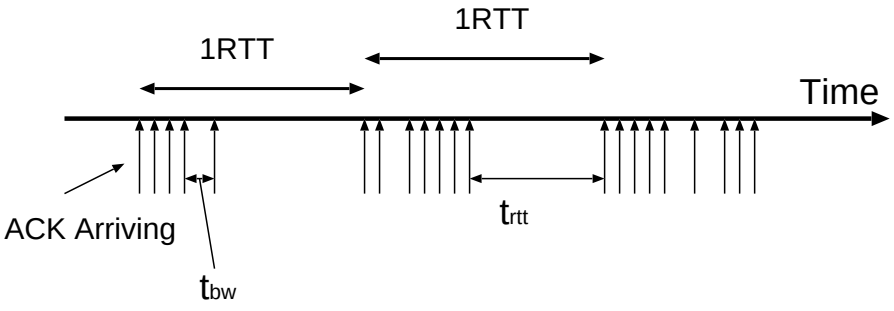


Figure 10: ACK interval.

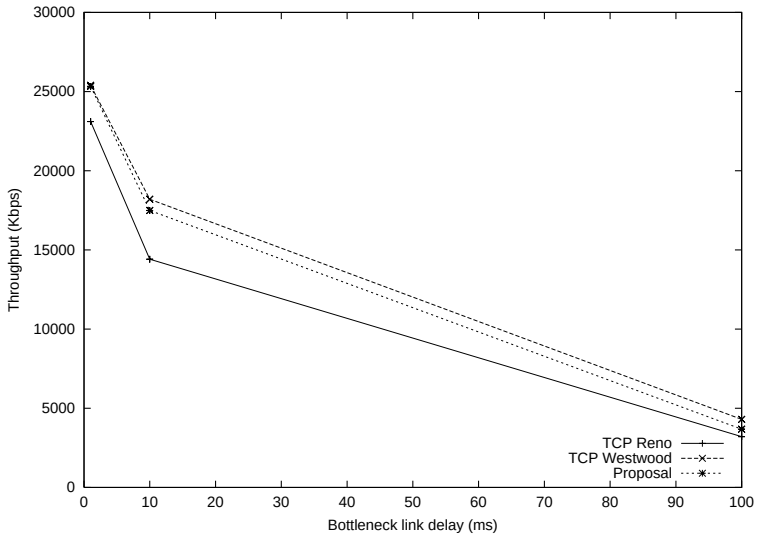


Figure 11: Throughput vs. bottleneck link delay (bulk data transmission in wired network model), bottleneck bandwidth: 100 Mbps, mean burst time: 1 s.

resulting throughput. We observe that the proposed algorithm achieves higher throughput performance than TCP Reno for any bottleneck bandwidth. However, the throughput of proposed algorithm is smaller than that of TCP Westwood. The main reason of this performance degradation is dynamical update of the filter parameter α. Fig. 10, illustrates how ACKs are received by the TCP sender host. In Fig. 10, the time interval of consecutive ACK arrivals is roughly classified into two patterns which result from window flow control mechanism; one is the interval reflected by bottleneck bandwidth (t_{bw}), and the other is the interval reflected by RTT (t_{rtt}). Note that t_{rtt} sometimes becomes much larger than t_{bw}. If some ACK interval becomes quite large, the sample available bandwidth BW_{Sample} becomes small. In our algorithm, the rapid change of BW_{Sample} greatly affects the bandwidth estimate $\mathrm{BW}_{Estimated}$. As a result, the longer RTT becomes, the smaller bandwidth estimate is obtained and the throughput performance of the proposed algorithm is worse than TCP Westwood in long RTT environment. This problem can be solved by using more sophisticated filter and this is our future work.

Fig. 11 illustrates how the throughput changes against bottleneck link delay. The throughput of the proposed algorithm is less than TCP Westwood as the wireless link delay becomes large. This is because large bottleneck link delay causes the long RTT.

4.4.2 Impact of packet loss rate on TCP throughput over wired-wireless network

In order to investigate throughput performance in wired-wireless network, we use the network model as shown in Fig. 1. The link between sender host and base station is wired link with 100Mbps bandwidth and 10ms propagation delay, and the link between base station and destination host is wireless link where bandwidth and propagation delay are variable. In this scenario, as for wireless communication error model, we consider a Markov burst error model in which the state of wireless link consists of good and bad [1, 26] (Fig. 12). In each state, packet loss occurs independently with loss probability depending on the state. The time interval between errors is thus exponentially distributed. The time spent in each state is also exponentially distributed. The mean durations of good and bad states are 1 s and 10 s, respectively. In good state, loss

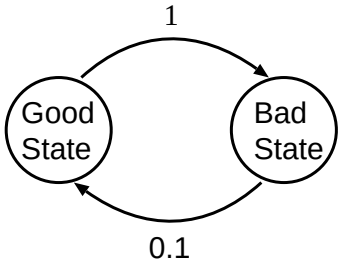


Figure 12: Markov burst error model.

rate of wireless link is 0.001% while in bad state, loss rate is set to the value in the range from 1% to 30%.

In Fig. 13, each graph plots the throughput against packet loss rate in bad state. In Fig. 13, figures from 13(a) to 13(e) represent cases of bottleneck bandwidth equal to 1, 10, 20, 50 and 100 Mbps, respectively. From Fig. 13, we observe that our proposed algorithm achieves higher performance than TCP Reno and TCP Westwood in any case. In Figs. 13(a), 13(b) and 13(c), our proposed algorithm achieves similar performance to that of TCP Westwood. Note that in these results, the bandwidth delay products are not so large. If bandwidth delay products is small, `cwnd` is also small. In this condition, even though the actual available bandwidth changes rapidly, the change of `cwnd` is small and the difference between `cwnd` by accurate estimation and `cwnd` by inaccurate estimation is also small. So our proposed algorithm with rapid adjustment of bandwidth estimate to actual bandwidth is not effective. However, in Figs. 13(d) and 13(e), proposed algorithm achieves higher performance than that of TCP Westwood because the bottleneck bandwidth, and hence bandwidth delay products is large enough.

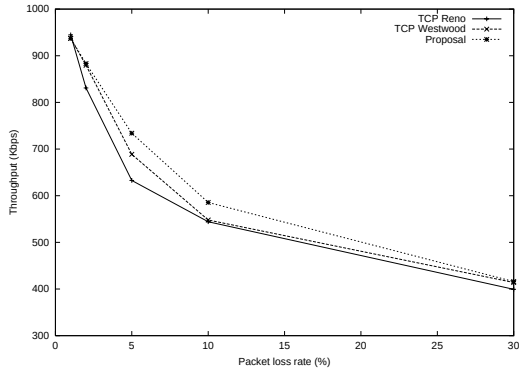
4.5 Small Sized-File Transmission

We have investigated the TCP performance for bulk data transmission where TCP session is long lived. In the small-sized document transmission like a Web document, however, TCP session is short lived. It is expected that the algorithm which achieves good throughput performance in bulk data transmission may degrade the performance with small-sized file transmission. In this subsection, we evaluate our proposed algorithm in small-sized file transmission. We assume that the source host sends some small-sized files by FTP. The file size is exponentially distributed with mean varying from 1KB to 1000KB. The source host sends 10,000 files in each simulation and the performance measure is the average of TCP throughput.

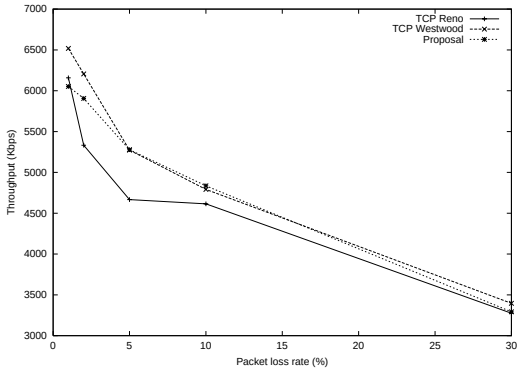
4.5.1 Wired environment

In this scenario, we use the network topology shown in Fig. 1 for simulation model. The background UDP is CBR whose bit rate is half of bottleneck bandwidth. The bottleneck bandwidth is in the range from 1 Mbps to 100 Mbps.

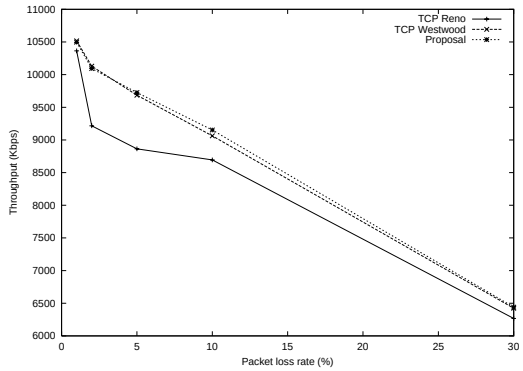
In Fig. 14, the horizontal axis means bottleneck bandwidth and the vertical axis represents resulting throughput. As shown in Fig. 14, the throughput performance of the proposed algorithm is similar to those of TCP Reno and TCP Westwood when bottleneck bandwidth is larger than 50 Mbps. Because that bottleneck bandwidth is large enough to transmit small-sized files, the packet loss rarely occurs at router buffer. On the other hand, the bottleneck bandwidth



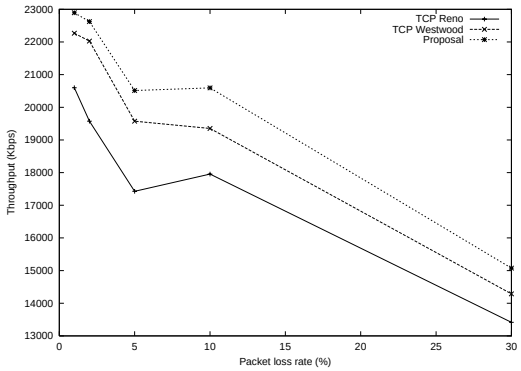
(a) Bottleneck bandwidth: 1Mbps



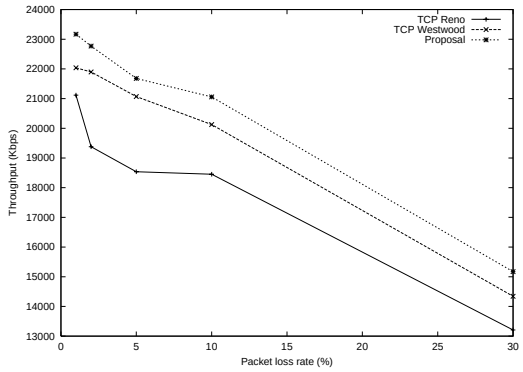
(b) Bottleneck bandwidth: 10Mbps



(c) Bottleneck bandwidth: 20Mbps



(d) Bottleneck bandwidth: 50Mbps



(e) Bottleneck bandwidth: 100Mbps

Figure 13: Throughput vs. packet loss rate in bad state (bulk data transmission in burst loss model), wireless link delay: 1 ms, mean burst time: 10 s.

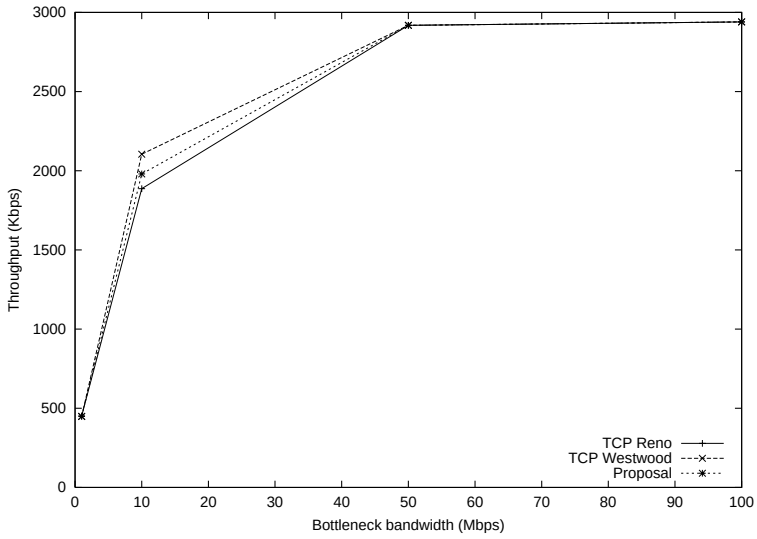


Figure 14: Throughput vs. bottleneck bandwidth (small-sized file transmission in wired network model), mean file size: 200 KB, bottleneck link delay: 10 ms.

is in the range from 1 Mbps to 50 Mbps, the throughput performance of the proposed algorithm is larger than TCP Reno but smaller than TCP Westwood. This is because the available bandwidth does not change so much. As we discussed in 4.4.1, the bandwidth estimation of our proposed algorithm is likely to be under-estimated and this results in the less performance of the proposed algorithm than TCP Westwood.

Fig. 15 shows the throughput against mean file size. In Fig. 15, when the mean file size is larger than 200 KB, the proposed algorithm provides the worst performance among three. This is because most of TCP connections are long lived in this case. On the other hand, when mean file size is smaller than 200 KB, the throughput of the proposed algorithm is almost the same as TCP Reno and TCP Westwood. However, our proposed algorithm is less throughput performance than TCP Westwood for any mean file size. This is due to the effect of underestimation of available bandwidth as discussed in 4.4.1.

4.5.2 Independent loss model

In this scenario, wireless error occurs according to independent (Bernoulli) loss model [26] and the packet loss rate is in the range from 1% to 30%. The bottleneck bandwidth is in the range from 1 Mbps to 100 Mbps and the range of wireless link delay is from 1 ms to 100 ms.

Fig. 16 illustrates the throughput against the mean file size. Figs. 16(a), 16(b) and 16(c) are cases with wireless link delay equal to 1 ms, 10 ms and 20 ms, respectively. In Figs. 16(a) and 16(b), we observe that our proposed algorithm achieves higher performance than existing TCP Reno and TCP Westwood. This is because new TCP connection is established for each file transmission and the bandwidth estimate of TCP Westwood does not fails in predicting actual available bandwidth just after the session starts even though the proposed algorithm can estimate actual available bandwidth very quickly. If packet loss occurs just after session start point, TCP Westwood greatly reduces its transmission rate while the proposed algorithm does not perform needless transmission rate reduction. In Fig. 16(c) where wireless link delay is 20 ms, our proposed algorithm achieves higher performance than TCP Reno but smaller than that

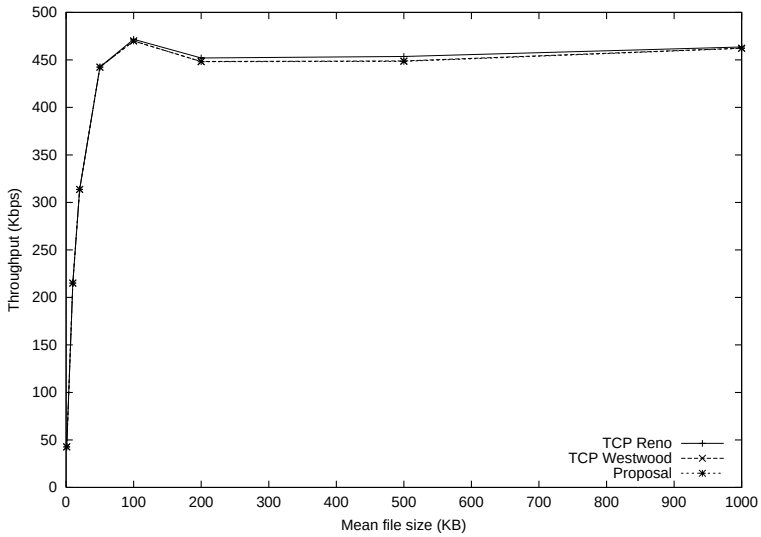


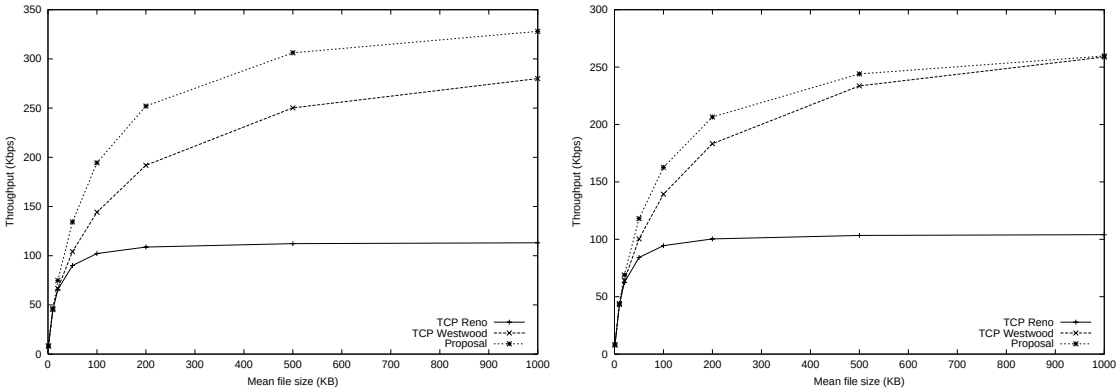
Figure 15: Throughput vs. mean file size (small-sized file transmission in wired network model), bottleneck bandwidth: 10 Mbps, bottleneck link delay: 10 ms.

of TCP Westwood. This results from the underestimation of available bandwidth due to large end-to-end RTT. Fig. 16 shows that the proposed algorithm is effective in the wireless link with small delay.

Fig. 17 illustrates the throughput against the mean file size. From Fig. 17, when the bottleneck bandwidth becomes large, the throughput of our proposed algorithm becomes smaller than TCP Westwood. This also results from underestimation of available bandwidth. The end-to-end RTT does not change in this case, but when the bottleneck bandwidth increase, ACK intervals tend to become small. Then the end-to-end RTT becomes much larger than ACK interval and this causes the underestimation of available bandwidth in our proposed algorithm. In small bottleneck bandwidth, however, the proposed algorithm achieves higher performance than TCP Reno and TCP Westwood. Note that new TCP connection is established for every file in Web document transmission model. In this situation, the proposed algorithm estimates available bandwidth estimation accurately just after session start point and this avoids needless transmission rate degradation when packet loss occurs.

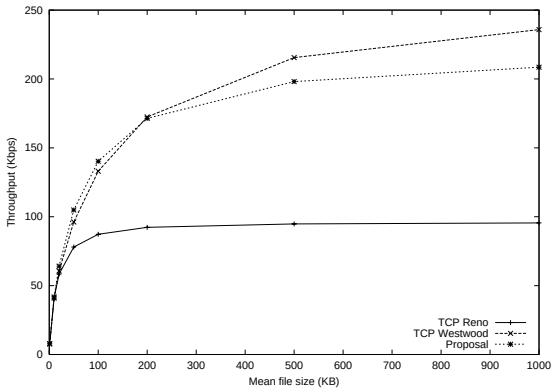
Fig. 18 also illustrates the throughput against the mean file size. In Fig. 18, we observe that our proposed algorithm achieves higher throughput performance than TCP Reno and TCP Westwood for any packet loss rate. This implies that our proposed algorithm is effective for small-sized file transmission in wired-wireless network environment. When both RTT and bottleneck bandwidth are small, the proposed algorithm does not underestimate actual available bandwidth and this results in higher performance than TCP Reno and TCP Westwood. In the case that packet loss rate is 10%, the proposed algorithm achieves higher performance than TCP Westwood. When packet loss occurs, TCP Westwood largely decreases the available bandwidth estimates and slowly updates the estimate. On the other hand, the proposed algorithm estimates actual available bandwidth very quickly after packet loss occurs. So the proposed algorithm is more effective over high packet loss rate link.

From numerical results in this section, it is shown that the proposed algorithm is effective in wired-wireless network and achieves good fairness and TCP friendliness. In particular, the proposed algorithm achieves very high performance for small-sized file transmission. The reason



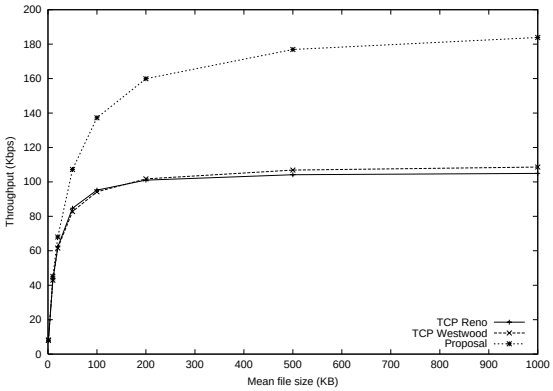
(a) Wireless link delay: 1ms

(b) Wireless link delay: 10ms

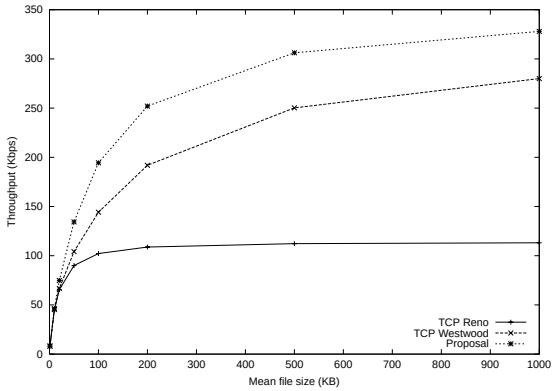


(c) Wireless link delay: 20ms

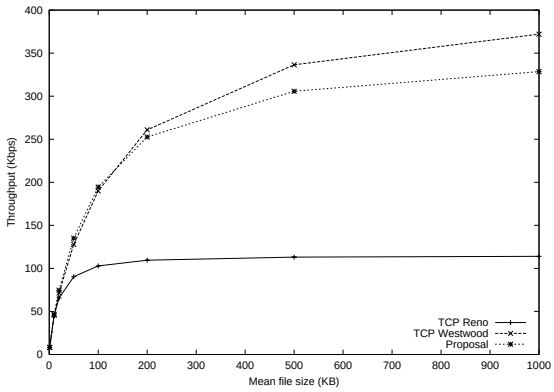
Figure 16: Throughput vs. mean file size (small-sized file transmission in independent loss model), bottleneck bandwidth: 10 Mbps, packet loss rate: 10%.



(a) Bottleneck bandwidth: 1Mbps

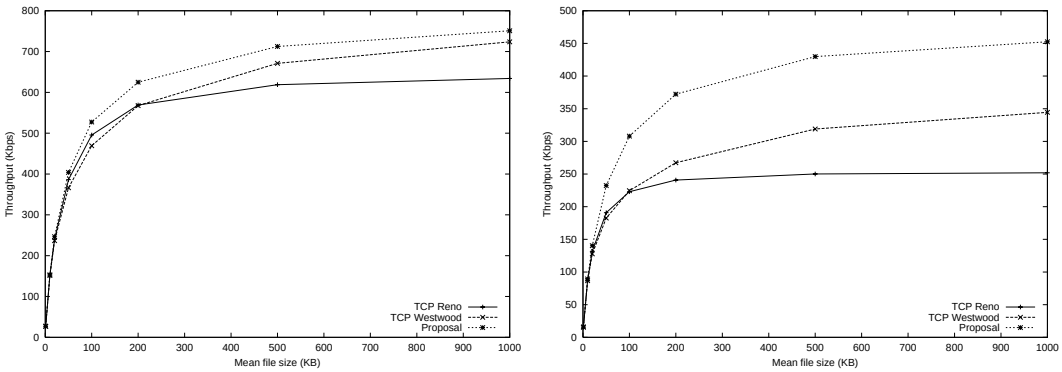


(b) Bottleneck: 10Mbps



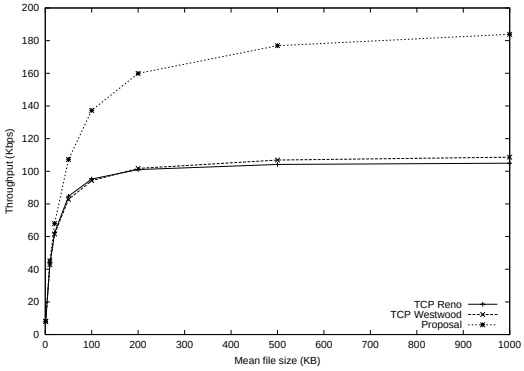
(c) Bottleneck bandwidth: 100Mbps

Figure 17: Throughput vs. mean file size (small-sized file transmission in independent loss model), wireless link delay: 1 ms, packet loss rate: 10%.

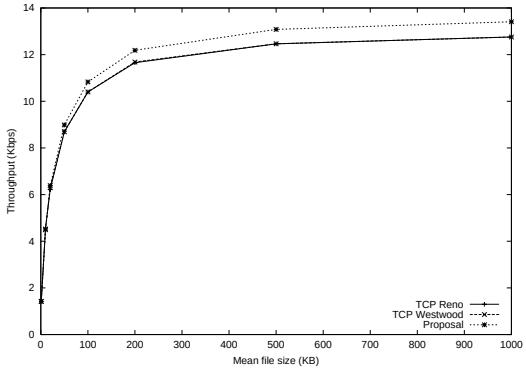


(a) Packet loss rate: 2%

(b) Packet loss rate: 5%



(c) Packet loss rate: 10%



(d) Packet loss rate: 30%

Figure 18: Throughput vs. mean file size (small-sized file transmission in independent loss model), wireless link delay: 1 ms, bottleneck bandwidth: 1 Mbps.

why the proposed algorithm achieves higher performance than existing TCP Reno and TCP Westwood is that the proposed algorithm estimates actual available bandwidth quickly and this prevents from needless degradation of transmission rate. Further advantage of the proposed algorithm is that the proposed algorithm is for sender side and does not need any modification of receiver-side TCP. This is great advantage for standardization of the Internet protocol. In addition, the proposed algorithm has no parameters to be pre-determined and no parameter tuning is needed.

5 Conclusion

In this paper, we proposed the algorithm for improving TCP performance for the transmission of small-sized file such as Web document. For this purpose, our proposed algorithm adapts to the rapid change of network available bandwidth by dynamical updating the key parameter of bandwidth estimation filter. We evaluated our proposed algorithm's performance by simulation. We observed that the fair-share of bottleneck bandwidth between two different RTT connections is achieved using our proposed algorithm and that our proposed algorithm also achieved fair-share of bottleneck bandwidth with existing TCP Reno. These results are the advantage for protocol standardization.

Moreover, we showed that the proposed algorithm can achieve higher throughput than existing TCP Reno and TCP Westwood even though our proposed algorithm uses simple bandwidth estimation filter. In particular, our proposed algorithm is effective for small-sized file transmission which is a typical application in the wired-wireless network environment.

As a future research work, the refinement of the available bandwidth estimation filter for preventing from underestimation should be investigated.

References

[1] A. A. Abouzeid, R. Sumit, and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proc. IEEE INFOCOM 2000, Tel-Aviv, Israel*, pp. 1724–1733, Mar. 2000.

[2] M. Allman, V. Paxson, and W. R. Stevens, "TCP congestion control," *RFC 2581*, Apr. 1999.

[3] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Prentice Hall, 1997.

[4] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL: A link-layer protocol for wireless networks," *ACM Wireless Networks*, vol. 1, no. 1, pp. 47–60, 1995.

[5] A. Bakre and B. R. Badrinath, "Handoff and system support for indirect TCP/IP," in *Proc. MLICS '95 Ann Arbor, MI, U.S.A.*, Apr. 1995.

[6] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," *ICDCS '95*, Oct. 1995.

[7] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Improving performance of TCP over wireless networks," in *Proc. ICDCS '97, Baltimore, Maryland, U.S.*, May 1997.

[8] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[9] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, no. 4, 1995.

[10] R. K. Balan, L. B. Peng, K. R. Kumar, L. Jacob, W. K. G. Seah, and A. L. Ananda, "TCP HACK: TCP header checksum option to improve performance over lossy links," in *Proc. IEEE INFOCOM 2001, Anchorage, Alaska*, Apr. 2001.

[11] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Communication Review*, vol. 27, no. 5, 1997.

[12] D. D. Clark, "The design philosophy of the DARPA internet protocols," in *Proc. ACM SIGCOMM '88, Stanford, CA*, (Stanford, CA), pp. 106–114, ACM, Aug. 1988.

[13] D. A. Eckhardt and P. Steenkiste, "Improving wireless LAN performance via adaptive local error control," in *Proc. IEEE ICNP '98, Austin, Texas*, Oct. 1998.

[14] M. Gerla, W. Weng, and R. L. Cigno, "BA-TCP: A bandwidth aware TCP for satellite networks," in *Proc. IEEE ICCCN'99, Boston, Massachusetts*, Oct. 1999.

[15] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *Proc. IEEE INFOCOM 2000, Tel-Aviv, Israel*, pp. 1537–1545, Mar. 2000.

[16] H. Inamura, "TCP over 2.5G and 3G wireless networks." IETF, Internet draft, Aug. 2001.

[17] H. Inamura and T. Ishikawa, "A TCP profile for W-CDMA: 3G wireless packet services." IETF, Internet draft, Aug. 2000.

[18] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM '88, Stanford, CA*, pp. 314–329, Aug. 1988.

[19] V. Jacovson and R. Braden, "TCP extensions for long-delay paths," *RFC 1072*, Oct. 1988.

[20] P. Karn, "The qualcomm CDMA digital cellular system," in *Proc. MLICS '93, Cambridge, Massachusetts*, pp. 35–40, Aug. 1993.

[21] Q. L. Li and D. L. Mills, "Jitter-based delay-boundary prediction of wide-area networks," *IEEE/ACM Trans. Networking*, vol. 9, pp. 578–590, Oct. 2001.

[22] S. Li, S. Chong, and C. Hwang, "Link capacity allocation and network control by filtered input rate in high-peed networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 1, pp. 10–25, 1995.

[23] S. Lin, D. J. Costello, and M. J. Miller, "Automatic repeat error control schemes," *IEEE Communication Magazine*, vol. 22, pp. 5–17, Dec. 1984.

[24] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "Forward error correction (FEC) building block," *RFC 3452*, Dec. 2002.

[25] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The use of forward error correction (FEC) in reliable multicast," *RFC 3453*, Dec. 2002.

[26] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proc. ACM MobiCom 2001, Roma, Italy*, pp. 287–297, July 2001.

[27] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," *RFC 2018*, Oct. 1996.

[28] D. J. Mitzel, "Overview of 2000 IAB wireless internetworking workshop," *RFC 3002*, Dec. 2000.

[29] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, and N. Vaidya, "Long thin networks," *RFC 2757*, Jan. 2000.

[30] Y. Nishida, "Thought of TCP enhancement in wireless networks," in *Proc. IPSJ SIGNotes MoBiLe computing and wireless communications No.014*, Sept. 2000. (in Japanese)

[31] "The Network Simulator – ns-2." Available from `http://www.isi.edu/nsnam/ns/`.

[32] J. Postel, "Transmission control protocol," *RFC 793*, Sept. 1981.

[33] K. Ratnam and I. Matta, "WTCP: An efficient transmission control protocol for networks with wireless links," in *Proc. IEEE/ISCC '98, Athens, Greece*, June 1998.

[34] N. Sato, K. Mitsunobu, and F. Teraoka, "TCP–J: New transport protocol for wireless network environments," *IPSJ*, vol. 43, pp. 3848–3858, Dec. 2002. (in Japanese)

[35] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *ACM Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.

[36] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

[37] W. R. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *RFC 2001*, Jan. 1997.

[38] "TCP Westwood module for ns-2." Available from `http://www.cs.ucla.edu/NRL/hpi/tcpw/`.

[39] V. Tsaoussidis and H. Badr, "TCP-Probing: Towards an error control schema with energy and throughput performance gains," in *Proc. IEEE ICNP 2000, Osaka, Japan*, Nov. 2000.

[40] N. Vaidya, M. Mehta, P. Charles, and G. Montenegro, "Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless," tech. rep., Computer Science Dept., Texas A&M University, 1999.

[41] R. Yavatkar and N. Bhagawat, "Improving end-to-end performance of TCP over mobile internetworks," in *Proc. IEEE WMCSA '94, Santa Cruz, California, U.S.*, 1994.