NJ+: An Efficient Congestion Control Mechanism for Wireless Networks

Jaehyung Lee[1], Jungrae Kim[1], Minu Park[1], Jahwan Koo[2], and Hyunseung Choo[1]
[1]Sungkyunkwan UniversitySchool of Information and Communication Engineering, Korea
[e-mail: jhyunglee@skku.edu, witjung@ece.skku.ac.kr, minupark@skku.edu, choo@skku.edu]
[2]Computer Sciences Department University of Wisconsin-Madison, USA
[e-mail: jhkoo@cs.wisc.edu]
*Corresponding author: Hyunseung Choo

Received November 11, 2008; revised December 8, 2008; accepted December 9, 2008; published December 25, 2008

Abstract

Transmission control protocols have to overcome common problems in wireless networks. TCP employing both packet loss discrimination mechanism and available bandwidth estimation algorithm, known as the good existing solution, shows significant performance enhancement in wireless networks. For instance, TCP New Jersey which exhibits high throughput in wireless networks intends to improve TCP performance by using available bandwidth estimation and congestion warning. Even though it achieves 17% and 85% improvements in terms of goodput over TCP Westwood and TCP Reno, respectively, we further improve it by exploring maximized available bandwidth estimation, handling bit-error-rate error recovery, and effective adjustment of sending rate for retransmission timeout. Hence, we propose TCP NJ+, showing that for up to 5% packet loss rate, it outperforms other TCP variants by 19% to 104% in terms of goodput when the network is in bi-directional background traffic.

Keywords: Transport control protocol (TCP), congestion control, congestion, wireless link bit error rate (BER), retransmission timeout (RTO), TCP New Jersey

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-(C1090-0801-0046)).

DOI: 10.3837/tiis.2008.06.004

1. Introduction

Transmission control protocol (TCP) provides reliable data transmission in wired networks [1][2]. However, TCP in wireless networks, such as wireless LANs, mobile, ad hoc networks, cellular networks, and wireless mesh networks, suffers from the repeated packet losses due to generic characteristics including limited bandwidth, high bit-error-rate (BER), handover, channel interference, and fading which bring performance degradation [3]. Even though recent researches focus on TCP performance during handover [4][5][6], the basic reason for the performance degradation is that TCP congestion control [2] mechanism cannot discriminate between the packet losses caused by wireless link errors and those caused by network congestion, thus, reacting to these losses by reducing its congestion window (*cwnd*). Therefore, these inappropriate reductions of the *cwnd* lead to unnecessary throughput degradation [7].

Over the last decade, a considerable number of studies have been conducted for the improvement of wireless TCP performance with the advances of wireless infrastructure technologies [8]. The wireless TCP schemes can be divided into split and end-to-end approaches [9]. The split approach attempts to prevent the wireless portion from the wired network by separating the TCP connection at the base station. The base station behaves as a terminal (or a proxy) in both the wired and wireless portions. Both end hosts communicate with the base station independently without knowledge of the other end. The drawback of split approach is that it violates end-to-end TCP semantic. On the other hand, the end-to-end approaches, such as TCP New Reno [10], Westwood [11], Jersey [12], and New Jersey [13], deal with the route from the sender to the receiver as an end-to-end path, and the sender is acknowledged directly by the receiver. The receiver provides feedback reflecting the network condition, and the sender makes decisions for sending rate.

TCP Westwood improves addictive increase multiplicative decrease (AIMD) [14] which is the conventional congestion window adjustment strategy of the regular TCP and intends to improve TCP performance by effectively adjusting its transmission rate on the basis of the available bandwidth estimation (ABE) algorithm at the sender. Although various TCP Westwood variants are studied, they have the same concept of using ABE at the sender side. On the other hand, TCP Jersey and New Jersey are based on the integration of the sender-side ABE algorithm and the packet loss differentiated scheme in the intermediate router, thus, resulting in higher throughput. TCP Westwood, TCP Jersey, and New Jersey are good solutions in wireless networks assuming decent signaling environment. Nowadays, because many wireless technologies including WiBro, Bluetooth, HSDPA, and so on, are mixed in our airspace, BER is increased temporarily because of the interference with each other. Hence an efficient method robust to the high BER environments is needed based on existing TCP techniques.

Although TCP New Jersey achieves 17% and 85% improvements in goodput over TCP Westwood and TCP Reno, respectively, we further increase TCP New Jersey performance by employing maximized available bandwidth estimation (MABE), handling BER error recovery (BERR), and effective adjustment of sending rate (EASR) for retransmission timeout mechanisms. In MABE, maximum estimation value is selected between sender side estimation and receiver side one. BERR and EASR reduce recovery time to compensate dropped sending rate. Hence, we propose TCP NJ+, showing up to 5% high BER wireless link error rate, it outperforms other TCP variants by 19% to 104% in terms of goodput regardless of background traffic when the network is in bi-directional congestion. Although our proposed

schemes, such as BERR and EASR, implement based on TCP New Jersey, it is possible to employ other TCP variants using packet loss discrimination schemes.

The rest of the paper is organized as follows: Section 2 reviews the related works of the existing wired and wireless TCP schemes. Section 3 describes the improved mechanisms of TCP NJ+ in detail. Section 4 presents performance evaluation via network simulator [15] under various network conditions. The final section offers some concluding remarks.

2. Preliminaries

In this section, we describe typical transport protocols TCP Reno and New Reno in wired networks. We also discuss TCPs in wireless networks. Here we describe TCP Westwood, Jersey, and New Jersey which are well-known methods discussed in the literature.

2.1 TCP Reno

TCP Reno is a standard TCP for the effective and reliable data transfer in wired networks. In the fast recovery algorithm [16] of TCP Reno which is the additional scheme comparing with TCP Tahoe [1], when the packet loss is detected at the sender, it enters the congestion avoidance phase instead of returning to the slow start phase after the lost packet is retransmitted. Hence, fast recovery inflates the reduced transmission rate quickly.

The fast recovery algorithm takes care of a single packet loss effectively within one congestion window(*cwnd*). But TCP Reno has a problem that if the multiple packet drops occur, it would be forced to invoke multiple fast recovery algorithms repeatedly, slowing down the sending rate continuously. Namely, it induces the unnecessary decreasing of sending rate in wireless networks because the packet is dropped frequently.

2.2 TCP New Reno

TCP New Reno improves fast recovery that brings congestion avoidance phase after performing fast retransmit [14] if the packet loss is detected by the sender. The multiple packet losses force TCP Reno to invoke slow down the recovery of the dropped transmission rate. TCP New Reno handles multiple packet losses for one congestion window. In TCP New Reno, the fast recovery does not terminate until receiving of full ACK. When the sender receives 3 duplicate acknowledgements (3-DUPACKs), and each is the partial ACK, it just retransmits the lost packet and does not terminate the fast recovery which does not reduce the *cwnd*. Hence, the *cwnd* is decreased once when multiple packet losses are occurred. Namely, TCP New Reno fast recovery takes care of the multiple packet drops from one *cwnd*.

However, the shortcoming of New Reno is that because it cannot distinguish the causes of packet loss, more effective fast recovery cannot be performed. In addition, the reduction of sending rate to use the AIMD mechanism is to invoke the dropping of throughput in wireless networks where the multiple packet loss usually happens.

2.3 TCP Westwood

TCP Westwood is a wireless TCP using end-to-end proactive congestion control. TCP Westwood estimates the current network bandwidth at sender side. The sender estimates the network bandwidth by exploiting the rate and pattern of returning ACK through the reverse links. Upon detecting a packet loss, the sender adjusts the *cwnd* according to the estimation. By employing this estimator it exhibits higher performance than other TCP variants such as TCP Reno, New Reno, and SACK [17] in both the wired and wireless networks.

However, TCP Westwood does not distinguish the cause of packet loss. It will adjust the transmission rate constantly, upon experiencing the packet loss. Therefore it decreases the throughput in high BER wireless networks. It is a problem that the accuracy of estimated available bandwidth depends on the network condition which is changed based on the network traffic in links. In addition, it infringes on the other connection's bandwidth when the multiple connections share one link. Hence, the problem of TCP Westwood is how accurately it estimates the rate.

2.4 TCP Jersey

TCP Jersey is employed to improve performance in wireless networks including congestion warning (CW) mechanism that differentiates the packet loss by network congestion between the packet loss by wireless link error and ABE algorithm. When the ACK packet arrives at the sender, the ABE algorithm computes the current available bandwidth based on the time interval of ACK packets. The optimized window (*ownd*) is calculated based on this estimation when the packet loss is detected.

CW is an intermediate router mechanism based on explicit congestion notification (ECN) scheme [18]. Besides the ECN, the CW is set to a threshold with fewer parameter settings at the intermediate router that shall mark all the packets when the average queue length exceeds the threshold. The sender who receives the marked packets decreases its *cwnd*, otherwise it maintains the current *cwnd*. CW does not detect the congestion in the link layer. However, the packet loss caused by the link layer congestion handles the link layer retransmission and the rate control in the TCP sender-side.

2.5 TCP New Jersey

TCP New Jersey improves the ABE algorithm using the TCP Jersey. It adjusts slow start threshold (*ssthreshold*) based on the estimation which is calculated by ABE algorithm. TCP Jersey and New Jersey consist of two key components, the ABE algorithm and the CW mechanism, that helps the sender effectively differentiate the cause of packet loss at intermediate router. TCP New Jersey sender estimates the current available bandwidth based on the packet interarrival time on the receiver. TCP New Jersey is robust to background traffic in reverse links because of timestamp-based ABE.

However, TCP New Jersey experiences the degradation of throughput depending on background traffic pattern such as congestion in the forward link and bi-directional background traffic. And it cannot increase the reduced *cwnd* according to the cause of packet loss effectively. Consequently, when the packet loss occurs consistently because of high BER in wireless networks, it may decrease the throughput due to the reason that is mentioned above.

3. The Proposed Scheme

TCP New Jersey proposed a timestamp-based available bandwidth estimation algorithm. As TCP New Jersey, however, considers the state of one link only, it is not without a flaw that unidirectional background traffic determines the data throughput. TCP New Jersey has another drawback that it is incapable of effective throughput recovery from TCP packet loss. In particular, packet loss caused by a high BER, as opposed to that caused by congestion, requires an effective recovery algorithm. In this respect, TCP NJ+ proposes a mechanism designed to maintain a high data throughput without compromising the fairness of the proposed algorithm and effectively recover the throughput from a reduction due to packet loss

by comparing the available bandwidths of a bidirectional link and securing an available bandwidth in a more aggressive manner. TCP NJ+ consists of three algorithms. MABE is an algorithm designed to measure and compare available bandwidths based on the ACK receiving time and data receiving time and aggressively calculate the available bandwidth. EASR, then, effectively handles the reduced transfer rate depending on the results of MABE when an RTO occurs. Lastly, BERR is capable of effectively compensating for the reduction of the congestion window size when packet loss occurs due to a high BER.

3.1 Maximized Available Bandwidth Estimation (MABE)

The calculation of an available bandwidth appropriate for network conditions is essential to the maintenance of a high transfer rate on the sender side. Therefore, TCP NJ+ improves the existing Timestamp-based available bandwidth estimation (TABE) algorithm, estimates the currently available bandwidth and ensures efficient transmission. In previous studies, TCP Jersey and TCP New Jersey predict the current available bandwidth using Formula (1).

$$R_n = \frac{RTT \times R_{n-1} + L_n}{(t_n - t_{n-1} + RTT)} \tag{1}$$

In TCP Jersey, R_n refers to the available bandwidth as calculated at t_n, when the nth ACK reached the sender, while t_{n-1} refers to the time at which the TCP sender received the $(n-1)^{th}$ ACK. In addition, RTT means the end-to-end round trip time as calculated at t_n, whereas L_n refers to the size of the data packet acknowledged by the nth ACK. R_{n-1} is the available bandwidth as calculated when the $(n-1)^{th}$ ACK is received, and helps prevent momentary changes in the network from resulting in sudden increase or reduction in the available bandwidth. As the basic assumption of the ABE algorithm is that under ideal conditions for the the link receiving an ACK, the pattern of an ACK returning to the sender is identical to the pattern of a data packet reaching the receiver, the available bandwidth from the sender to the receiver may be indirectly estimated. This estimation, however, is incorrect if ACK compression, ACK delays and/or ACK losses occur.

In order to address this drawback, TCP New Jersey uses Formula (1) to measure the available bandwidth, where, in contrast to TCP Jersey, t_n and t_{n-1} refer to the time at which the nth and $(n-1)^{th}$ packets are received, respectively. In other words, timestamp options are used to calculate the difference between the time at which the receiver received the nth data packet and the time at which it received the $(n-1)^{th}$. Accordingly, if there is background traffic in the direction in which the ACK returns, TCP New Jersey ensures more appropriate estimates for available bandwidths. TCP New Jersey, however, still remains incapable of providing higher transfer rate depending on the conditions of a link. TCP NJ+ compares the two methods shown above and selects a higher value, redressing the issue of the bandwidth estimation relying on one link only. It thereby prevents a decline in the transfer rate due to unidirectional background traffic and ensures higher transfer rate.

If the link transmitting data and the link receiving ACKs have different available bandwidths and delays in **Fig. 1**, the data reception interval and ACK reception interval may differ depending on the background traffic present in each link. Effective measurement of bandwidths, therefore, requires consideration of the conditions of the bidirectional link. **Fig. 2** illustrates the procedure for the estimation of the available bandwidth used when the sender in TCP NJ+ receives an ACK containing the timestamp option.

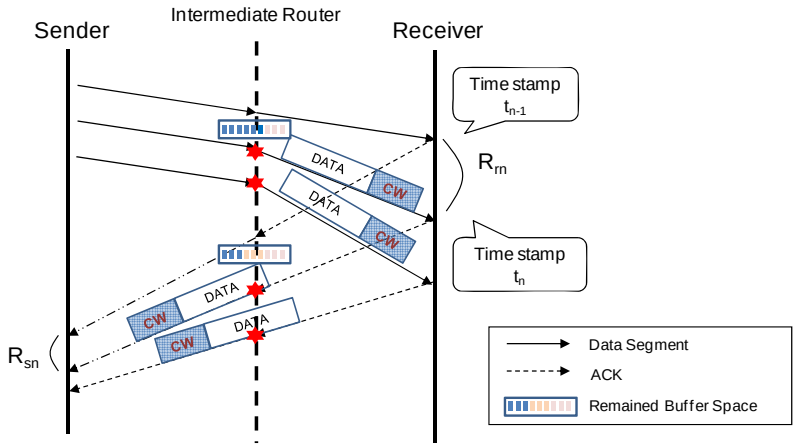


Fig. 1. MABE Algorithm Scenario in the Asymmetric Network

Initialization:

1: $n \leftarrow 1$

2: $R_{s0}, R_{r0}, t_{s0}, t_{r0} \leftarrow 0$

Procedure:

1: /* ACK packet arrived at the sender */

2: **if**(timestamp)

3: $R_{sn} \leftarrow (RTT \times R_{sn-1} + L_n)/((t_{sn} - t_{sn-1}) + RTT)$

4: */* ABE based on ACK packet inter arrival time */*

5: $R_{rn} \leftarrow (RTT \times R_{rn-1} + L_n)/((t_{rn} - t_{rn-1}) + RTT)$

6: */* ABE based on data packet inter arrival time */*

7: $R_n \leftarrow \max(R_{sn}, R_{rn})$

8: */* maximum value of two estimations */*

9: $n \leftarrow n + 1$

10: **end if**

Fig. 2. Maximized Available Bandwidth Estimation Algorithm

RTT refers to the end-to-end round trip time as measured at t_{sn}, while L_n means the length of the data packet referenced by the nth ACK. R_{sn}, therefore, refers to the available bandwidth as measured based on the ACK receiving rate. In addition, R_{rn} is the available bandwidth as measured when the receiver received nth ACK at t_{rn}, where t_{rn-1} refers to the time at which it received the $(n-1)^{th}$ ACK. RTT refers to the end-to-end round trip time as measured at t_{rn}, while L_n means the length of the nth data packet. Put another way, R_{rn} is the available bandwidth resulting from the difference in time between the receiving of two data packets. TCP NJ+ then compares the two measured values, R_{sn} and R_{rn}, to use the higher value as the available bandwidth. It therefore prevents the size of the available bandwidth from decreasing due to the conditions of the link transmitting data packets only or the link transmitting ACKs only. It also ensures faster transfer rate recovery by acquiring the maximum value of the bandwidth when an RTO occurs. TCP NJ+ may address the issues with the bandwidth estimation of TCP New Jersey relying on the background traffic of one link by measuring and comparing the bandwidths of both links.

3.2 Effective Adjustment for Retransmission Timeout

The second algorithm used in TCP NJ+, EASR, calculates the optimal congestion window (*ownd*) size based on MABE and uses it according to the cause of an RTO. EASR is designed to build on the algorithms used for TCP Jersey and TCP New Jersey and ensures a higher transfer rate. The TCP flow control revolves around the additive increase /multiplicative-decrease (AIMD) algorithm. That is, a TCP sender raises the data transfer rate by gradually increasing the size of the congestion window. If packet loss occurs, however, the RTO mechanism cuts the size of the congestion window down to 1, greatly reducing the data transfer rate. This mechanism, however, is problematic in that it misinterprets an RTO caused by a high BER as congestion and then reduces the transfer rate without reservation. In a wireless network where a high BER is common, in particular, this problem is widely known to cause TCP to experience performance degradation. This in turn calls for an algorithm to utilize an optimum congestion window value and maintain the transfer rate depending on the cause of the RTO.

$$ownd_n = \frac{RTT \times R_n}{seg\_size} \tag{2}$$

TCP Jersey and TCP New Jersey use Formula (2) to calculate the optimum congestion window size, $ownd_n$. In this formula, *seg_size* refers to the size of the TCP packet, R_n refers to the bandwidth as estimated by ABE or TABE, and *RTT* means the end-to-end round trip time. If an RTO occurs with the transmission of the *n*th packet, TCP Jersey and TCP New Jersey determine if the cause of the RTO is network congestion or a high BER. The CW algorithm is used to determine the cause of the RTO occurrence. The CW algorithm is based on ECN and has seen wide use since it is introduced with the Random Early Detection (RED) [20] algorithm in the 1990's [12]. The determination of the cause of the RTO based on the CW algorithm, therefore, does not lose its persuasiveness even in consideration of the actual network environment. After determining the cause of the RTO, the TCP Jersey or TCP New Jersey sender reduces the value of the congestion window size to 1 if the RTO is caused by network congestion, or changes the size to the optimum size, $ownd_n$, if it is caused by a high BER.

```
if (RTO expired)
    if (Congestion Warning)
    /* if RTO due to congestion */
        cwnd = 1;
        ssthresh = ownd_n;
    else
    /* if RTO due to BER */
        cwnd = ownd_n ;
        ssthresh = ownd_n;
    end if
end if
```

(a)TCP NJ

```
if (RTO expired)
    if (Congestion Warning)
    /* if RTO due to congestion */
        cwnd = 1;
        ssthresh = ownd_n;
    else
    /* if RTO due to BER */
        cwnd = (ownd_n + ownd_n-1) / 2;
        ssthresh = ownd_n;
    end if
end if
```

(b)TCP NJ+ EASR

Fig. 3. Comparison between TCP NJ and TCP NJ+

The optimum size of the congestion window as calculated here, however, is based on the available bandwidth as calculated when the state of the network link temporarily deteriorated. This is why the $ownd_n$ value at the time of the high BER does not actually mean the optimum congestion window value. In other words, the RTO occurrence caused by a high BER differs from that caused by congestion and does not indicate the congestion of the link, it means that the bandwidth of the link is sufficiently available. The actual available bandwidth is higher than the measured value. This therefore calls for an algorithm for the calculation of the optimum congestion window size that takes into account the conditions existing prior to the high BER.

For this reason, TCP NJ+ considers $ownd_{n-1}$, the value effective immediately prior to the RTO occurrence, and thereby ensures a higher transfer rate. **Fig. 3** compares how TCP New Jersey and TCP NJ+ work in pseudo code. If RTO occurs, the TCP NJ+ sender determines the causse of the RTO. If the RTO is caused by network congestion, the sender reduces *cwnd* to 1, as with the case with TCP New Jersey, and uses the $ownd_n$ value to set *ssthresh*. If, however, the RTO is caused by a high BER, TCP NJ+ uses the $ownd_n$ value to set *ssthresh* and the mean between $ownd_n$ and $ownd_{n-1}$ to set *cwnd*. The $ownd_{n-1}$ value is measured before the network event where the current link error causes the RTO, and therefore is higher than $ownd_n$. In other words, the mean between $ownd_n$ and $ownd_{n-1}$ is obtained to prevent a sudden drop in *ownd* due to a link error. The use of a higher *cwnd* value than with TCP New Jersey ensures faster recovery from the RTO caused by a high BER.

3.3 Handling BER Error Recovery

BERR provides a mechanism to set the appropriate congestion window size and *ssthresh* value depending on the cause of a 3-DUPACKs if it occurs. TCP New Jersey provides the following recovery algorithm when the sender receives a 3-DUPACKs. The sender first determines the cause of the packet loss, and if the cause is congestion, it reduces *ssthresh* to $ownd_n$ and compares *cwnd* with *ssthresh* to adjust it. That is, the current *cwnd* value is maintained if *cwnd* is lower than *ssthresh*, or the $ownd_n$ value is used to *cwnd* if it is higher than *ssthresh*. On the other hand, the sender maintains the current *cwnd* and *ssthresh* values if the packet is lost by a high BER.

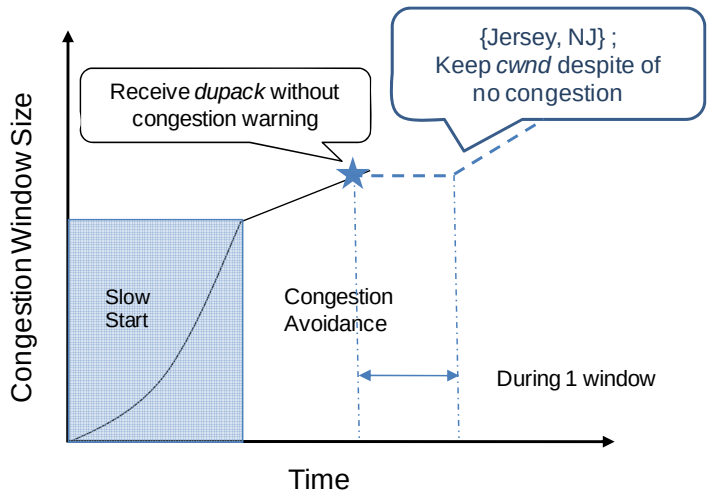


Fig. 4. Problem When the Sender Receive dupack without congestion warning

As the packet loss is caused by a high BER, not by congestion, however, the buffer of the router in the middle has space to store packets. As seen in **Fig. 4**, the failure to increase *cwnd* under no congestion, therefore, means a relative reduction in *cwnd*. TCP NJ+ uses the BERR algorithm to resolve this issue with TCP New Jersey. When TCP NJ+ sender has received a 3-DUPACKs, the BERR algorithm works as follows: First, TCP NJ+ works as the same way as TCP New Jersey does if the packet loss is caused by congestion. Second, if a high BER caused the packet loss, *ssthresh* is set to $ownd_n$ and *cwnd* is incremented by the MSS value. A 3-DUPACKs occurred due to a high BER in TCP New Jersey prevents *cwnd* from increasing any more despite no congestion has occurred. TCP NJ+, therefore, increases *cwnd* by 1 maximum segment size (MSS) in order to compensate for the increase in *cwnd* that have not been attained due to the high BER. This *cwnd* compensation algorithm ensures high transfer rate by fully utilizing the available bandwidth that has not been utilized due to the failure to expand *cwnd*. In a high BER network environment where packet loss occurs more often, in particular, TCP NJ+ obtains relatively higher *cwnd* values and ensures better performance than other wireless TCP approaches. Although the expansion of *cwnd* may seem more or less aggressive, the buffer space at the router does not run short as the packet loss is caused by a high BER. In addition, even if the receiver buffer is full of out-of-sequence packets, buffer overflow due to the expanded *cwnd* may be prevented as the buffer of the TCP receiver is flow-controlled by means of advertised window (*awnd*). The buffer of the router is also controlled by the CW mechanism. The downside effects of expanding *cwnd*, therefore, is countered by the above mechanisms.

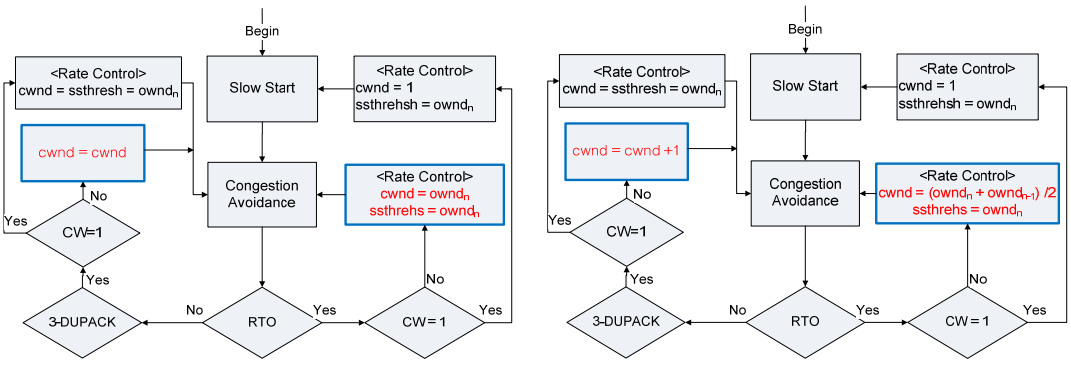


Fig. 5. Operation Flow Chart of both NJ and NJ+

TCP NJ+ proposes MABE, EASR and BERR. **Fig. 5** compares in flow charts how TCP New Jersey and TCP NJ+ work. The three proposed algorithms allow TCP NJ+ to have higher data throughputs than TCP New Jersey as well as TCP proper. The performance gains of TCP NJ+ are obtained by compensating for the relatively low measured value of the congestion window if packet loss is caused by a high BER. This advantage of the congestion window compensation algorithm may ensure higher transfer rate than other TCP schemes. It also enhances the occupancy of the allocated bandwidth and makes effective use of it so that network resources are not wasted.

4. Performance Evaluation

We evaluate the performance of TCP NJ+ by showing the metrics, such as goodput, fairness,

and friendliness using the ***ns-2*** simulator [23]. We experiment the proposed scheme in two network topologies, which are either simple or more realistic with various simulation parameters including the link bandwidth, the packet size, the propagation delay, and the queue size as shown in **Table 1**, that is referred to [21].

Table 1. Simulation parameters

Bandwidth	Packet Size	Propagation Delay	Queue Size
Wired : 100MB	726byte	Wired : 10~20ms	20~200 packets
Wireless : 2MB		Wireless : 1ms	

4.1 Goodput Performance in Simple Wireless Links

Goodput is the effective amount of data delivered through the network. It is a direct direction indicator of network performance. We evaluate the goodput of previous studies including TCP NJ+, TCP New Jersey, TCP Westwood, and TCP Reno on various wireless link error rates: 0.1%, 1%, 2%, 3%, 4%, and 5% in the topology of **Fig. 6.**

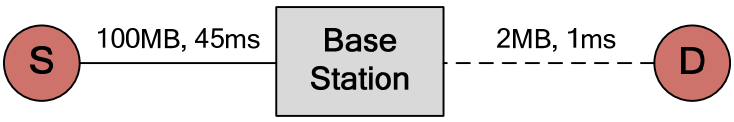


Fig. 6. Simulation topology

The source (node S) is connected to the node BS via a 100 MB wired link with 45 *ms* propagation delay. The node BS is linked to the destination (node D) via a 2 MB wireless link with 1 ms propagation delay. The queue size of the wired link is set to 150 and the wireless link queue size is set to 20. The goodput result is shown in **Fig. 7**.

Especially, in 5% wireless link error rate, TCP NJ+ highly outperforms TCP New Jersey by 19% and TCP Westwood by 54%. Because the BERR and EASR are performed when the packet loss is caused by BER, the result shows that the higher wireless link error rate, the higher the goodput of TCP NJ+.

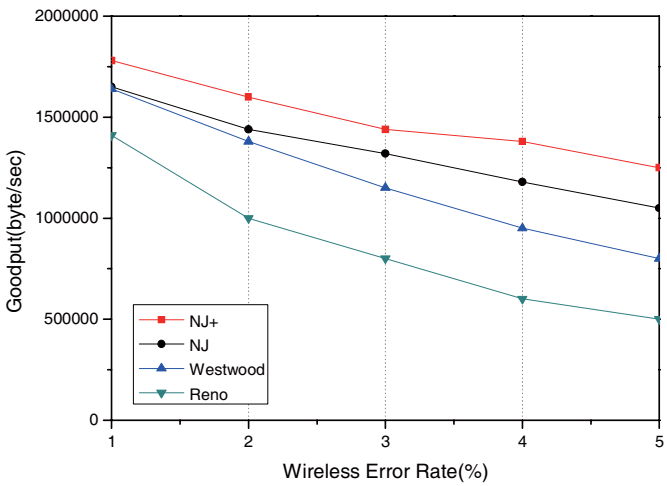


Fig. 7. Goodput vs. wireless link error rate

4.2 Goodput Performance with Bi-directional Background Traffic

In TCP NJ+, the ABE algorithm ensures high throughput regardless of the background traffic pattern. And it achieves high performance in the congestive state of wireless links as we see it in simulation results. As illustrated in **Fig. 8**, we simulate the goodput of TCP NJ+, TCP New Jersey, Westwood, and Reno on various wireless link error rates (1%, 2%, 3%, 4%, and 5%) under three environments. One is forward link where data segments are transmitted has traffic. Another is reverse link where ACKs are traversed has traffic. The other is bi-directional background traffic which leads to congestion (congestive state), or not (non-congestive state). We trace arrival, departure, and dropped packets on wireless link (from node AP to node D) which has 1% wireless link error rate.

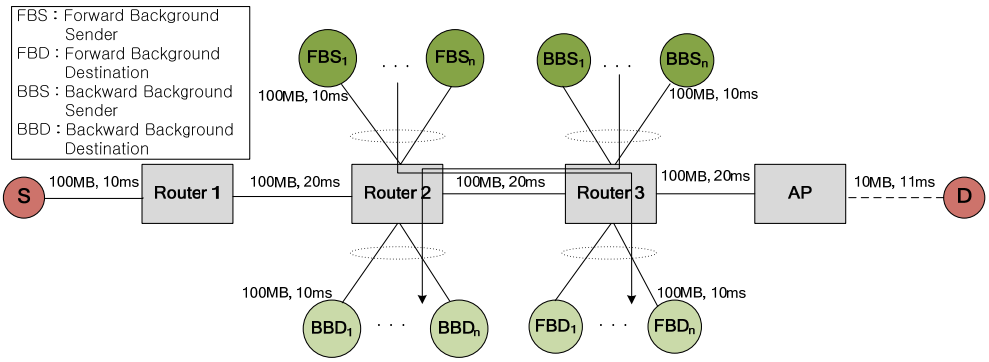
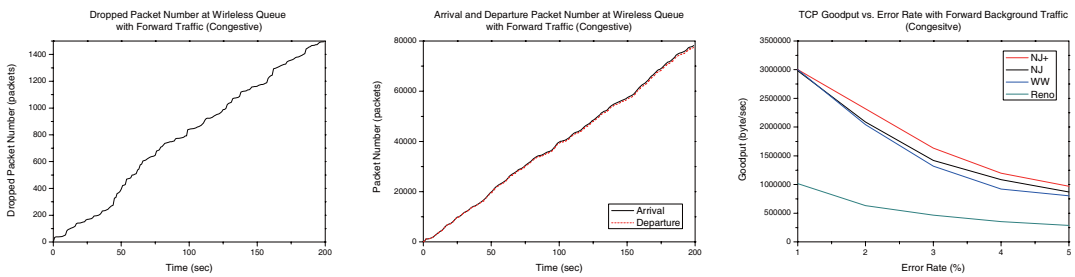


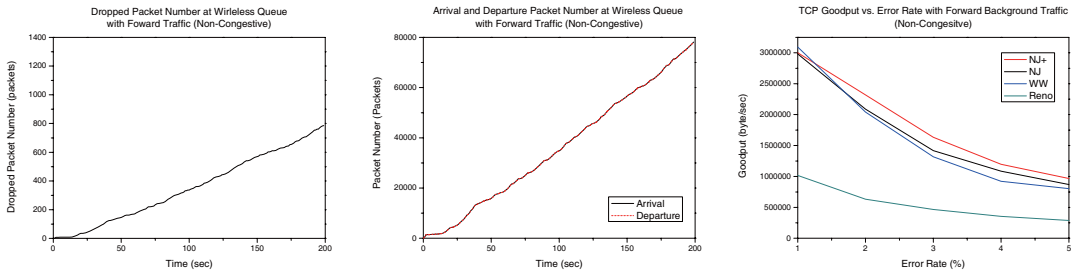
Fig. 8. Simulation topology via background traffic direction

The source (node S) is connected to Router 1 via a 100 MB wired link with 10 *ms* propagation delay. Router 1 is linked to Router 2 and Router 2 is linked to Router 3 via a 100 MB wired link with 20 *ms* propagation delay. Router 3 is linked to AP via a 100 MB wired link with 20 *ms* propagation delay. AP is connected to destination (node D) via a 10 MB wireless link with 10 *ms*. The background traffic flows, from node FBS_n to node FBD_n (forward) and from node BBS_n to node BBD_n(reverse), and in bi-direction are FTP background traffic via a 100 MB wired link with 10 *ms* delay. The number of background traffic nodes is changed from 5 (non-congestive state) to 10 (congestive state). The queue size of the wired link is set to 200 and the wireless link queue size is set to 20. We show the results of goodput and queue state for congestion status in the wireless queue.



(a) Dropped packet number in wireless queue (b) Arrival and departure packet number in wireless queue (c) Goodput vs. wireless link error rate with forward background traffic

Fig. 9. Simulation with forward background traffic (Congestive state)

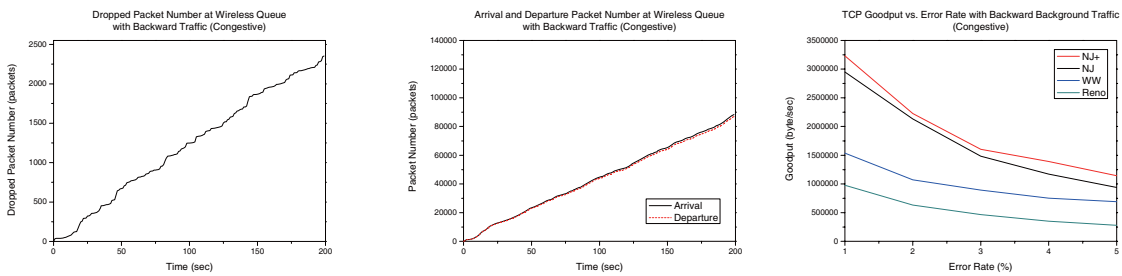


(a) Dropped packet number in wireless queue (b) Arrival and departure packet number in wireless queue (c) Goodput vs. wireless link error rate with forward background traffic

Fig. 10. Simulation with forward background traffic (Non congestive state)

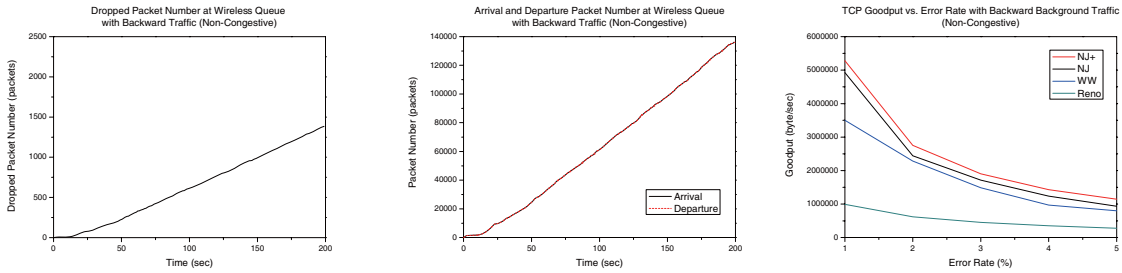
The simulation results consist of forward, reverse, and bi-directional background traffic cases with congestive and non-congestive states including goodput, dropped packets, arrival and departure packets in the wireless queue. Results of the FTP forward background traffic are illustrated in **Fig. 9** and **Fig. 10**. **Fig. 9** represents the dropped packet number in the wireless queue, the arrival and departure packet number in wireless queue, and goodput result with forward background traffic, respectively, that are the results of the wireless queue for congestive state. **Fig. 10** illustrates the same for non-congestive state.

Fig. 9(b) shows that the wireless link suffers from network congestion because the number of departure packet is less than the number of arrival packet. So, the result describes that departure packet line is located under the arrival packet line. As shown in **Fig. 9(a)**, the dropped packet number is larger than the one shown in **Fig. 10(a)** of non-congestive state. Therefore, we evaluate the proposed scheme in the wireless links with congestive and non-congestive conditions. **Fig. 9(c)** and **Fig. 10(c)** presents that the goodput of TCP NJ+ in forward link congestive and non-congestive state is the higher than any other TCP variant in the same network condition. Especially, TCP NJ+ outperforms New Jersey by 24% and Westwood by 75% in 5% wireless link error rate with forward background traffic for congestive or non-congestive states. TCP NJ+ shows higher performance than any other TCP scheme for the wireless links in congestive or non-congestive state. The reason why TCP NJ+ shows high performance is that TCP NJ+ achieves *cwnd* by MABE regardless of the background traffic pattern because it estimates the optimal (maximum) available bandwidth. If RTO due to BER occurs, the EASR mechanism inflates by reducing the *cwnd* more quickly than other TCP schemes.



(a) Dropped packet number in wireless queue (b) Arrival and departure packet number in wireless queue (c) Goodput vs. wireless link error rate with backward background traffic

Fig. 11. Simulation with reverse background traffic (Congestive state)



(a) Dropped packet number in wireless queue

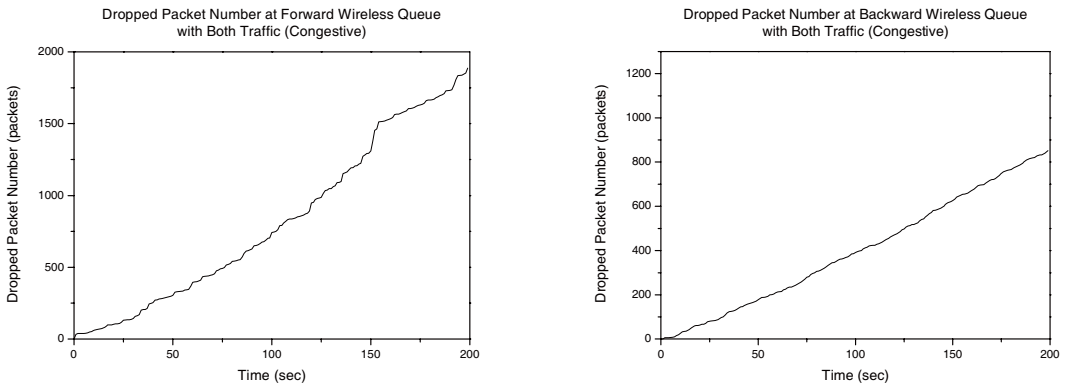
(b) Arrival and departure packet number in wireless queue

(c) Goodput vs. wireless link error rate with backward background traffic

Fig. 12. Simulation with reverse background traffic (Non congestive state)

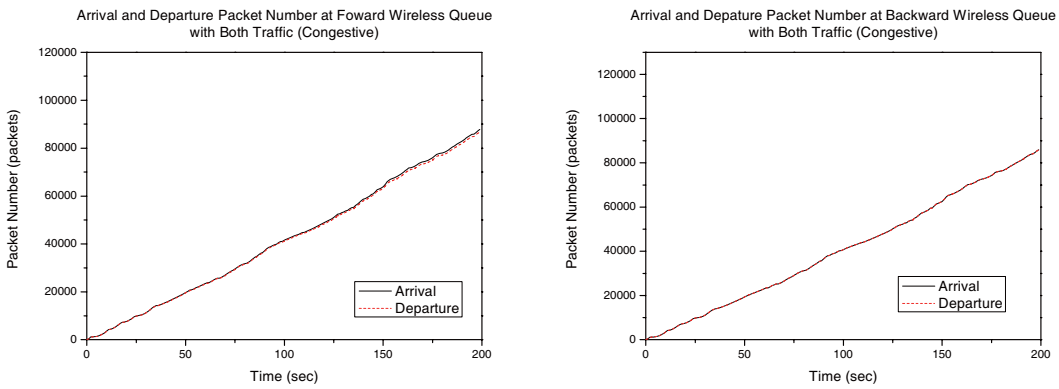
Fig. 11 and **Fig. 12** represent the simulation results with reverse background traffic. **Fig. 11** describes the dropped packet number in the wireless queue, the arrival and departure packet numbers in wireless queue, and goodput result with reverse background traffic, respectively, for the wireless queue in congestive state. **Fig. 11** illustrates for the non-congestive state.

Although the forward link is free, the performance of all TCP variants is decreased because the ACKs are not traversed back to the sender due to the reverse link congestion. In **Fig. 11 (a, b)** and **Fig. 12(a, b)**, we simulate the same environments (congestive and non-congestive states) as in forward background traffic situation.



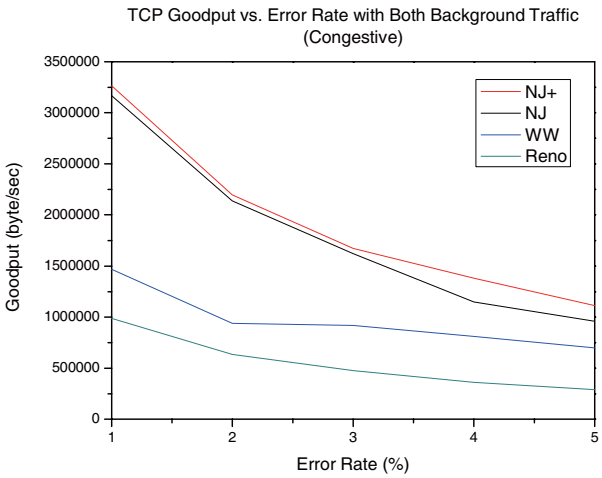
(a) Dropped packet number in forward wireless queue

(b) Dropped packet number in reverse wireless queue



(c) Arrival and departure packet number in forward wireless queue

(d) Arrival and departure packet number in backward wireless queue

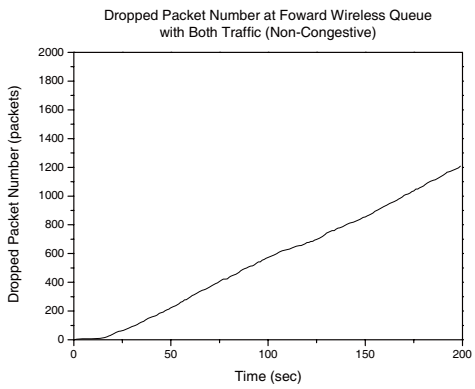


(e) Goodput vs. wireless link error rate with bi-directional background traffic

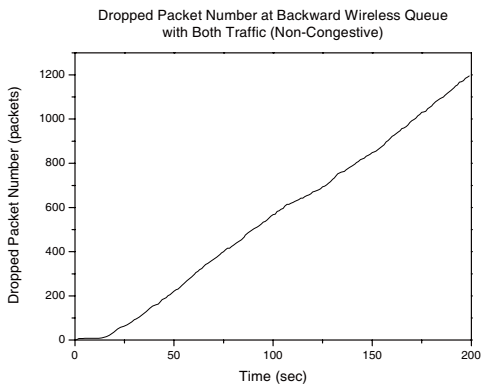
Fig. 13. Simulation with bi-direction background traffic (Congestive state)

The goodput result of the reverse background traffic in congestive state is illustrated in **Fig. 11(c)**. TCP NJ+ has 21% and 65% improvement over TCP New Jersey and Westwood, respectively in 5% wireless link error rate with reverse background traffic. In **Fig. 11(b)** and **Fig. 12(b)**, the simulation results are plotted in the congestive state. **Fig. 12(c)** describes the simulation result of the reverse background traffic in non-congestive state. TCP NJ+ shows higher performance.

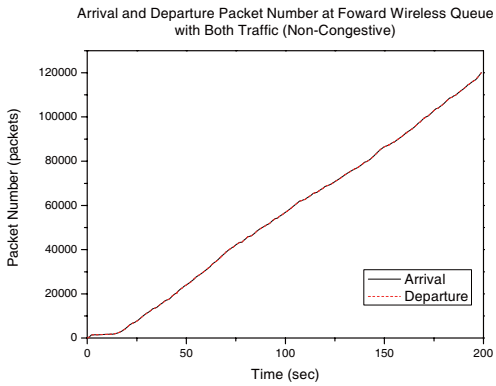
The simulation results with bi-directional background traffic are illustrated in **Fig. 13** and **Fig. 14**. We describe the forward wireless queue state in **Fig. 13(a, c)** and the reverse wireless queue state (**Fig. 13(b, d)**) in congestive state. **Fig. 13(e)** shows the goodput result of TCP NJ+ in congestive state. We represent the forward wireless queue state (**Fig. 14(a, c)**) and the reverse wireless queue state (**Fig. 14(b, d)**) in non-congestive state. **Fig. 14(e)** illustrates the goodput result of TCP NJ+ in non-congestive state. In **Fig. 13(e)** and **Fig. 14(e)**, TCP NJ+ outperforms TCP New Jersey by 16% and Westwood by 280% in 5% wireless link error rate with bi-directional background traffic. The simulation results show that TCP NJ+ achieves higher goodput than New Jersey and Westwood regardless of the background traffic patterns (forward, reverse, and bi-directional) for the wireless links in congestive or non-congestive state. Especially, it is robust to congestive state.



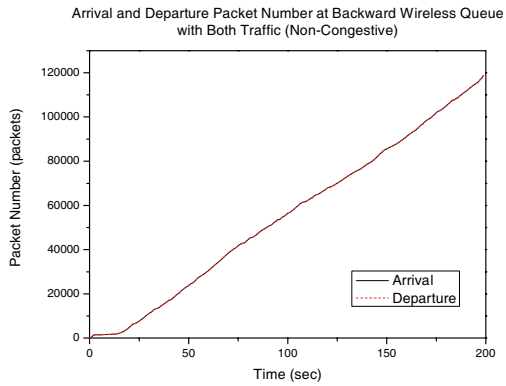
(a) Dropped packet number in forward wireless queue



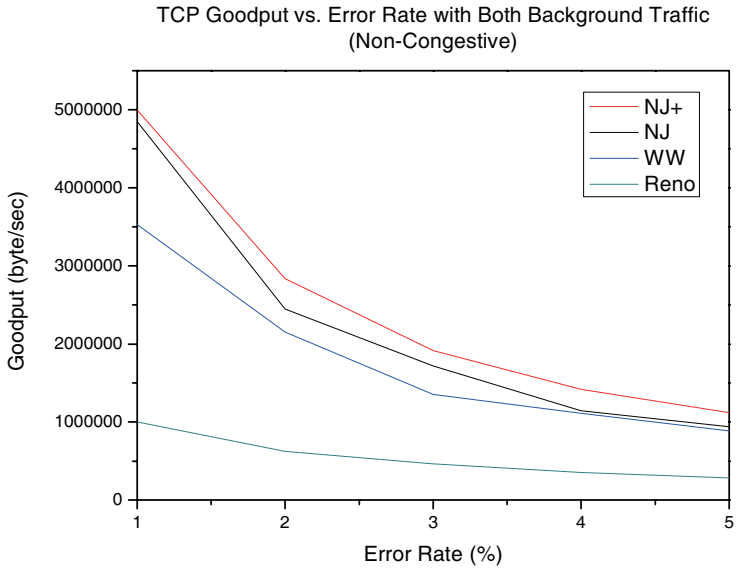
(b) Dropped packet number in reverse wireless queue



(c) Arrival and departure packet number in forward wireless queue



(d) Arrival and departure packet number in backward wireless queue



(e) Goodput vs. wireless link error rate with bi-directional background traffic

Fig. 14. Simulation with bi-direction background traffic (Non Congestive state)

4.3 Fairness Evaluation

Fairness is also an essential measure of TCP performance evaluation. It is the bandwidth allocation measure for the multiple connections of the same TCP. We use the Jain's fairness index proposed in [22] in order to show the fairness of TCP NJ+, New Jersey, Westwood, and Reno on various link error rates using the topology in **Fig. 15**.

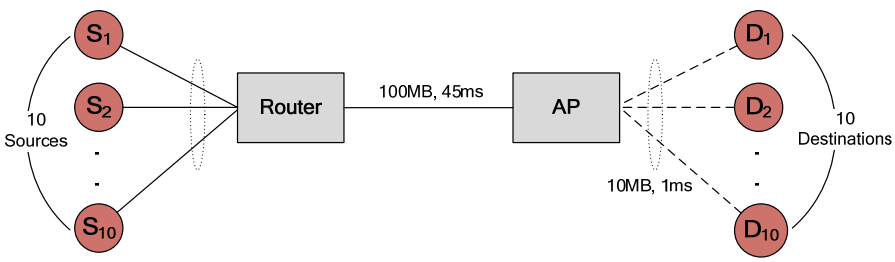


Fig. 15. Fairness simulation topology

Given 10 sources are transferred to 10 destinations by using same TCP scheme. The Jain's fairness index function is expressed in Formula (3)

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)} \tag{3}$$

Where *F(x)* is the fairness index of *x* flow. x_i is the throughput of *i*-th flow, and *n* is the number of connections. *F(x)* ranges from *1/n* to 1.0. A perfectly fair bandwidth allocation results in a fairness index *F(x)* of 1.0. The fairness evaluation results are summarized in **Table 2.** In conclusion, TCP NJ+ satisfies good fairness like the other TCP variants.

Table 2. Fairness of TCP schemes vs. link error rate

Error Rate(%)	NJ+	NJ	Westwood	Reno
0.0	0.9999	0.9999	1.0000	1.0000
0.1	0.9999	0.9999	0.9999	0.9998
0.5	0.9999	0.9999	0.9999	0.9986
1.0	0.9999	0.9999	0.9998	0.9989
5.0	0.9994	0.9994	0.9964	0.9980
10	0.9903	0.9904	0.9811	0.9875

4.4 Friendliness Evaluation

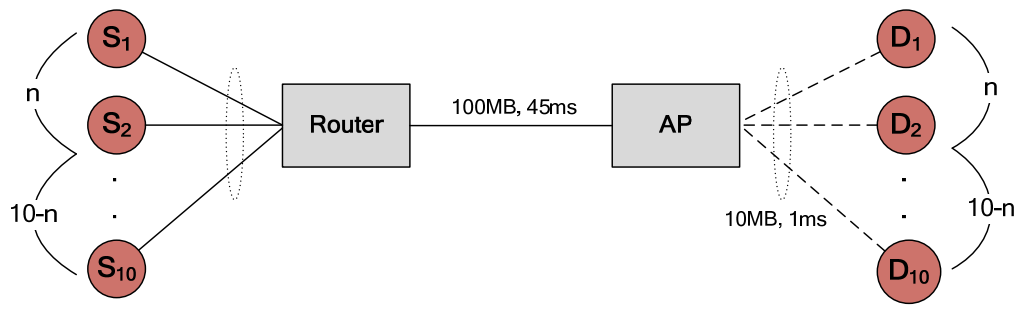


Fig. 16. Friendliness simulation topology

Friendliness is a metric to measure whether TCP schemes are able to coexist with other TCP variants and does not cause them starvation. In low-bandwidth network with many TCP connections, TCP NJ+ may affect the throughput degradation of other TCP connections. Consequently, it causes TCP global synchronization. TCP NJ+ compensates *cwnd* when the packet loss is detected, however, it does not starve other TCPs in the same network as you see the friendliness result. To verify the friendliness of TCP NJ+, we construct the simulation topology, where TCP NJ+ coexists with TCP Reno. The topology is presented in **Fig. 16**. Here the wireless link error rate is set to 1%. There are 10 pairs of connections sharing the 1MB per connection ideally. During the simulation, the number of TCP Reno connections is changed from 0 to 10 and the corresponding number of TCP NJ+ flows is changed from 10 to 0. The results of the friendliness are shown in **Fig. 17**. According to the simulation results, TCP Reno

uses the bandwidth for 0.5MB per each connection because of the degradation of the throughput in wireless networks. TCP NJ+ achieves average bandwidth for 1.1MB per each connection because it is designed to perform better in wireless networks. Hence, the friendliness of TCP NJ+ is satisfied and it cannot reduce the throughput of other TCP connections in the same network.

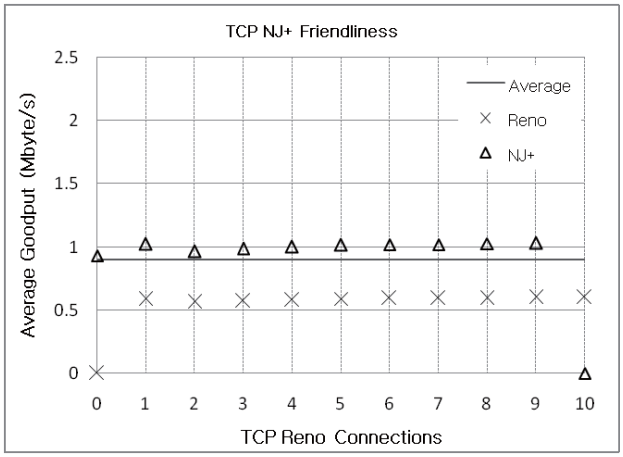


Fig. 17. Results of friendliness

5. Conclusion

We have proposed TCP NJ+ which enhances the performance of TCP New Jersey. Three effective mechanisms are proposed in TCP NJ+. First, the maximized ABE algorithm guarantees significant throughput regardless of the background traffic pattern because it estimates the optimal (maximum) available bandwidth. Second, when the packet loss caused by BER occurs, the BERR mechanism makes the reduced *cwnd* to be increased speedily. Third, if *RTO* caused by BER occurs, the EASR mechanism inflates the *cwnd* more quickly than other TCP schemes.

Simulation results demonstrate that TCP NJ+ improves the performance even when wireless link error rates increase. Particularly, TCP NJ+ outperforms New Jersey by 19% and Westwood by 54% in 5% wireless link error rate with no cross-traffic. Under a 5% wireless link error rate with background traffic, TCP NJ+ achieves 27% and 52% enhancement over TCP New Jersey and Westwood, respectively. In addition, the fairness and friendliness are also satisfied and TCP NJ+ does not starve other TCP variants, although it is more aggressive than any other TCP variants. In conclusion, TCP NJ+ with the maximized ABE, handling BER error recovery, and effective adjustment of sending rate for *RTO* mechanisms is robust to high BER environments, shows significant performance improvements.

References

[1] J. Postel, “Transmission Control Protocol,” *RFC 793*, September 1981.
[2] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control,” *RFC2581*, April 1999.
[3] F. Lefevre and G. Vivier, “Understanding TCP's Behavior over Wireless Links,” *Symposium on Communications Vehicular Technology 2000*, pp.123-1302, November 2000.
[4] W. Liao, C.J. Kao, and C.H. Chien, “Improving TCP Performance in Mobile Networks,” *IEEE Transactions on Communications*, Vol.53, No.4, pp.569-571, April 2005.

[5] C.Y. Ho, Y.C. Chan, and Y.C. Chen, An Effcient Mechanism of TCP-Vegas on Mobile IP Networks," *IEEE INFOCOM*, Vol.4, pp.2776-2780, March 2005.
[6] X. Wu, M.C. Chan, and A.L Ananda, "TCP HandOff: A Practical TCP Enhancement for Heterogeneous Mobile Environments," *IEEE ICC*, pp.6043-6048, June 2007.
[7] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, Vol.5, No.3, pp.336-350, June 1997
[8] H. Elaarag, "Improving TCP Performance over Mobile Networks," *ACM Computing Surveys*, Vol.34, Issue.3, pp.357-374, September 2002.
[9] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Prob-lems and Solutions," *IEEE Radio Communications*, Vol.43, Issue.3, pp.27-32, March 2005.
[10] S. Floyd and T. Henderson, "The New Reno Modification to TCP's Fast Recovery Algorithm," *RFC 2582*, April 1999.
[11] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *ACM/IEEE Mobile Computing and Networking*, pp.287-297, July 2001.
[12] K. Xu, Y.Tian, and N. Ansari, TCP-Jersey for Wireless IP Communications," *IEEE Journal of Selected Areas in Communications*, Vol.22, Issue.4, pp.747-756, May 2004.
[13] K. Xu, Y. Tian, and N. Ansari, "Improving TCP Performance in Integrated Wireless Communications Networks," *Computer Networks*, Vol.47, pp.219-237, February 2005.
[14] V. Jacobson, "Congestion Avoidance and Contol," *ACM SIGCOMM*, Vol.18, pp.314-329, August 1988.
[15] UCB/LBNL/VINT Network Simulator [Online]Avilable: http://www.isi.edu/nsnam/ns
[16] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms," *RFC 2001*, January 1997.
[17] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to The Selective Acknowledgement Option for TCP," *RFC 2883*, July 2000.
[18] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communications Review*, Vol.24, No.5, pp.10-23, October 1994.
[19] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," *RFC 1323*, May 1992.
[20] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, Vol.1 No.4, pp. 397-413, August 1993.
[21] C. Song, P.C. Cosman, and G.M. Voelker, "End-to-End Differentiation of Congestion and Wireless Losses," *IEEE/ACM Transactions on Networking*, Vol.11, pp.703-717, October 2003.
[22] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," *DEC Research Report TR-301*, September 1984.

Jaehyung Lee has received BS in computer education from Sungkyunkwan University, Korea in 2007. Since 2007, he has been a research student from Networking Laboratory at Graduate School of Information and Communication Engineering from Sungkyunkwan University, Korea. Moreover, he is a research member with HCI software platform development from the Intelligent HCI Convergence Research Center (8-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program. His research interests include congestion control algorithm and mobile wireless networks.

Jungrae Kim has received BS in computer science from Inchon University, Korea in 2005, MS in Information and Communication Engineering from Sungkyunkwan University, Korea. He was a research member with HCI software platform development from the Intelligent HCI Convergence Research Center (8-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program. Since 2008, he has been a system engineer at Xener Systems. His research interests include congestion control algorithm and mobile wireless networks.

Minu Park received a BS degree in computer science from Sungkyunkwan University, Korea in 2007. Since 2007, he has joined the Telecommunication Network Business of Samsung Electronics, and is also a research student from Networking Laboratory at Graduate School of Information and Communication Engineering from Sungkyunkwan University, Korea. Moreover, he is a research member with HCI software platform development from the Intelligent HCI Convergence Research Center (8-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program. His research interests include service discovery, congestion control algorithm and mobile wireless networks.

Jahwan Koo Jahwan Koo received the B.S., M.S. and Ph.D. degrees in Computer Communication and Networking from Sungkyunkwan University (SKKU), South Korea, in 1995, 1997, and 2006, respectively. For more than five years (from 1997 to 2002), he was a system engineer and infrastructure architect at Korea Information Systems and LG CNS Co., Ltd. (initially LG-EDS), South Korea. For one and half years, he joined at the School of Information and Communication Engineering, SKKU, as a Research Professor. He authored Network Principles and Practices (Life & Books Press, 2004), Understanding Information and Communication Technology (Life & Books Press, 2005) and Internet QoS Differentiation (VDM Verlag Dr. Mueller e.K., 2008). He is a member of ACM, IEEE, IAENG, KSII, etc. He is currently working as a Postdoctoral Fellow at Computer Sciences Department, University of Winsconsin - Madison, USA. His research interests include Internet Quality of Service, Network Management, Network Security, and Information Technology Architecture.

Hyunseung Choo received a BS in mathematics from Sungkyunkwan University, Korea in 1988. He received an MS in computer science from the University of Texas at Dallas, USA in 1990, and a PhD in computer science from the University of Texas at Arlington, USA in 1996. From 1997 to 1998, he was a patent examiner at Korean Industrial Property Office. Since 1998, he has been with the School of Information and Communication Engineering at Sungkyunkwan University, and he is an associate professor and a director of Convergence Research Institute of the university. Since 2005, Dr. Choo is Director of the Intelligent HCI Convergence Research Center (8-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment. His research interests include wired/wireless/optical embedded networking, mobile computing, and grid computing. Dr. Choo has been Editor-in-Chief of the Journal of Korean Society for Internet Information(KSII) for 3 years and journal editor of the “Journal of Communications and Networks,” “ACM Transactions on Internet Technology,” “International Journal of Mobile Communication,” and “Springer-Verlag Transactions on Computational Science Journal” since 2006. He has published over 200 papers in international journals and refereed conferences. Dr. Choo is a member of IEEE, ACM, and IEICE.