TCP NJ+: Packet Loss Differentiated Transmission Mechanism Robust to High BER Environments

Jungrae Kim[1], Jahwan Koo[2], and Hyunseung Choo[1] *

School of Information and Communication Engineering,
Sungkyunkwan University, Suwon, Korea
{witjung, choo}@ece.skku.ac.kr
Intelligent HCI Convergence Research Center
Sungkyunkwan University
440-746, Suwon, Korea
jhkoo@songgang.skku.ac.kr

Abstract. Transmission mechanisms that include an available bandwidth estimation algorithm and a packet loss differentiation scheme, in general, exhibit higher TCP performance in wireless networks. TCP New Jersey, known as the best existing scheme in terms of goodput, improves wireless TCP performance using the available bandwidth estimation at the sender and the congestion warning at intermediate routers. Although TCP New Jersey achieves 17% and 85% improvements in goodput over TCP Westwood and TCP Reno, respectively, we further improve TCP New Jersey by exploring improved available bandwidth estimation, retransmission timeout, and recovery mechanisms. Hence, we propose TCP New Jersey PLUS (shortly TCP NJ+), showing that under 5% packet loss rate, a characteristic of high bit-error-rate wireless network, it outperforms other TCP variants by 19% to 104% in terms of goodput even when the network is in bi-directional congestion.

1 Introduction

Transmission control protocol (TCP) designed for wired networks is a connection-oriented transport protocol that provides reliable data communications [1], [2]. However, wireless infrastructures such as cellular networks, wireless LANs, and mobile computing have such characteristics as high bit-error-rate (BER), limited bandwidth, fading, and handoff, which severe performance degradation [3]. The main reason is that the congestion control mechanism in TCP cannot distinguish between the packet loss caused by wireless link error and that caused by network congestion, thus, reacting to the loss by reducing its congestion window (*cwnd*). Therefore, these inappropriate reductions of the *cwnd* lead to unnecessary throughput degradation for TCP applications [4].

* Corresponding author

Over the last decade, a considerable number of studies have been conducted on improving wireless TCP performance with the advances in wireless infrastructure technologies. According to the operations of TCP connection, wireless TCP algorithms can be divided into split or end-to-end approach [5]. Split approach attempts to prevent the wireless portion from the traditional network by separating TCP connection at the intermediate router (or a base station). The intermediate router behaves as a terminal (or a proxy) in both the wired and wireless portions. Both end hosts communicate with the intermediate router independently without knowledge of the other end. The drawbacks of split approach will be described in [5]. On the other hand, the end-to-end approach, such as TCP New Reno [6], Westwood [7], Jersey [8], and New Jersey [9], treats 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 congestion control.

TCP Westwood modifies addictive increase multiplicative decrease (AIMD) mechanism [2] which is the common congestion window control strategy of the wired TCP and intends to improve TCP performance by effectively adjusting its transmission rates on the basis of the available bandwidth estimation (ABE) algorithm at the sender. On the other hand, TCP Jersey and New Jersey are based on the integration of the sender side ABE algorithm and the receiver side packet loss differentiation scheme, thus, resulting in higher throughput and goodput than any other TCP variants.

Even though TCP New Jersey achieves 17% and 85% improvements in goodput over TCP Westwood and TCP Reno, respectively, we further improve TCP New Jersey's performance by exploring improved available bandwidth estimation, retransmission timeout, and recovery mechanisms. Hence, we propose TCP New Jersey PLUS (shortly TCP NJ+), showing that under 5% packet loss rate, a characteristic of high BER wireless network, 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.

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 in detail the improved mechanisms of TCP NJ+. Section 4 presents performance evaluation via *NS-2* [11] network simulator under various network conditions. The final section offers some concluding remarks.

2 Related Works

2.1 TCP New Reno

TCP New Reno improves standard TCP Reno's fast recovery [2]. The fast recovery is performed when packet loss is detected by the sender and then it enters the congestion avoidance phase after performing fast retransmit [2]. The multiple packet losses force the TCP Reno to invoke a slow down in the recovery of the dropped transmission rate. In TCP New Reno, the fast recovery does not terminate until receipt of full ACK. If the sender receives 3-dupack, which is a partial ACK, it will simply retransmit the lost packet and will not terminate the fast recovery until the all dropped packets are recovered. Hence transmission rates is maintained because it reduces the sending rate after all lost packets are retransmitted. Namely, TCP New Reno fast recovery takes care of the multiple packet drop from one *cwnd*.

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

2.2 TCP Westwood

TCP Westwood is wireless TCP using end-to-end proactive congestion control. TCP Westwood estimates the current network bandwidth at the sender side. The sender estimates network bandwidth by exploiting the rate and pattern of returning ACK through the reverse links.

However, TCP Westwood does not distinguish the cause of packet loss. It will adjust the transmission rates 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 network condition which is changed by network traffic in links.

2.3 TCP New Jersey

TCP New Jersey improves the available bandwidth estimation algorithm using the TCP Jersey. It also adjusts the slow start threshold(ssthresh) based on the current estimation. TCP Jersey and New Jersey consist of two key components, the available bandwidth estimation algorithm and the congestion warning mechanism that helps the sender to effectively differentiate the cause of packet loss at an intermediate router. In New Jersey, the TCP sender estimates current available bandwidth based on the packet interarrival time on the receiver. TCP New Jersey handles reverse links with background traffic.

However, TCP New Jersey may experience the decreasing of throughput depending on background traffic in forward link to bring unexpected available bandwidth estimation. In addition, it cannot increase the reduced sending rates due to the packet loss effectively according to the cause of packet loss which contains both the packet loss by BER and by network congestion. Consequently, TCP New Jersey may degrade performance in wireless networks where packets are lost consistently by the high BER or the network congestion.

3 TCP NJ+

In TCP New Jersey, throughput may be reduced depending on the background traffic pattern. In addition, when the sender detects the packet loss or the retransmission timeout (RTO) is expired, it may not recover the dropped sending rate effectively according to the cause of packet loss.

We propose TCP NJ+ which improves the available bandwidth estimation algorithm and the recovery mechanism of TCP New Jersey in this paper. In the improved available bandwidth algorithm, the sender selects the maximum estimation to overcome the problem of available bandwidth estimation algorithm depending on background traffic pattern in comparison of two factors, the interarrival time of ACK packets at the sender and the interarrival time of data segments at the receiver. And also TCP NJ+ guarantees high throughput using the improved recovery mechanism which increases the reduced *cwnd* more effectively when the sender detects the packet loss or RTO expiration.

3.1 Improved Available Bandwidth Estimation

Both TCP Jersey and TCP New Jersey estimate the current available bandwidth based on Eq. (1).

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

Here R_n is the estimated bandwidth when ACK packet n arrives at time t_n at the sender, t_{n-1} is the previous ACK packet arrival time at the sender, L_n is the size of data packet n, and RTT is the round trip time at time t_n in TCP Jersey. Meanwhile the data segment arrival time at the receiver is found using the timestamps [10] option in the TCP header instead of using the ACK packet arrival time at the sender. Hence t_n and t_{n-1} in Eq. (1) are the data segment arrival time of the nth data packet and its previous data packet arrival at the receiver, respectively, in TCP New Jersey.

Because the available bandwidth estimation algorithm on TCP New Jersey is computed by the packet transmission time for the receiver, it can explore accurate estimation, if the network condition is degraded by background traffic in reverse links, which returns the ACK packet. On the other hand, it cannot calculate the accurate available bandwidth estimation, if the network condition is deteriorated from the background traffic in forward links that transmits the data packet. Accordingly, both TCP Jersey and TCP New Jersey suffer from the problem that the available bandwidth estimation algorithm depends on the background traffic pattern which brings the degradation of performance.

Hence, TCP NJ+ estimates the available bandwidth to compare two estimations in Fig. 1. The R_{sn} is the estimated bandwidth when the ACK packet n arrives at time t_{sn} at the sender, and t_{sn-1} is its previous ACK packet arrival time at the sender. The R_{rn} is the estimated bandwidth, by using timestamps option, when the data segment n arrives at time t_{rn} at the receiver and t_{rn-1} is the previous data segment arrival time at the receiver. L_n is the size of data packet n, and RTT is the round trip time. As the Fig. 1 presents, TCP NJ+ selects the maximum value over R_{sn} and R_{rn} to guarantee the appropriate sending rate. In conclusion, TCP NJ+ resolves the problem that the available bandwidth estimation algorithm on TCP New Jersey depends on the background traffic pattern.

```
Initialization :
        $n \leftarrow 1$
        $R_{s0}, R_{r0}, t_{s0}, t_{r0} \leftarrow 0$
Procedure:
        ACK packet arrived at the sender
        if(timestamp )
            $R_{sn} \leftarrow (RTT \times R_{sn-1} + L_n)/((t_{sn} - t_{sn-1}) + RTT)$
            /* ABE based on ACK packet inter arrival time */
            $R_{rn} \leftarrow (RTT \times R_{rn-1} + L_n)/((t_{rn} - t_{rn-1}) + RTT)$
            /* ABE based on data packet inter arrival time */
            $R_n \leftarrow \max(R_{sn}, R_{rn})$
            /* maximum value of two estimations */
            $n \leftarrow n + 1$
        end if
```

Fig. 1. Improved available bandwidth estimation

It achieves higher throughput regardless of the direction of background traffic as you see in section 4.

3.2 Improved *RTO* Mechanism

If the 'timeout' is expired, the TCP sender concludes that the network is congested and reduces the *ssthresh* to the half of the current *cwnd* and the *cwnd* to one. In TCP Reno and TCP New Reno, *RTO* mechanism is operated by the AIMD algorithm [2]. Hence TCP Reno and TCP New Reno decrease the throughput because the *RTO* mechanism, which induces to drop the sending rate, is frequently occurred due to the high probability of packet loss in high BER wireless networks.

An optimized window (*ownd*) in TCP New Jersey is computed by Eq. (2) shown below.

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

Here R_n is the value of available bandwidth estimation. If the *RTO* is expired by the nth packet, TCP New Jersey decides whether the packet loss is caused by either BER or the network congestion. If packet loss is caused by network congestion, the sender sets the *cwnd* to one and the *ssthresh* to $ownd_n$. Otherwise, it adjusts the *cwnd* and *ssthresh* to $ownd_n$. Since *RTO* is caused by BER instead of network congestion and the network condition becomes poor incidently, we can utilize the remainder of network bandwidth due to *cwnd* sets to one. Therefore it is inappropriate to set the *cwnd* to $ownd_n$ which is computed by the minimum estimation when the link condition is degraded temporarily. In addition, when the network condition is dropped temporarily, it will decrease the throughput because the *cwnd* may be set to unexpected values.

For this reason, as TCP NJ+ experiences *RTO*, it distinguishes the cause of network congestion and the BER. Depending on the cause, with *RTO* due to especially network congestion, TCP NJ+ sets the *cwnd* to one and the *ssthresh* to $ownd_n$ as in the TCP New Jersey behavior. And for *RTO* due to the BER, TCP NJ+ sets the *ssthresh* to $ownd_n$ and the *cwnd* to the value according to the algorithm given in Fig. 2 which is to employ $ownd_{n-1}$ as well as $ownd_n$. In

```
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
```

Fig. 2. Improved *RTO* mechanism

TCP NJ+, $ownd_{n-1}$ value is higher than $ownd_n$ because it is computed when the network condition is in a good state. TCP NJ+ achieves higher *ownd*, average of $ownd_{n-1}$ and $ownd_n$, and results in higher *cwnd* than TCP New Jersey, because the average of $ownd_{n-1}$ and $ownd_n$ are always higher than $ownd_n$. Hence the throughput is guaranteed to reduce the recovery time of the dropped *cwnd* in order to avoid the network congestion and to utilize link bandwidth effectively in TCP NJ+. In addition, because *RTO* frequently occurs in high BER wireless networks, it shows higher performance over other wireless TCP schemes.

3.3 Improved Recovery Mechanism

TCP New Jersey differentiates the packet loss caused by the BER from that caused by network congestion using CW mechanism [8]. When the TCP New Jersey receives 3-dupack, the error recovery mechanism executes as follows. First, if the packet loss is caused by the network congestion, *ssthresh* is set to $ownd_n$, then if *cwnd* is lower than *ssthresh*, it is set to the current *cwnd*. But if the *cwnd* is higher than the *ssthresh*, it is set to $ownd_n$. Second, if the cause of packet loss is the BER, TCP New Jersey maintains the current *ssthresh* and *cwnd*. Therefore, there is no way to adjust *cwnd* when packet loss is caused by the BER. However the sender increases *cwnd* for every ACK it receives. But, when the sender receives the 3rd duplicated ACK, it does not increase the *cwnd*. This means that the relative loss of *cwnd* is increased according to growing packet loss.

```
if (3 dupack received by sender)
    if (Congestion Warning)
    /* if packet loss due to congestion*/
        ssthresh = $ownd_n$;
        if (ssthresh < cwnd)
            cwnd = $ownd_n$;
        end if
    else
    /* if packet loss due to BER */
        ssthresh = $ownd_n$
        cwnd = cwnd + 1;
    end if
end if
```

Fig. 3. Improved recovery mechanism

TCP NJ+ handles such problem in TCP New Jersey. The improved error recovery mechanism in TCP NJ+ is shown in Fig. 3. When TCP NJ+ receives 3-dupack, the improved error recovery mechanism is performed as follows. First, if packet loss is caused by network congestion, TCP NJ+ operates the same as TCP New Jersey. Second, when packet loss caused by BER occurs, *ssthresh* is set to $ownd_n$ and *cwnd* is increased by 1 maximum segment size (MSS). The reason why the *cwnd* is increased by 1 MSS is to compensate the lost *cwnd* by the 3rd duplicated ACK, because the *cwnd* is not increased by fast recovery algorithm. However, the packet loss caused by the BER may utilize the remaining bandwidth. Hence the adjustment of the *cwnd* effectively is to invoke the improvement in performance. In TCP NJ+, the new recovery mechanism, which achieves the higher *cwnd*, ensures the remarkable performance improvement comparing to other wireless TCP schemes in high BER links where the packet loss occurs more frequently.

4 Simulation Results

We evaluate the goodput and fairness performance of TCP NJ+ using the *NS-2* network simulator. One is a simple network topology and the other is a more realistic network topology with background traffic, as shown in Fig. 4 and Fig. 6, respectively. Various simulation parameters we use are presented in Table 1 [12].

Table 1. Simulation parameters

Bandwidth	Wired : 100MB / Wireless : 2MB
Packet Size	762byte
Propagation Delay	Wired : 10 - 20ms / Wireless : 1ms
Queue Size	20 - 200 packets

4.1 Goodput Performance as Simple Topology

Goodput is the effective amount of data delivered through the network. It is a direct indicator of network performance. We evaluate the goodput of TCP NJ+, New Jersey, Westwood, and Reno on various wireless link error rates using the simple topology shown in Fig. 4.

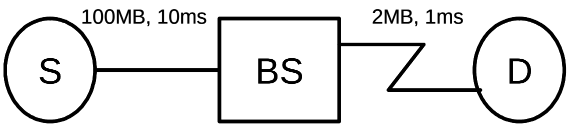


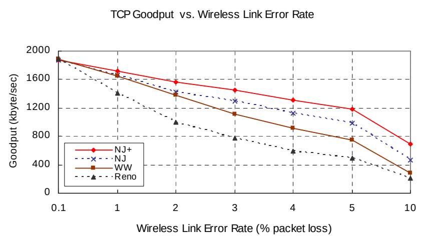
Fig. 4. Simple network topology

The source (Node S) connects to Node BS via a 100MB wired link with 10ms propagation delay. Node BS is linked to the destination (Node D) via a 2MB wireless link with 1ms propagation delay. The queue size of the wired link is set to 150 and the wireless link queue size is set to 20 respectively. The goodput result is shown in Fig. 5(a).

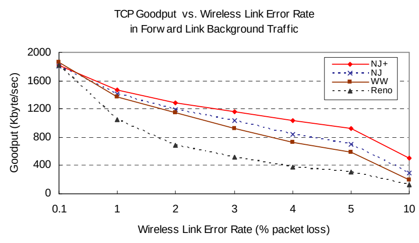
TCP NJ+ shows a higher goodput performance, when wireless link error rate increases. Especially, in a 5% wireless link error rate, TCP NJ+ outperforms TCP New Jersey by 19% and TCP Westwood by 54%.

4.2 Goodput Performance with Background Traffic

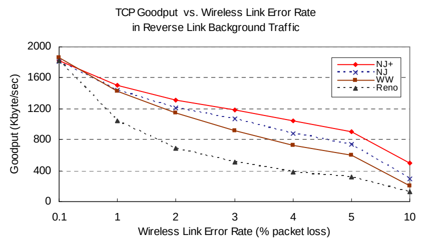
In TCP NJ+, the available bandwidth estimation algorithm guarantees considerable throughput regardless of the background traffic pattern. As illustrated in Fig. 6, we measure the goodput of TCP NJ+, TCP New Jersey, TCP Westwood, and TCP Reno on various wireless link error rates under forward where data packets are transferred, reverse where ACK packets are traversed, or bi-directional background traffic. The source (Node S) connects to Node R1 via a 100MB wired link with 10ms propagation delay. R1 is linked to Node BS via a 100MB wired link with 20ms propagation delay. The asymmetric wireless link from BS to the destination(Node D) is represented by the differing bandwidth on



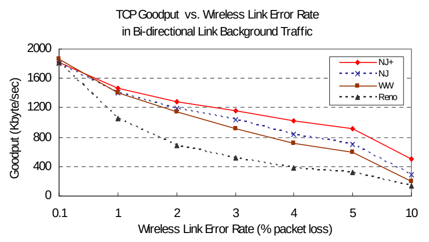
(a) Goodput vs. wireless link error rate



(b) Forward link background traffic



(c) Reverse link background traffic



(d) Bi-directional link background traffic

Fig. 5. TCP NJ+ goodput performance

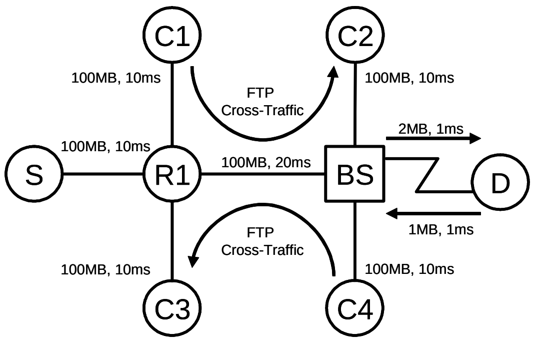


Fig. 6. Simulation topology with background traffic pattern

the downlink (2MB) and uplink (1MB) with 1ms propagation delay. The cross-traffic flows, Node C1 to Node C2 (forward direction) and Node C4 to Node C3 (reverse direction), and in both direction are FTP background traffic via a 100MB wired link with 10ms propagation delay. The queue size of the wired link is set to 200 and the wireless link queue size is set to 20.

The goodput result of the FTP forward background traffic is illustrated in Fig. 5(b). In a 5% wireless link error rate, TCP NJ+ outperforms TCP New Jersey by 30% and TCP Westwood by 55%. The goodput result of the FTP reverse background traffic is presented in Fig. 5(c). In 5% wireless link error rate, TCP NJ+ outperforms New Jersey by 20% and Westwood by 45%. The goodput result of the FTP bi-directional background traffic is presented in Fig. 5(d). TCP NJ+ outperforms New Jersey by 31% and Westwood by 56% in a 5% wireless link error rate. The simulation result shows that TCP NJ+ achieves higher goodput than New Jersey regardless of the background traffic pattern when the wireless link error rate increases.

4.3 Fairness

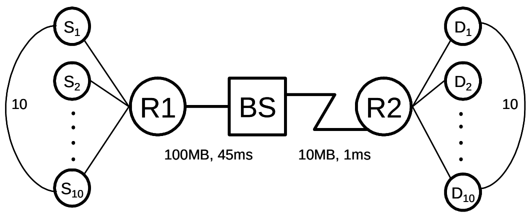


Fig. 7. Simulation topology for fairness

Fairness is also an important metric 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 [13] in order to show the fairness of TCP NJ+, New Jersey, Westwood, and Reno on various link error rates using the topology in Fig. 7. The fairness 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	WW	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

5 Conclusion

We have proposed TCP NJ+, to improve the performance of TCP New Jersey. Three enhanced mechanisms are proposed in TCP NJ+. First, the improved

ABE guarantees higher throughput regardless of the background traffic pattern because it estimates the optimal available bandwidth. Second, when *RTO* due to BER occurs, the improved *RTO* mechanism inflates by reducing the *cwnd* more quickly. Third, when the packet loss caused by BER occurs, the improved recovery mechanism makes the reduced *cwnd* to be increased quickly.

Results from the simulations 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 a 5% wireless link error rate with no background traffic. Under a 5% wireless link error rate with background traffic, TCP NJ+ outperforms New Jersey by 27% and Westwood by 52%. In addition, the fairness is also satisfied. In conclusion, TCP NJ+ with the improved ABE, *RTO*, and recovery mechanisms are robust to high BER environments, showing significant performance improvements.

Acknowledgments. This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment), IITA-2006-(C1090-0603-0046)

References

1. Postel, J.: Transmission control protocol. RFC **793**, (1981)
2. Allman, M., Paxson, V., and Stevens, W.: TCP Congestion control. RFC **2581**, (1999)
3. Xylomenos, G., Polyzos, G.C., Mahonen, P., and Saaranen, M.: TCP performance issues over wireless links. IEEE Communications Magazine, Vol. **39**, (2001) 52–58
4. Lakshman, T. V. and Madhow, U.: The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Transactions on Networking, Vol. **5**, (1997) 336–350
5. Tian, Y., Xu, K., and Ansari, N.: TCP in wireless environments: problems and solutions. IEEE Radio Communications, Vol. **43**, (2005) 27–32
6. Floyd, S. and Henderson, T.: The New Reno modification to TCP'S fast recovery algorithm. RFC **2582**, (1999)
7. Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M.Y., and Wang, R.: TCP Westwood: bandwidth estimation for enhanced transport over wireless links. ACM/IEEE MobiCom, (2001) 287–297
8. Xu, K., Tian, Y., and Ansari, N.: TCP-Jersey for wireless IP communications. IEEE Journal of Selected Areas in Communications, Vol. **22**, (2004) 747–756
9. Xu, K., Tian, Y., and Ansari, N.: Improving TCP performance in integrated wireless communications networks, Computer Networks, Vol. **47**, (2005) 219–237
10. Jacobson, V., Braden, R., and Borman, D.: TCP extensions for high performance, RFC **1323**, (1992)
11. UCB/LBNL/VINT Network Simulator Online Avilable: http://www.isi.edu/nsnam/ns
12. Song, C., Cosman, P.C., and Voelker, G.M.: End-to-end differentiation of congestion and wireless losses, IEEE/ACM Transactions on Networking, Vol. **11** (2003) 703–717

13. Jain, R., Chiu, D., and Hawe, W.: A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, Research Report TR-301, (1984)