Selective-TCP for Wired/Wireless Networks

Rajashree Paul and Ljiljana Trajković[*]
Simon Fraser University
Vancouver, BC, Canada
{rpaul2, ljilja}@cs.sfu.ca

Keywords: Wired/wireless networks, TCP NewReno, Selective-TCP, loss detection.

Abstract
One of the main reasons for TCP's degraded performance in wireless networks is TCP's interpretation that packet loss is caused by congestion. However, in wireless networks, packet loss occurs mostly due to high bit error rate, packet corruption, or link failure. TCP performance in wired/wireless networks may be substantially improved if the cause of packet loss could be distinguished and appropriate rectifying measures taken dynamically. We propose a new end-to-end TCP protocol named Selective-TCP, which distinguishes between congestion and wireless link transmission losses (high bit error rate and/or packet corruption). When detecting packet loss, Selective-TCP invokes correction mechanisms. It is suited for mixed wired/wireless networks and shows increase in goodput when compared to TCP NewReno.

1. INTRODUCTION

The transmission control protocol (TCP) is the most extensively used transport protocol by Internet applications due to its robust and reliable connectivity. Wireless communication technology has become widely popular for access networks. These wireless access networks, such as Wireless Local Area Networks (WLAN) and cellular networks, are usually connected to a wired backbone network. Although TCP is very reliable in wired networks, its performance deteriorates in wireless environments. The main reason for TCP's performance degradation in wireless networks is TCP's inability to distinguish congestion losses from other types of losses. In wireless links, the reasons for packet loss are high bit error rates (BER) due to bursty nature of wireless traffic, packet corruption, or link outage. However, TCP treats all these errors as congestion and initiates the congestion control mechanism. This results in low utilization of available bandwidth, unnecessary retransmissions, and, ultimately, low goodput and throughput. (Throughput is defined as the number of bits transmitted by the source host and it is presented in kbps, while goodput is the number of bits received by the destination host, less the duplicates.)

In this paper, we propose Selective-TCP, an end-to-end design that improves TCP performance in wired/wireless networks. It distinguishes packet loss due to congestion from packet loss due to transmission in wireless links. Selective-TCP is an extension to TCP NewReno. We used the ns-2 network simulator to evaluate its performance.

This paper is organized as follows. In Section 2, we provide background material and a short review of previous work. Selective-TCP is introduced in Section 3. Simulation results are given in Section 4. We conclude with Section 5.

2. BACKGROUND AND RELATED WORK

2.1 Background

Several techniques have been proposed to mitigate the effects of non-congestion related losses on TCP's performance. They may be classified as: end-to-end (TCPReno [1], NewReno [2], and SACK [3]), link layer (Snoop-TCP [4] and TCP Packet Control [5]), and split-connection (M-TCP [6] and I-TCP [7]) approaches. Comparative analysis of these approaches [8] indicates that link layer techniques are most effective in improving TCP performance in wireless networks, while split-connection based methods sometimes lead to poor end-to-end throughput due to shielding of the wireless from the wired section of the network. End-to-end schemes, although less effective than link layer based techniques, are the most promising because they achieve significant performance gain without requiring expensive changes in the intermediate nodes. We consider here end-to-end solutions that require changes only to the sender and the receiver.

Selective-TCP algorithm employs a loss differentiation scheme [9] that enables a TCP receiver to distinguish congestion losses from wireless losses based on packet inter-arrival times at the receiver. In the case of wireless losses, Selective-TCP uses TCP with Selective Negative Acknowledgement (SNACK) option. SNACK is an extension of the Space Communication Protocol Standard-Transport Protocol (SCPS-TP) [10]. SNACK improves performance of SCPS-TP in the case of high bit error rates

[*] This research was supported by the NSERC Grant 216844–03 and Canada Foundation for Innovation.

by increasing link utilization and throughput. It has been widely used to improve TCP performance over satellite links and is rarely used in wireless links even though satellite and wireless links have similar characteristics: both are prone to high BER and packet corruption and introduce long network delays compared to wireline networks. Selective-TCP consists of a TCP-SNACK module implemented in TCP sink and a SNACK processing module implemented in TCP NewReno source [11]. In the case of packet loss due to wireless link errors, TCP sink sends acknowledgement with SNACK option to the source. This results in better bandwidth utilization, end-to-end throughput, and goodput. In the case of congestion loss, receiver sends the information about the bandwidth (measured at the receiver) to the sender. Sender then accordingly sets its congestion window. This restricts the TCP's additive increase multiplicative decrease (AIMD) algorithm from setting congestion window lower than necessary. As a result, network goodput and throughput are improved.

2.2 Related Work

Previous work on improving TCP performance in wired/wireless networks was based on distinguishing types of packet losses.

TCP Veno [12], a combination of TCP Vegas [13] and TCP Reno [1], employs an end-to-end congestion control mechanism. If a packet is lost, TCP Veno employs proactive congestion control of TCP Vegas and, thus, distinguishes between the congested and non-congested network states. TCP Veno does not address the issue of burst errors and no corrective action is taken for wireless losses.

The differentiation between congestion and random losses in wireless networks is achieved by measuring the variation of round trip delay [14], [15]. If the loss is not due to congestion, TCP congestion control is suppressed and a modified recovery strategy is implemented. Two additional detection schemes [16], [17] are based on controlling the TCP AIMD algorithm. None of these schemes imposes corrective actions in the case of wireless losses.

SNACK-Snoop [18] combines SNACK with Snoop [4]. It uses SNACK to provide explicit wireless loss notification between a base station and a mobile host. This is a link layer based scheme and requires major modification at the intermediate base station. It also introduces processing and memory overhead of the Snoop protocol at the base station.

TCP-Jersey [19] incorporates available bandwidth estimation at the sender, as is the case in TCP Westwood [20]. TCP-Jersey improves network throughput by estimating bandwidth in the case of congestion losses. It differentiates congestion from non-congestion losses with the help from intermediate routers and, thus, requires expensive changes at the routers. TCP-Jersey does not address corrections specific to wireless losses.

TCP-Real [21] is a receiver-oriented congestion control mechanism. If the network is congested, the receiver determines data sending rate and communicates that information to the sender. TCP-Real employs two corrective mechanisms: congestion avoidance and advanced error detection.

3. SELECTIVE-TCP ALGORITHM

3.1 Overview of the Proposed Algorithm

Selective-TCP algorithm is based on differentiating the types of losses at the TCP receiver. It may be implemented as an extension to TCP NewReno. If an out-of-sequence packet is received at the sink, Selective-TCP distinguishes the loss due to congestion from the loss due to wireless transmission error. The loss differentiation technique ([9], [22]) used in Selective-TCP is based on packet inter-arrival times measured at the receiver. After distinguishing the type of packet loss, one of the corrective measures is taken to improve TCP performance.

In the case of loss due to congestion, the bandwidth is measured at the receiver and sent to the sender, rather than estimating available bandwidth at the sender only [19]. The sender then accordingly adjusts its congestion window size. Thus, Selective-TCP prevents TCP's AIMD scheme from setting the sender's congestion window size lower than necessary. Selective-TCP helps the TCP NewReno sender to achieve the optimum bandwidth faster, resulting in higher bandwidth utilization.

In the case of wireless transmission losses, receiver sends acknowledgement with SNACK option. This helps avoid TCP NewReno's congestion control mechanism. As a result, the slow start threshold and congestion window size are not reset unnecessarily, resulting in better bandwidth utilization.

3.2 Loss Differentiation Mechanism

The loss differentiation technique used in Selective-TCP is based on measuring the packet inter-arrival times at the receiver. The heuristic technique assumes that the wireless link is the only network bottleneck, that it is the last hop in the connection path, and that sender performs bulk data transfer [9]. We also assume that all packets are of same size. The wireless link is the only bottleneck in the network and, hence, packets accumulate at the base station. Therefore, most packets will be sent back-to-back over the wireless link. In the case of no packet loss, let T be the minimum packet inter-arrival time at the receiver. If there is no packet loss then the inter-arrival time between two consecutive packets is ~T as shown in Fig. 1(a). If a packet is lost in the wired link, the packet inter-arrival gap is still ~T because the packets queue at the base station before being transmitted on the wireless link, as shown in Fig. 1(b). However, if a packet is lost in the wireless link, the inter-arrival gap at receiver is ~2T because the lost packet has traveled on the wireless link for some time before being lost, as shown in Fig. 1(c). Using this heuristic, the type of packet loss is differentiated according to Algorithm 1.

```
n = number of packets lost between two packet arrivals
if ( packet loss){
        if ((n+1)T ≤ packet inter-arrival time < (n+2)T)
                wireless transmission loss
        else
                congestion loss
}
```

Algorithm 1. Pseudo-code of the algorithm for differentiating types of packet losses [9].

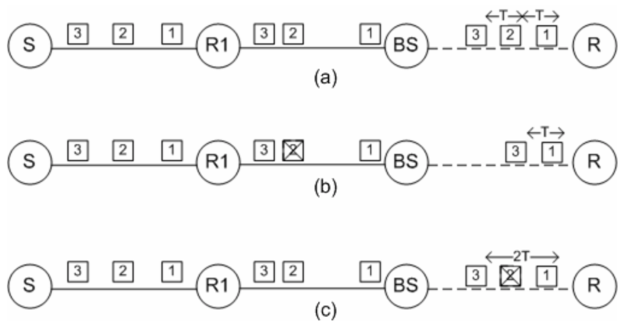

Fig. 1. Inter-arrival times between consecutive TCP packets: (a) no packet loss, (b) packet loss in the wired link, and (c) packet loss in the wireless link. R1 is an intermediate router and BS is the base station for the wireless destination node R. Solid and dashed lines represent wired and wireless link, respectively.

3.3 Selective Negative Acknowledgement (SNACK)

SNACK is an option of SCPS-TP [10]. It is widely used for improving TCP performance over satellite links. By sending SNACK, the receiver informs the sender about the segments that it did not receive. SNACK conveys information about multiple lost segments. This helps TCP to continue transmission even after a packet loss without reducing the congestion window size. SNACK information is stored in the optional area of the TCP header. In the presence of packet re-ordering, it is advantageous to delay SNACK because this permits re-ordered packets to reach the receiver. On receiving a SNACK, the TCP sender aggressively retransmits all packets indicated as empty slots in the receiver queue. These aggressive retransmissions prevent retransmission time-outs and avoid link idle time, resulting in higher bandwidth utilization [11].

3.4 Selective-TCP Module at the TCP Receiver

If an out-of-sequence packet arrives at a TCP receiver, Selective-TCP first distinguishes the type of packet loss. In the case of congestion loss, Selective-TCP increments *congestion_count* counter. When a threshold value *k* is reached, the receiver measures the bandwidth and sends it to the sender for setting the congestion window size, rather than searching for the optimum congestion window size and waiting for TCP to initiate congestion control. Bandwidth is measured as: *(no. of received packets × size of packets in bits) / (inter-arrival time between previous in-sequence packet received and most recent in-sequence packet received × 1,000)* in kbps. To explain this further, bandwidth of a network is defined as the data rate supported by a network connection.

TCP-Westwood [20] estimates the available bandwidth at the sender side based on the interval of returning acknowledgements:

$$b_k = \frac{d_k}{(t_k - t_{k-1})},$$

where, d_k is the amount of acknowledged data at time t_k and t_{k-1} is the time when previous acknowledgement was received. This sample bandwidth is smoothed further by a low-pass filtering to obtain estimated bandwidth.

TCP-Jersey [19] adopts the same idea. It employs time-sliding window (TSW) estimator at the sender and estimated available bandwidth as:

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

where, R_n is the estimated bandwidth when the n-th acknowledgement arrives at time t_n. t_{n-1} is the time when previous acknowledgement arrived. L_n is the size of data acknowledged by the n-th acknowledgement and RTT is the TCP's estimation of the end-to-end RTT delay at time t_n. Since duplicate acknowledgements also account for available bandwidth, both these bandwidth estimation approaches [19], [20] consider duplicate ACKs in the cases of packet loss in the network.

Unlike TCP-Westwood and TCP-Jersey, Selective-TCP estimates bandwidth at the receiver and, hence, it measures the available bandwidth as the number of packets received in a given period of time. In cases of packet loss, the available bandwidth would be smaller since fewer packets will be received than in the case of no packet loss. Selective-TCP measures available bandwidth as:

$$BW = \frac{n_p * s_p * 8}{(t_n - t_{n-1}) * 1{,}000} kbps,$$

where, t_n is the time when most recent in-sequence packet is received, t_{n-1} is the time when previous in-sequence packet was received, n_p is the number of packets received within ($t_n - t_{n-1}$), and s_p is the size of packets in bytes.

In the case of wireless loss, the receiver sends ACK with SNACK option to the sender. As a consequence, the TCP sender retransmits the missing packets indicated by SNACK. Acknowledgments with SNACK options are sent after a certain delay (*snack_delay*). As a result, the chance of unnecessarily retransmitting a delayed or misordered segment is limited [23]. Since the SNACK option triggers a retransmission, there is no reliance on the Fast Retransmit algorithm to detect the loss. This independence from the Fast Retransmit algorithm is important because duplicate ACKs may never be received when operating over a highly lossy link. The pseudo-code is shown in Algorithm 2.

```
if (out-of-order packet received) {
    // check type of loss
    if ( wireless loss) {
            if (snack_delay = 0)
                send SNACK
            else
                do nothing
        }
        else { // congestion loss
            1) set congestion_count = congestion_count + 1
            2) set congestion_info = current bandwidth  measured at
            the TCP receiver
            if (congestion_count = k) {
                1) send congestion_info to the TCP sender
            2) reset  congestion_count
                }
            else
                send ACK //as in the case of TCP NewReno
                receiver
        }
}
else // in-sequence packet received
    send ACK // same as TCP NewReno receiver
```

Algorithm 2. Pseudo-code of the Selective-TCP algorithm at the receiver.

3.5 Selective-TCP Module at the TCP Sender

When a SNACK is received, the TCP sender aggressively retransmits the packet(s) indicated as lost packet(s), without waiting for retransmission time-out to occur. Hence, congestion control mechanism and unnecessary retransmissions are avoided, leading to higher bandwidth utilization. *Congension_info* stores the bandwidth measured at the receiver in time, packet loss is detected. If the *congestion_info* field in the TCP header has a non-zero value, the sender sets its congestion window size equal to *congension_info*base_rtt*, where *base_rtt* is the initial round trip time. This prevents the TCP AIMD algorithm from setting the congestion window size to be unnecessarily small. *Congestion_info* is multiplied by *base_rtt* to increase the congestion window size. The pseudo-code is shown in Algorithm 3.

```
if (SNACK received) {
    1) retransmit packet(s) indicated as lost
    2) reset retransmission timer
    }
else if (congestion_info ≠ 0) {
    // set size of congestion window equal tothe bandwidth
    // measured at receiver
    1) set cwnd_ = congestion_info * base_rtt
    // cwnd_ denotes congestion window size and base_rtt //is the
    base round trip time measured at sender
    2) reset congestion_info
    }
else // standard ACK received
    do as TCP NewReno sender
```

Algorithm 3. Pseudo-code of the Selective-TCP algorithm at the sender.

4. SIMULATION MODEL AND RESULTS

We simulated mixed wired/wireless network with a dumbbell topology where the source nodes are wired and destination nodes are wireless. Wireless links are the network bottleneck. We describe the network topology used in simulation scenarios, the error model, and the simulation results. We used network simulator ns-2.27 [24] to simulate the wired/wireless network and to evaluate performance of Selective-TCP as an extension to TCP NewReno.

4.1 Network Topology

The network (dumbbell) topology is shown in Fig. 2. The TCP sender is a wired node, while the TCP receiver is a wireless node. The TCP source sends file transfer protocol (FTP) traffic. The user datagram protocol (UDP) source sends constant bit rate (CBR) traffic. The UDP source is a wired node, while the UDP sink is a wireless node. The FTP traffic and the CBR traffic share a common wired link from router R1 to base station BS. The TCP and UDP sender rates are 2 Mbps and 512 Kbps, respectively. Wired links have 2 Mbps bandwidth. Bandwidth of the wired link between R1 and BS is: (i) 2 Mbps when simulating Selective-TCP's performance in presence of congestion (the sum of TCP and UDP data rates is 2.5 Mbps) and (ii) 4 Mbps when simulating the case without congestion. Propagation delay of the wired links is 1 ms. Wireless links have 1 Mbps bandwidth and 5 ms propagation delay. The model used for the wireless links is WLAN.

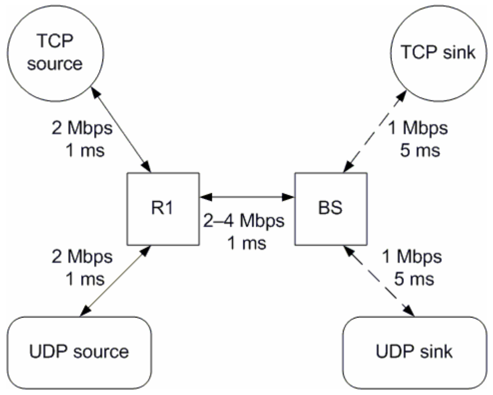

Fig. 2. Simulated network topology: R1 is an intermediate router. BS is the base station for the wireless destination nodes. Solid and dashed lines represent wired and wireless links, respectively.

4.2 The Error Model

We simulate wireless links with burst errors [25] using a two-state Markov model (Gilbert model), shown in Fig. 3. It has a good (error-free) and a bad (erroneous) state. Good state implies no packet loss, while bad state denotes 1 packet loss.

The model is defined by a transition probability matrix Π and the steady state error rate ε [26]:

$$\Pi = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix} \text{ and } \varepsilon = \frac{1-p}{2-p-q}.$$

We assume the wireless link is in the good state at the beginning of simulation and the transition probabilities p = 0.9913 and q = 0.8509 model the effect of burst errors. The error rate ε = 5%. These parameters have been reported for burst errors measured in deployed wireless networks [27].

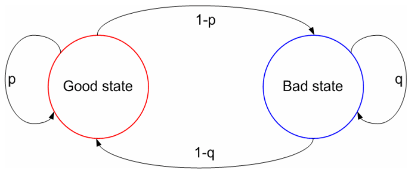

Fig. 3. Two-state Markov model: p is the probability of successfully transmitting a packet given that the previous packet was successfully transmitted; 1-q is the probability of successfully transmitting a packet given that the previous packet was dropped.

4.3 Simulation Results

We compare performance of Selective-TCP and TCP NewReno in the presence and in the absence of a congested link. In both cases, the error in the wireless link has been introduced. For the first 100 s of simulation time, there is only CBR (constant bit rate)/UDP traffic. After 100 s, TCP connection starts and exists along with UDP connection. All connections end after 300 s of simulation time.

1) Presence of a congested link: The common wired link has bandwidth 2 Mbps. Goodput in the presence of congested link is shown in Fig. 4. Selective-TCP achieves up to 45% higher goodput when compared to TCP NewReno. Selective-TCP shows larger congestion window size than TCP NewReno, indicating better utilization of available bandwidth, as shown in Fig. 5.

The slow start threshold and the average network throughput for Selective-TCP are shown in Figs. 6 and 7, respectively. A comparison of Selective-TCP performance with no wireless error, 1% random error, and 5% burst error, is shown in Fig. 8.

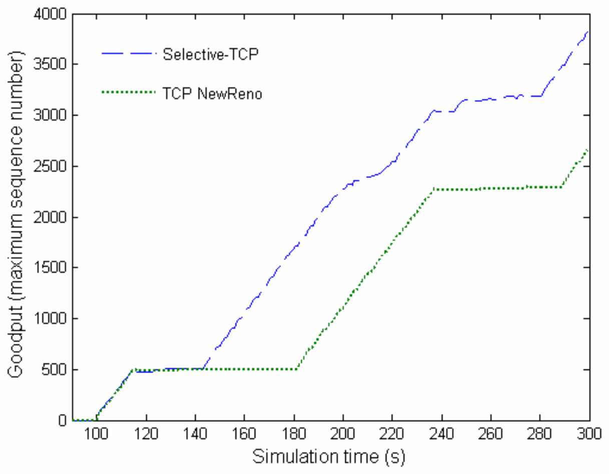

Fig. 4. Goodput is represented as the maximum number of packets that reached the destination.

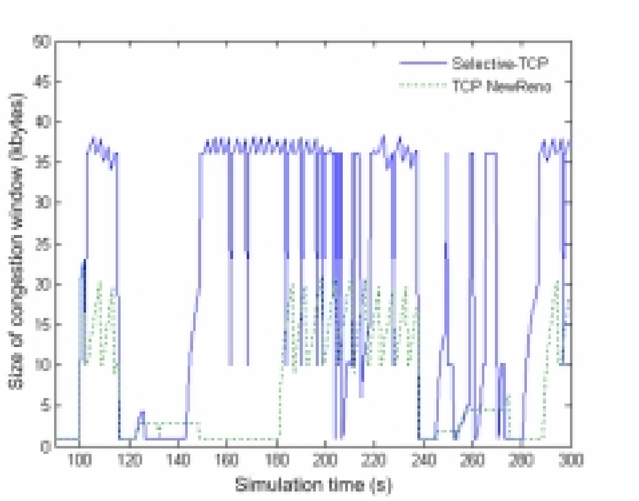

Fig. 5. Size of congestion window for Selective-TCP is significantly larger than for TCP NewReno.

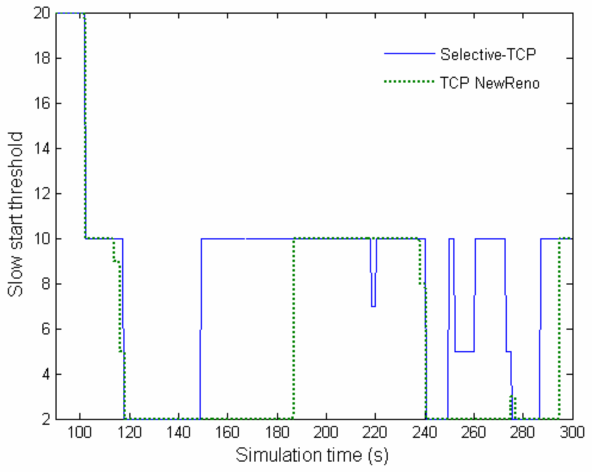

Fig. 6. Selective-TCP sender maintains a constant value of slow start threshold over a longer period of time. The initial value of the slow start threshold is equal to 20.

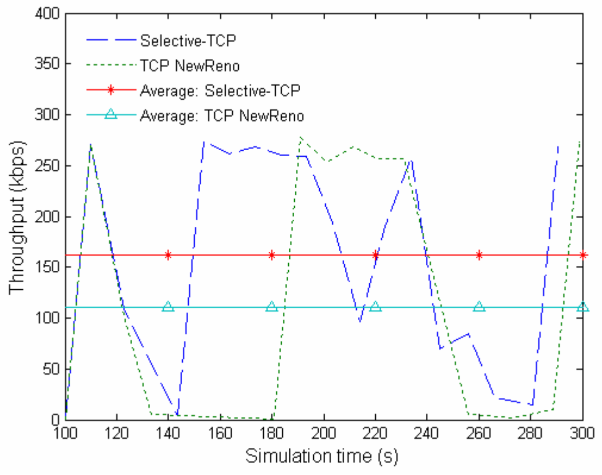

Fig. 7. The average throughput of Selective-TCP (161.5 kbps) is larger than the average throughput of TCP NewReno (110.91 kbps).

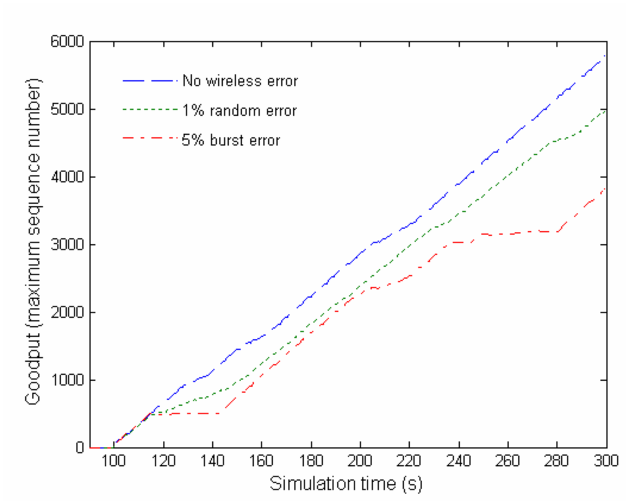

Fig. 8. Goodput of Selective-TCP: maximum goodput is achieved when no wireless error is introduced.

2) Absence of a congested link: The bandwidth of the common wired link is 4 Mbps. Goodput (maximum sequence number received) vs. simulation time without congestion in the common wired link is shown in Fig. 9. Selective-TCP improves goodput by 45% compared to TCP NewReno over 300 s of TCP connection time. Congestion window size, slow start threshold, and throughput as functions of time are shown in Figs. 10, 11, and 12, respectively. We compare performance of Selective-TCP and TCP NewReno in a non-congested network. Similar to the case of congested network, Selective-TCP performs better than NewReno for non-congested network. If no error is introduced in wireless link, Selective-TCP achieves ~1.5 times the goodput in the case of 5% burst error, as shown in Fig. 13.

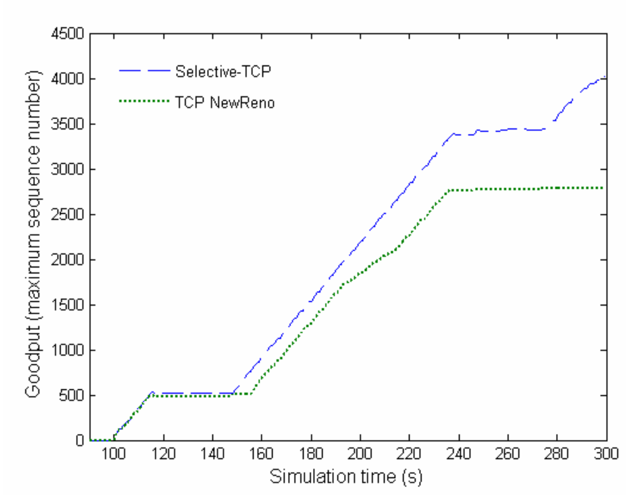

Fig. 9. Selective-TCP shows significant increase in goodput.

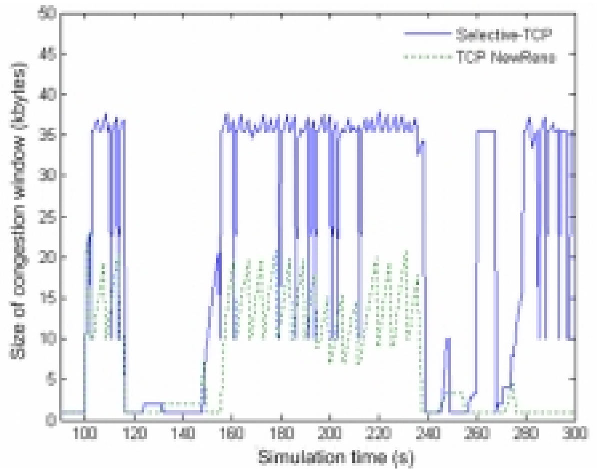

Fig. 10. The size of congestion window for Selective-TCP remains larger than for TCP NewReno.

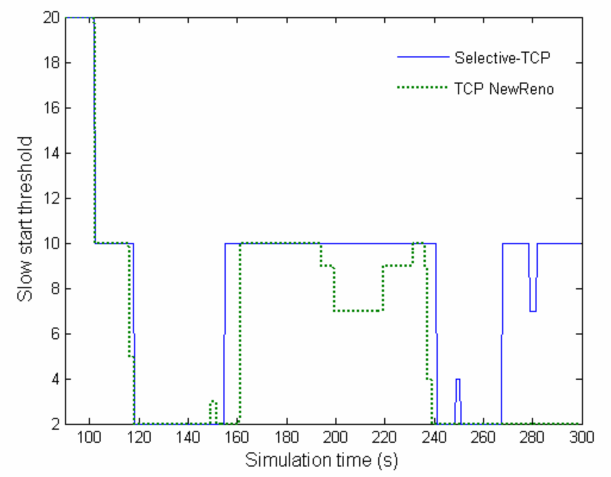

Fig. 11. Slow start threshold of Selective-TCP remains constant over a longer period of time.

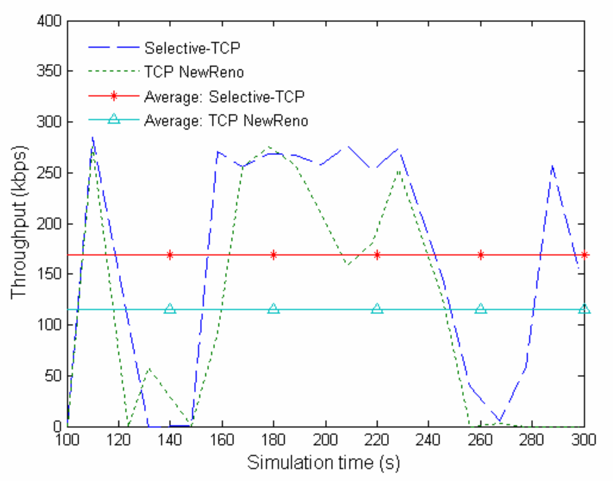

Fig. 12. Average throughput of Selective-TCP (169.15 kbps) and of TCP NewReno (115.65 kbps).

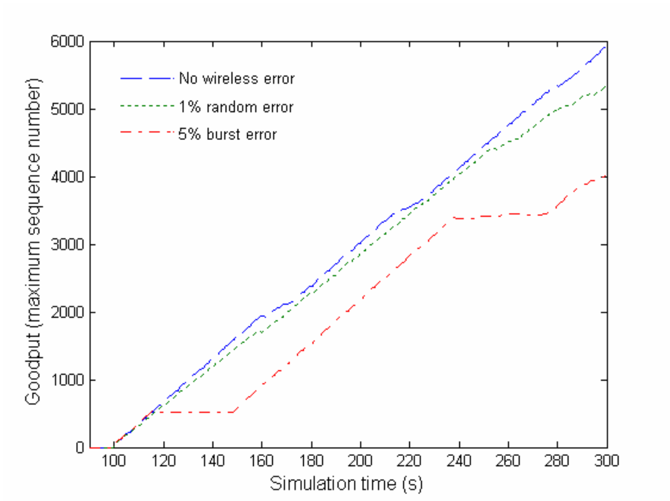

Fig. 13. Effect of wireless errors: goodput of Selective-TCP in the absence of congested link.

4.4 Comparison of Selective-TCP to Other TCP Variants

We also compare Selective-TCP and TCP NewReno with TCP Reno [1], TCP SACK [3], TCP Westwood [20], and TCP Packet Control algorithm [5].

1) Presence of congested link: Selective-TCP achieves larger congestion window compared to other TCP variants, as shown in Fig. 14. TCP SACK and TCP Westwood show smaller bandwidth utilization.

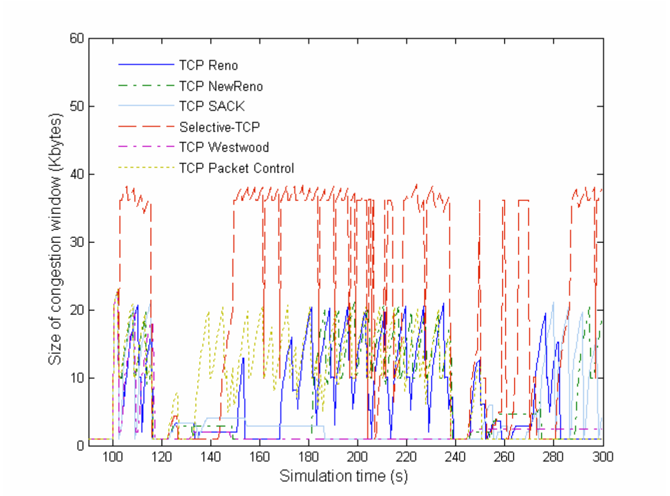

Fig. 14. Size of congestion window vs. simulation time: congestion window size is the largest for Selective-TCP, compared to other TCP variants.

TCP Packet Control algorithm achieves the highest goodput, followed by Selective-TCP, TCP Reno, and NewReno, as shown in Fig 15. However, performance of TCP Westwood and TCP SACK deteriorates significantly due to poor bandwidth utilization. TCP Packet Control algorithm, being the only link layer based algorithm, achieves the highest goodput in a mixed wired/wireless network with 5% burst error in the wireless links.

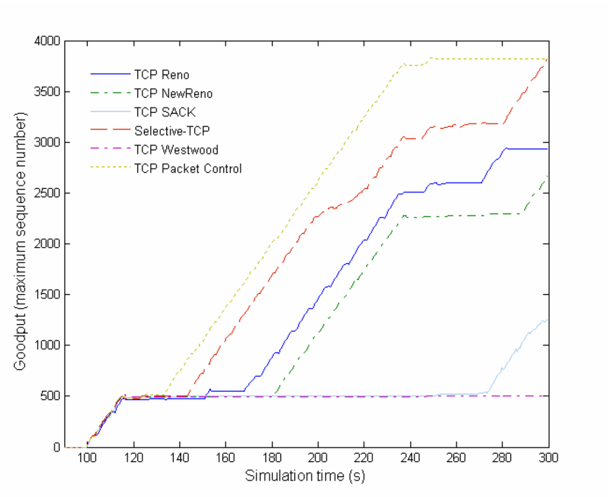

Fig. 15. Goodput vs. simulation time: network performance deteriorates for TCP SACK and TCP Westwood.

2) Absence of congested link: The congestion window sizes are shown in Fig. 16. Selective-TCP again has the largest congestion window because it measures the available bandwidth at the time of packet loss and accordingly sets congestion window size. TCP Westwood employs bandwidth estimation to set congestion window size, which should be larger compared to TCP Reno, NewReno, SACK, and TCP Packet Control algorithm (none employs bandwidth measurement/estimation). However, congestion window for TCP Westwood is similar to other TCP variants.

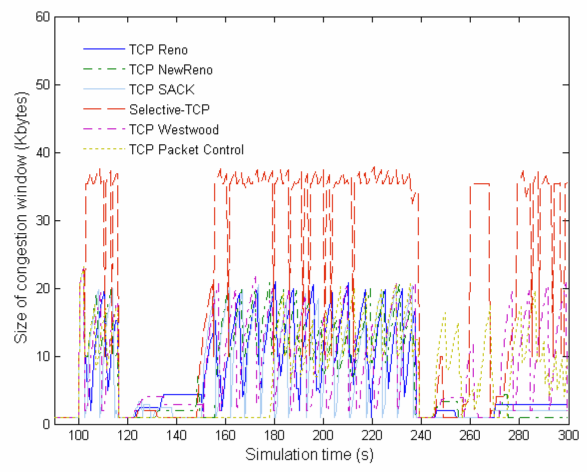

Fig. 16. Congestion window size for 300 s of simulation time.

Fig. 17 shows the goodput of TCP Reno, NewReno, SACK, Westwood, Selective-TCP, and TCP Packet Control algorithm for 300 s of simulation time. Selective-TCP performs best, followed by TCP Westwood because both algorithms employ bandwidth measurement/estimation. TCP Reno, NewReno, and SACK perform comparably. TCP Packet Control algorithm, unlike in the case of congestion, achieves lower goodput. Selective-TCP performs well in cases of both congested and non-congested links.

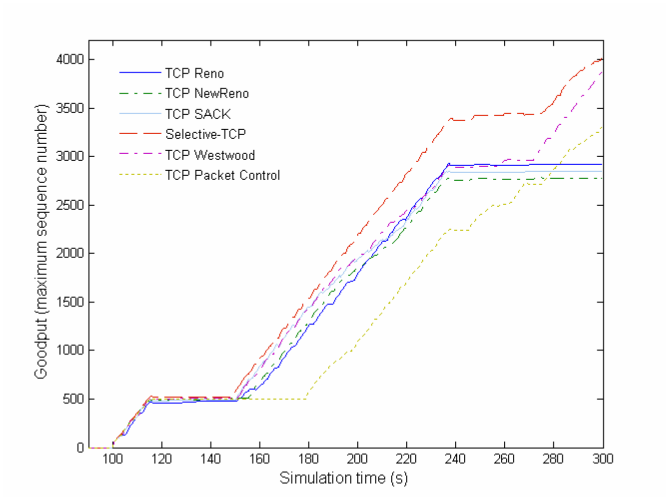

Fig. 17. Goodput vs. simulation time: Selective-TCP performs the best in the absence of congested link.

5. CONCLUSIONS

In this paper, we proposed Selective-TCP, a new end-to-end protocol for mixed wired/wireless networks. Selective-TCP distinguishes between congestion and wireless errors and takes corrective measures. In the case of wireless errors, the receiver sends SNACK to the sender and, thus, prevents the initiation of congestion control otherwise performed by TCP. When the loss is due to network congestion, the receiver informs the sender of the measured bandwidth and the sender accordingly sets the congestion window size. Thus, Selective-TCP stops the TCP AIMD algorithm from setting the congestion window lower than necessary. Selective-TCP is implemented as an extension to the TCP NewReno sender and receiver and requires no modifications in the intermediate routers. It improves the bandwidth utilization and increases goodput up to 45% compared to the TCP NewReno.

REFERENCES

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

[2] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," *IETF RFC 2582*, Apr. 1999.

[3] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," *IETF RFC 2018*, Apr. 1996.

[4] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. ACM Int. Conf. on Mobile Computing and Networking*, Berkeley, CA, Nov. 1995, pp. 2–11.

[5] W. G. Zeng and Lj. Trajković, "TCP packet control for wireless networks," in *Proc. IEEE WiMob*, Montreal, Canada, Aug. 2005, pp. 196–203.

[6] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Commun. Review*, vol. 27, no. 5, pp. 19–43, Oct. 1997.

[7] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proc. 15th ICDCS*, Vancouver, Canada, May 1995, pp. 136–143.

[8] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *ACM Computer Commun. Review*, vol. 26, no. 4, pp. 256–269, Aug. 1996.

[9] S. Biaz and N. H. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver," in *Proc. IEEE Symposium on ASSET*, Richardson, TX, Mar. 1999, pp. 10–17.

[10] Consultative Committee for Space Data Systems, *Space Communications Protocol Specification—Transport Protocol (SCPS-TP)*, Blue Book, issue 1, May 1999.

[11] D. Anantharaman, "Performance analysis of SNACK in satellite networks through simulation," M.S. Thesis, Lamar University, Lamar, TX, 2004.

[12] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE J. Select. Areas Commun.*, vol. 21, no. 2, pp. 216–228, Feb. 2003.

[13] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," in *Proc. SIGCOMM*, London, UK, Oct. 1994, pp. 24–35.

[14] N. K. G. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links," *IEE Proc. Commun.*, vol. 146, no. 4, pp. 222–230, Aug. 1999.

[15] D. Barman and I. Matta, "Effectiveness of loss labeling in improving TCP performance in wired/wireless networks," in *Proc. 10th IEEE ICNP*, Boston, MA, Nov. 2002, pp. 2–11.

[16] C. Parsa and J. J. Garcia-Luna-Aceves, "Differentiating congestion vs. random loss: a method for improving TCP performance over wireless links," in *Proc. IEEE WCN*, Chicago, IL, Sept. 2000, vol. 1, pp. 90–93.

[17] T. Kim, S. Lu, and V. Bharghavan, "Improving congestion control performance through loss differentiation," in *Proc. ICCCN*, Boston, MA, Oct. 1999, pp. 412–418.

[18] F. Sun, V. O. K. Li, and S. C. Liew, "Design of SNACK mechanism for wireless TCP with new snoop," in *Proc. IEEE WCNC*, Atlanta, GA, Mar. 2004, vol. 2, pp. 1051–1056.

[19] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE J. Select. Areas Commun.*, vol. 22, no. 4, pp. 747–756, May 2004.

[20] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: end-to-end congestion control for wired/wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 467–479, Sept. 2002.

[21] V. Tsaoussidis and C. Zhang, "TCP-Real: receiver-oriented congestion control," *ACM Computer Networks*, vol. 40, no. 4, pp. 477–497, Nov. 2002.

[22] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 703–717, Oct. 2003.

[23] R. C. Durst, G. J. Miller, and E. J. Travis, "TCP extensions for space communications," in *Proc. MOBICOM*, Rye, NY, Nov. 1996, pp. 15–26.

[24] ns-2 [Online]. Available: http://www.isi.edu/nsnam/ns.

[25] A. Gurtov and S. Floyd, "Modeling wireless links for transport protocols," *ACM Computer Commun. Review*, vol. 34, no. 2, pp. 85–96, Apr. 2004.

[26] J. McDougall and S. Miller, "Sensitivity of wireless network simulations to a two-state Markov model channel approximation," in *Proc. GLOBECOM*, San Francisco, CA, Dec. 2003, pp. 697–701.

[27] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A Markov-based channel model algorithm for wireless networks," *Wireless Networks*, vol. 9, no. 3, pp. 189–199, May 2003.