JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Updated Congestion Control Algorithm for TCP Throughput improvement in Wired and Wireless 5G Network

Kalpana Verma
Department of Computer Science and Engineering Institute of Technology & Management Lucknow,

Abstract-The main idea put forth in this paper is to find the Optimal Congestion Window for a TCP Sender in a specific network configuration (which corresponds to the fair share of that connection) and maintain this congestion window constant up until a point at which the network's fair share has significantly changed from the last window's size calculation instance. The TCP Congestion Window is now adjusted based on the specifics of the novel situation. Due to Modified TCP's congestion window, which maintains transmission at the same rate regardless of packet losses (whether from congestion or corruption), the suggested method is especially successful over wireless networks, which are intrinsically loss-prone.

Keyword- TCP Congestion, Wireless Network, Reserve field

I. INTRODUCTION

The well-known problem with using the TCP congestion control method in a wired/wireless setting is that it depends on packet loss as a gauge of network congestion [1, 2, 12]. A TCP Reno Sender reduces the congestion window (henceforth referred to as cwnd and represented in number of segments) and stops delivering packets in order to lessen the congestion scenario and prevent a congestion collapse. A crowded router is almost always the most likely source of packet loss in the network's wired segment, but a fading, loud radio channel is more likely to be the source in the wireless segment. Because TCP Reno is unable to differentiate and separate wireless loss from congestion loss, this causes issues for the protocol. Balakrishnan et al.'s work [3]–[4] compares and discusses various approaches to solving this issue. A thorough comparison was conducted between three alternative approaches: Split Connection, Localized Link Layer, and End-to-End (E2E). The split-connection method [13]–[14] defines the E2E reliability semantics. Second, a lot of condition maintenance at the base station is necessary with this strategy.

We suggest modifying the TCP congestion control method on the sender side in this study [5]. The main notion is that the Modified TCP Sender calculates its ideal fair share of bandwidth in the link and sets its own such that it can transmit at a rate that efficiently uses the fair share of bandwidth for a given network circumstance. Once the cwnd is adjusted tohas somewhat changed, considerably changing the connection's fair share. Performance degradation and cwnd reduction are not likely to occur when stray packet losses occur because the value of cwnd is not reduced at any packet loss indication, such as retransmission upon receiving a triple DUPKT or a coarse timeout due to the Retransmission Timer expiring. As a result, there is an improvement in wireless performance because the losses are not a sign of congestion but rather result from the radio propagation medium's intrinsic loss-proneness. Our claim that constant cwnd can beat TCP Reno in static (i.e., certain time interval) network settings is supported by simulation findings. The remainder of this essay is structured as follows:

Section 2 provides a summary of some related work;
Section 3 provides the analytical approach;
Section 4 explains the sender's algorithm;
Section 5 summarizes the simulation results;
Section 6 provides an idea of the difficulties encountered when putting such a strategy into practice; and
Section 7 wraps up the paper.

II. RELATED WORK

The measured rtt and the lowest rtt—or the one at the knee of the goodput – load curve—are compared using NCPLD [15]. Wireless faults are thought to be the source of a packet loss if the former is near the latter. Every time a packet is identified lost, TCPW [8]–[11] determines the congestion window based on the goodput, or reception rate. Congestion is thought to be the source of a packet loss if the current goodput is below a specific band around the mean; if not, wireless faults are blamed for the loss. The performance of the Modified sender and the TCPW sender is compared in this study using the TCPW bandwidth estimate scheme. In order to determine how much bandwidth is currently being used by the connection and hence accessible, the TCPW [8]–[11] sender keeps an eye on ACKs. Specifically, the sender makes use of two things: (1) the ACK reception rate; and (2) the information that an ACK contains about the quantity of data that has been transmitted to the destination. A quick explanation of the Westwood algorithm is given below. Assume at the moment tk that the source receives an ACK informing it that the TCP receiver has received dk bytes. By using the formula bk = dk/Δk, where Δk = tk−tk−1 and tk−1 is the time the previous ACK was received, we may determine the sample bandwidth used by that connection. The Tustin approximation is used to discretize a continuous low-pass filter, yielding the following discrete-time filter.

$$b'_k = \alpha_k b'_{k-1} + (1-\alpha_k)(b_k + b_{k-1})/2$$

Where b_k is the filtered estimate of the available bandwidth at time $t=t_{k..}$ a_k =(2- and is the cutoff frequency of the filter.
Here, seg_size identifies the length

```
Algorithm after n duplicate ACKs
The pseudo code of the algorithm is the following:
if (n DUPKTs are received)
    ssthresh = (BWE * RTTmin)/seg_size;
    if (cwin > ssthresh) /* congestion avoid */
        cwin = ssthresh;
    endif
endif
```

Here, seg_size identifies the length of the payload of a TCP segment in bits.

```
Algorithm after coarse timeout expiration
The pseudo code of the algorithm is:

if (coarse timeout expires)
    ssthresh = (BWE * RTTmin)/seg_size;
    if (ssthresh < 2)
        ssthresh = 2;
    endif;
    cwin = 1;
endif
```

III. ANALYTICAL APPROACH

The following would be a summary of the reasoning behind employing a constant window: As in [1], if Li is the load at instant i and we measure the network load using the average queue length at fixed intervals of a suitable length, we have the following for a congested network:

$$L_i = N + \gamma L_{i-1}$$

where γLi-1 represents the traffic that remains from the previous time interval and N, a constant, represents the average arrival rate of the incoming traffic. It appears that the phrase γLi-1 appears when a sender sends at a rate higher than its fair share, leaving a portion of the previous round's packets in the network when the packets from the subsequent round arrive. However, the γLi-1 disappears if the sender is sending at a rate that makes use of its fair share; equation (1) therefore reduces to

L_i=N

This is a constant; using a constant congestion window is based on this.

IV. TCP MODIFICATIONS

The primary notion here [5] is that we may partition a TCP connection's lifetime into a finite number of slots so that, in a given slot, the connection's fair share of the network stays essentially unchanged; that is, we can assume that the network scenario stays essentially unchanged with that particular slot. A slot finishes and the next one begins when there is a change in the network's available share because some connections are joining or departing the network. During these slots, we propose to use a constant TCP Congestion Window under the assumption that the network scenario stays the same. A window recalculation would occur at the start of each new slot, and the connection's available share would determine the cwnd. In the suggested mechanism, we estimate the available fair share using a bandwidth estimation algorithm akin to TCPW. Our model exploits the knee area in the rtt curve as in [15] to identify a change in fare share and prompt recalculation. The change in the rtt measurements is utilized as a trigger to switch from the constant window phase to the recalculation phase. The lifetime of the Modified TCP Sender in our model is divided into three separate phases: the startup phase, the mutually interleaved window recalculation phase, and the constant window phase.

A. The Startup Phase

The sender is completely unaware of the network condition during connection establishment. The Sender does not use standard default values for these fundamental connection properties in order to convey dynamic nature. As in [1], the sender employs a gradual start method. The sender keeps up the slow start procedure for, let's say, k rounds, during which it gathers a variety of crucial network data, including the minimum rtt measurement and a measurement of the connection's bandwidth. The sender computes the cwnd for the first time after the first k rounds since it has enough knowledge about the network by then.

B. Window Recalculation Phase

The TCP sender enters this phase when the trigger detects a change in the available fair share. The connection's most important phase is this one since it calculates the cwnd, which is maintained constant in the following phase. Therefore, the computed cwnd determines how well the sender performs and uses its portion of the network. In addition to the window recalculation procedure, the current value of the smoothed rtt measurements—obtained by applying a low pass filter, per Jacobson's recommendation [1], to the coarse rtt readings of the particular segment—is saved for later use. In order to ascertain the equitable portion of the connection within the network, an effective bandwidth estimation method needs to be implemented and stored for future use. The precision of this technique in estimating thenetwork share would determine the performance of the Modified TCP Sender

C. The Constant Window Phase

The sender monitors the rtt estimates from the segments that have been delivered during this phase of the connection, and if the percentage change in the smoothed rtt measurements over the archives rtt measure is greater than a specified threshold, the sender exits the constant window phase and enters the Window Recalculation Phase, i.e. if a window recalculation is made. The algorithm's pseudo code is as follows.

```
if( slow_start_state)
        slow_start(); /* open cwnd by one segment on each ACK arrival */
else
{
        if($|rtt_{arc}–rtt_{var}|/rtt_{arc}>\beta$)   /*fractional increase greater than threshold */
        {
                /* recalculate window and archive the value of $rtt_{var}$ */
                cwnd_ = (Estimated_Bandwidth* $rtt_{min}$) / seg_size_;
                if(cwnd_ < 1) cwnd_=1;
                        $rtt_{arc} = rtt_{var}$ ;
        }
}
```

The pseudo code uses three variables: rttmin, which is the estimated minimum value of rtt for the course of a certain connection, eg size_, which indicates the length of TCP segments in bytes, and Estimated Bandwidth, which is the bandwidth estimate produced by a bandwidth estimation algorithm.

V. PERFORMANCE ANALYSIS

The basic performance behavior of the modified TCP senders and their fairness when multiple connections share a bottleneck link are covered in this section. The TCP Reno and TCP Westwood [8]–[11] sources functioning in comparable network circumstances are compared in terms of performance. According to the literature reviewed, the buffer capacity of the intermediate node is always set to the bandwidth delay product for the bottleneck link. Performance is unaffected by further increasing the buffer capacity [15]. In order to give the sender data to send whenever the network allows, the traffic model utilized is FTP with infinite data to send. The packet size is fixed at 1000 bytes (1040 bytes with headers) in all trials. Error in the wireless subnetWe have employed the standard TCP Sink in our simulations, which returns an ACK for each packet it receives. The ACK path is free of errors or congestion. Every simulation has been run for 250 seconds, during which time the TCP senders have continuously transmitted data. With a maximum accessible bandwidth of 1Mbps, 802.11 MAC has been used for all of the simulations. An omni-directional antenna is utilized in conjunction with a two-ray ground propagation model. While the wireless subnet is subject to variable error rates, the cable subnet is error-free.

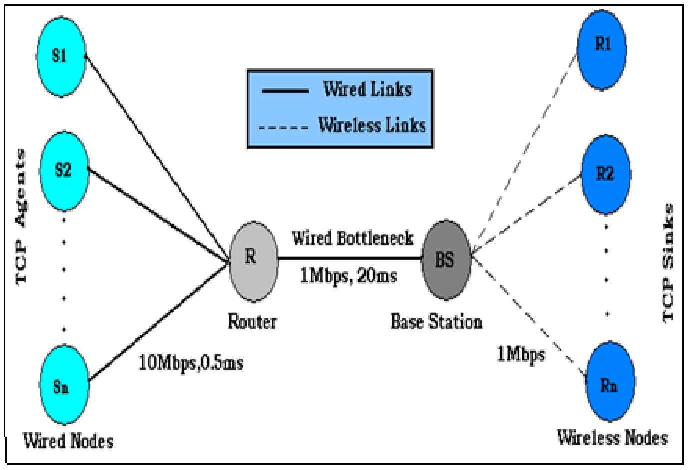

Figure 1: Network Scenario used for simulation

The throughput metric—that is, the quantity of data packets received by the sender—has been used to compare the performance of the Modified TCP Senders, TCP Westwood, and TCP Reno. We have examined the effectiveness of the Modified TCP's Constant Congestion Window feature and concluded that, in cases of wireless faults, a Constant cwnd TCP performs better than the Reno and Westwood sources during the time slots when a connection's share in the network stays constant. We assume that over the course of the simulation, a connection's share does not vary. The adjusted TCP Sender's cwnd has been established in accordance with the evaluation of the best cwnd for a particular circumstance. It should be mentioned that we are not modeling a modified TCP sender's whole lifetime. Instead, we limit our analysis to the connection's Constant Congestion Window phase. Using the LBL network simulator ns2 [6], [7], all simulations in this paper were run, with the necessary adjustments made to incorporate the modifications in the modified TCP sender. The matching TCPW modules were utilized for comparison with TCP Westwood [16]. A schematic of the simulation scenario is presented in Figure 1. The cable bottleneck connecting the intermediate router to the base station is shared by several TCP connections. Congestion does not cause any loss on a network with just one TCP connection. This means that wireless failures are the cause of any packet loss. Therefore, we are able to assess how well the Modified congestion management algorithm performs in situations where packet loss is only caused by wireless loss. The wired link between the router and base station will eventually get crowded as the number of source/receiver pairs increases. As a result, packet loss will occur from wireless failures as well as congestion. Thus, the same scenario may also be used to assess the Modified TCP sender's performance on crowded networks.

A. Constant Bit Error rates

The wireless subnet is vulnerable to constant bit error rates in the circumstances that are being examined. The performance of the Constant cwnd senders for various cwnd values is contrasted in Figure 2. It is clear that in every scenario, there is a value of cwnd (often more than one) at which the TCP Sender performs at its best. In the given network configuration, this is the best cwnd. Figure 2 shows the expression of cwnd in segments. Figure 3 shows that, under similar network conditions, a Constant cwnd TCP performs better than the Reno and Westwood senders. Figure 3 shows that a throughput improvement of 10-15% has been achieved. The mistake rates are shown as a percentage in Figure 3. Figure 4 presents a comparison of the TCP versions' performance when numerous connections share a wired bottleneck, resulting in packet loss due to congestion. In these kinds of situations, the Constant cwnd TCP sender performs better than Reno and Westwood.

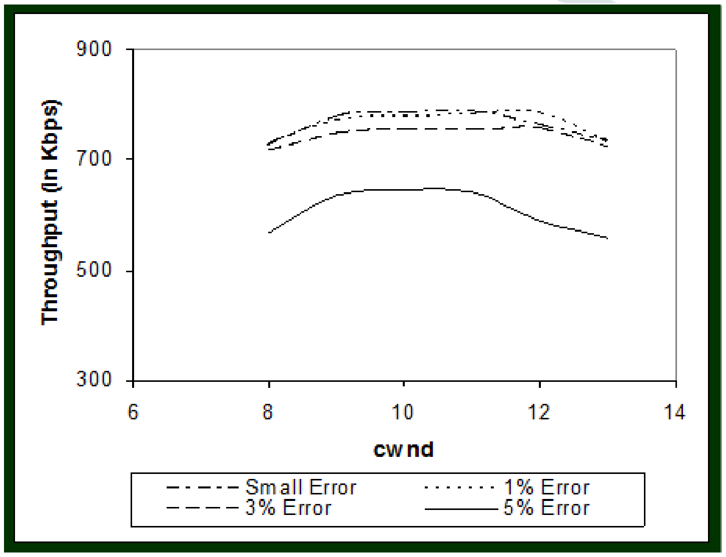

Figure 2: Variation of Throughput with varying cwnd for various bit error rates for single S/R pair.

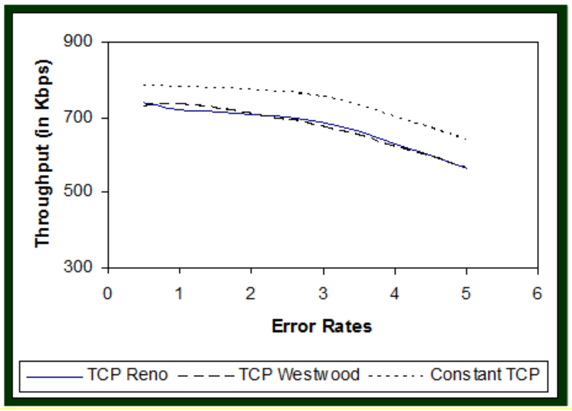

Figure 3: Variation of throughput with varying error rates in scenarios with constant bit error rates for single S/R pair

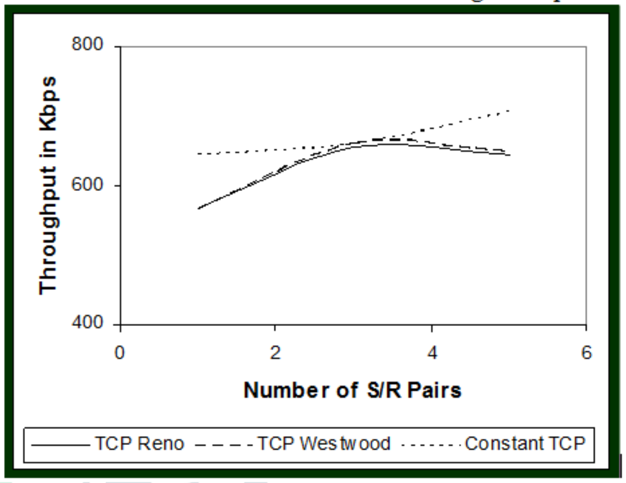

Figure 4: Variation of Throughput with number of TCP connections sharing the link in scenarios with 5% loss in constant bit error

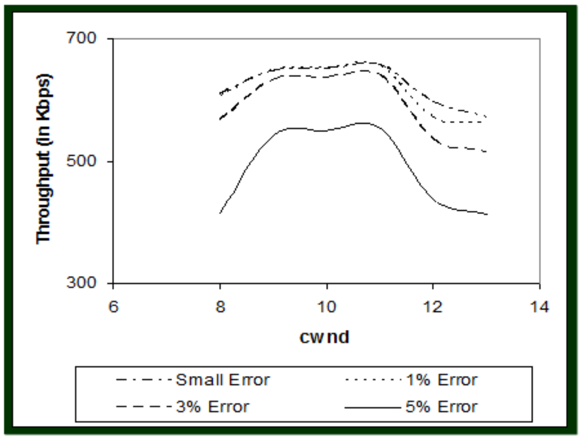

Figure 5: Variation of Throughput with varying cwnd error for various burst error for single S/R pair

B. Burst Error

Based on the throughput statistic, Modified TCP, TCP Reno, and TCP Westwood are compared in this subsection. The wireless subnet is prone to burst error in the cases that are being considered. A first order discrete time Markov model is used to model the burst mistake. The transition matrix describes the pattern of errors.

M|PBB PBG|

| PGB PGG|

The other elements in the matrix are defined similarly. Where is pBG, the change from bad to good, i.e., the conditional probability that successful transmission occurs in a slot given that a failure happened in the preceding slot? It should be emphasized that, according to a geometric random variable, represents 1/(1 - pBB), the average length of an error burst.

The performance of the Constant cwnd senders for various cwnd values is contrasted in Figure 5. It is clear that in every scenario, there is a value of cwnd (often more than one) at which the TCP Sender performs at its best. In the given network configuration, this is the best cwnd. Figure 5 shows the expression of cwnd in segments. The cwnd is set to the best value for the particular case when comparing the Constant cwnd TCP Sender's performance. Figure 6 shows that, under similar network conditions, a Constant cwnd TCP performs better than the Reno and Westwood senders. Figure 6 shows that a throughput improvement of 10–20% has been achieved. The mistake rates are shown as a percentage in Figure 6. Figure 7 presents a comparison of the TCP versions' performance when numerous connections share a wired bottleneck, resulting in packet loss due to congestion. In these kinds of situations, the Constant cwnd TCP sender performs better than Reno and Westwood.

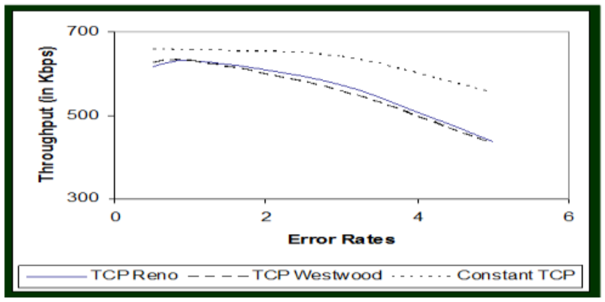

Figure 6: Variation of throughput with varying error rates in scenarios with burst error and for single S/R pair

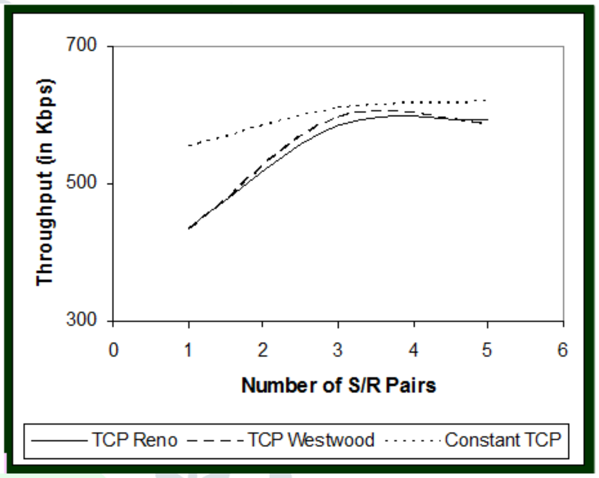

Figure 7: Variation of Throughput with number of TCP connections sharing the link in scenarios with 5% loss in burst error

VI. CHALLENGES IN IMPLEMENTING THE PROPOSED MECHANISM

We have suggested Sender side modification of the TCP congestion control method in the previous portions of the paper. In order for this system to function more effectively, a few obstacles must be overcome. First off, the available fair share of the connection in the network would be exactly calculated by the bandwidth determination algorithm. In the event of a wireless error, an inaccurate estimation would offset the performance improvement that would result from not decreasing the window. Second, a change in the network's fair share of available bandwidth could be quickly detected by the triggering mechanism. If this isn't done, there could be a risk of using more or less of the available fair share, depending on whether the connection's fair share rises or falls.

VII. CONCLUSION AND FUTURE WORK

We suggest a sender side tweak to the TCP congestion control mechanism in this work. Apart from this suggestion, we have assessed and contrasted the functioning of the altered TCP sender throughout a certain stage of its existence, known as the Constant Congestion Window Phase. The conducted simulations have demonstrated a performance boost of 10-15% when compared to TCP Reno and Westwood, and around 10-20% when there is a burst error, which corresponds to a discrete time first-order Markov model. A crucial element of this altered TCP protocol is that the cwnd ought to be adjusted to an ideal value for a specific connection. In order to do this, a productive bandwidth estimation algorithm would be created, which would dynamically calculate the fair share of a connection depending on a number of seen and measured criteria. Our goal is to design a function that, at the window recalculation phase, would dynamically calculate the cwnd.

VIII. REFERENCES

1) Congestion Avoidance and Control by Jacobson and M. J. Karels, Proceedings of ACM SIGCOMM, August 1988, vol. 18, pp. 314-329.
2) Jacobson's algorithm for modifying TCP congestion prevention. Sent from 1990 to the end2end-interest email list. [3] IEEE/ACM Transactions on Networking, December 1997; H. Balakrishnan et al.,
3) The ns-2 network simulator, UCB/LBNL/VINT, June 2004. nsnam/ns/http://www.isi.edu. [5] Fall and Varadhan, K., The NSM Manual, December 13, 2003, http://www.isi.edu/nsnam/ns/ns-documentation.html; Vol. 23, No. 2, February 2005, pp. 235-248; on Selected Areas in Communications.
4) C. Casetti et al., Kluwer Academic Publishers, TCP Westwood: End-to-End Bandwidth Estimation for Enhanced Transport over Wireless Links. 2002; Wireless Networks 8, 467–479 [7] Computer Communications xx (2003) 1–18, www.elsevier.com/locate/comcom; Gerla, B.K.F. Ng, M.Y. Sanadidi, M. Valla, R. Wang, TCP Westwood with adaptive bandwidth estimate to optimize efficiency/friendliness tradeoffs
5) Fluid-flow Analysis of TCP Westwood with RED, IEEE GLOBECOM 2003, J. Chen, F. Paganini, R. Wang, M.Y. Sanadidi, M. Gerla.
6) TCP With Sender-Side Intelligence to Handle Dynamic, Large, Leaky Pipes, R. Wang, K. Yamada, M. Y. Sanadidi, and M. Gerla, IEEE Journal on Selected Areas in Communications, Vol. 23, No. 2, Feb. 2005, pages 235-248.
7) TCP congestion control, Allman, M., Paxson, V., and Stevens, W.R. RFC 2581, April 1999. Improved End-to-End TCP Performance across Mobile Internets, R. Yavatkar and N. Bhagwat, Proceedings of Workshop on Mobile Computing Systems and Applications, December 1994.
8) I-TCP: Indirect TCP for Mobile Hosts, A. Bakre and B.R. Badrinath, International Conference on Distributed Computing Systems, 1995, pp 136–143.
IEE Proc. on Communications, Vol. 146, No. 4, Aug. 1999, pp. 222–230; N. K. G. Samaraweera, Non-Congestion Packet Loss Detection for TCP error recovery using wireless networks.
9) TCP Westwood modules for ns-2, Network Research Lab, Department of Computer Science, UCLA, http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.htmlImproved End-to-End TCP Performance across Mobile Internets, R. Yavatkar and N. Bhagwat, Proceedings of Workshop on Mobile Computing Systems and Applications, December 1994.

10) I-TCP: Indirect TCP for Mobile Hosts, A. Bakre and B.R. Badrinath, International Conference on Distributed Computing Systems, 1995, pp 136–143.

IEE Proc. on Communications, Vol. 146, No.

11) , Aug. 1999, pp. 222–230; N. K. G. Samaraweera, Non-Congestion Packet Loss Detection for TCP error recovery using wireless networks.

12) TCP Westwood modules for ns-2, Network Research Lab, Department of Computer Science, UCLA, http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.html