Int. J. Communications, Network and System Sciences, 2015, 8, 130-145
Published Online May 2015 in SciRes. http://www.scirp.org/journal/ijcns
http://dx.doi.org/10.4236/ijcns.2015.85015

Adaptive TCP: A Sender Side Mechanism with Dynamic Adjustment of Congestion Control Parameters for Performance Improvement in WLAN

Purvang Dalal[1], Mohanchur Sarkar[2], Kankar Dasgupta[3], Nikhil Kothari[1]

[1]Department of Electronics and Communication, D. D. University, Nadiad, India
[2]Space Application Center, ISRO, Ahmedabad, India
[3]Indian Institute of Science and Technology, Trivandrum, India
Email: pur_dalal.ec@ddu.ac.in, msarkar@sac.isro.gov.in, ksd@iist.ac.in, nil_kothari.ec@ddu.ac.in

Received 30 March 2015; accepted 4 May 2015; published 11 May 2015

Abstract

This paper presents a sender side only TCP mechanism to prevent compromise for bandwidth utilization in IEEE 802.11 wireless networks. In absence of mechanism for accurate and immediate loss discrimination, the TCP sender unnecessarily reduces its *Loss Window* in response to the packet losses due to transmission errors. At the same time, frequent transmission losses and associated link retransmissions cause inaccuracy for available bandwidth estimate. The proposal, Adaptive TCP tackles the above issues using two refinements. First, sender estimates the degree of congestion by exploiting the statistics for estimated *Round Trip Time* (RTT). With this, it prevents unnecessary shrinkage of *Loss Window* and bandwidth estimate. Second, by concluding the uninterrupted evolution of its sending rate in recent past, the Adaptive TCP advances bandwidth estimate under favorable network conditions. This in turn, facilitates for quick growth in TCP's sending rate after loss recovery and consequently alleviates bandwidth utilization. The authors implement the algorithm on top of TCP NewReno, evaluate and compare its performance with the wireless TCP variants deployed in current *Internet*. Through intensive simulations it is demonstrated that the Adaptive TCP outperforms other well-established TCP variants, and yields more than 100% of the throughput performance and more than 60% of improvement for bandwidth utilization, compared to TCP NewReno. The simulation results also demonstrated compatibility of Adaptive TCP in a shared wireless environment.

Keywords

Bandwidth Estimation, Round Trip Time, Congestion Estimator, Network Utilization Factor, Bit

How to cite this paper: Dalal, P., Sarkar, M., Dasgupta, K. and Kothari, N. (2015) Adaptive TCP: A Sender Side Mechanism with Dynamic Adjustment of Congestion Control Parameters for Performance Improvement in WLAN. *Int. J. Communications, Network and System Sciences*, **8**, 130-145. http://dx.doi.org/10.4236/ijcns.2015.85015

Error Rate, Fast Retransmit and Recovery

1. Introduction

Transmission Control Protocol (TCP) [1] is the most popular transport layer protocol used for reliable data transfer in *Internet* [2]. In absence of the support for admission control or resource reservation, the TCP sender takes the responsibility to recover from the packet drops; detected either on arrival of 3 *Duplicate Acknowledgments* (*DupAcks*) or on expiration of *Retransmission Timeout* (RTO) timer [1]. TCP's congestion control and loss recovery algorithms such as *Fast Retransmit & Recovery* (FRR) and *Timeout* unconditionally reduce the transmission rate by shrinking its *congestion window* (*cwnd*), based on the assumption that the packets are lost mainly due to the network congestion [1]. The size of *cwnd*, adjusted by TCP sender after loss recovery is referred as *Loss Window* in rest of the draft. Unfortunately, the above loss recovery mechanism is inappropriate in wireless networks where packet losses are mostly accounted for the reasons other than network congestion; like *abrupt delay variations* and *link transmission errors* [3] [4].

With FRR, conventional TCP sets (a) *Loss Window* and (b) *ssthresh* (that describes estimation of the bandwidth availability) to an arbitrary value (which is half of the current size of *cwnd*) [1]. When the packet loss is due to the non-congestion reason, this reduction using a stagnant rule is unfair [5]. TCP schemes with *Loss Differentiation Algorithms* (LDA) [6] [7] advocate for retaining of the aforesaid parameters after loss recovery. However, such approaches are incongruous particularly when loss discrimination is merely indicative and information about current network state is either ignored or not available [8]. Furthermore, in wireless networks, a RTO is inevitable (if not *spurious* [9] [10]), when either the fast retransmission of a lost packet fails to reach the destination or the burst of TCP acknowledgments (TCP-acks) is lost [11]. In absence of notification either from the receiver or from intermediate node, to detect and to differentiate the cause for above, TCP schemes react to such RTO with drastic reduction in *cwnd* and initiate *slow-start* process using a smallest value for *Loss Window*. RTOs caused by non-congestion events have been reported as one of the major factors for TCP performance degradation [12] [13] in wireless networks due to two reasons. First, with non-congestion RTO, the TCP sender resets *Loss Window* to a minimum size and hence loss recovery is followed by inferior transmission rate. The rate probing using the minimum size of *Loss Window* after RTO, leads to severe inefficiency in networks with large *Round Trip Time* (RTT) [14]. Second, the uncorrelated reduction in *ssthresh* causes quick termination of *slow-start* process and TCP sender prematurely enters into *congestion-avoidance* process. This prevents sender to obtain usable share of network capacity quickly [15].

Few approaches in improving the performance of TCP over wireless networks largely focus either on increasing the size of *Loss Window* speedily based on the approval from intermediate node [16] [17] or adjusting TCP's congestion control parameters after loss recovery (*i.e. Loss Window*, *ssthresh*) to an advance size based on network state or loss discrimination [6] [7] [18] [19]. Unfortunately, the efficiency of all above schemes is not guaranteed in a shared wireless environment due to the limited ability of a TCP sender in observing the network resource together with the fast changing network bandwidth availability [20], and the deployment issues in the *Internet*. The problem the authors tackle in this paper is how a particular connection can acquire healthier transmission rate and thus take advantage of the available capacity more quickly than endorsed by TCP's traditional congestion control algorithms. The major considerations behind the proposed approach are as follows:

a) In the beginning of a new connection, the *slow-start* process is employed with an objective to discover the usable network capacity quickly, whereas the *slow-start* process after RTO is employed to attain immediate network utilization up to the estimated network capacity after loss recovery [1]. Therefore, the *slow-start* process after non-congestion RTO using a minimum value of *Loss Window* is unfortunate [5]. The sender may be benefited using a larger size of *Loss Window* based on the estimation.

b) The wireless network is generally characterized with high delay variations and frequent losses [4]. It is not easy to set advance value for *ssthresh* based on the estimation [15], particularly when information gathering is limited to only the most recent values in use and all previous ones are simply discarded (though they can be valuable). When the changes in network characteristic are fast and inconsistent, the fair estimate may be obtained by concluding uninterrupted evolution of TCP's sending rate.

In this work, a sender side only mechanism is presented to avoid unnecessary shrinkage of *Loss Window* and

ssthresh; rationally based on the statistics for estimated RTT. Additionally, TCP sender with proposed mechanism distinguishes connection progression in congestive or non-congestive state and determines the size of *ssthresh* above the minimum value derived. The TCP NewReno is modified to instrument the above mechanism and the modified scheme is referred as Adaptive TCP in rest of the draft. The Adaptive TCP minimizes the penalty by savings of RTTs that a TCP NewReno would entail, even under favorable conditions. Besides, Adaptive TCP shows good convergence to usable network capacity in a shared wireless environment. It means that the throughput gain is realized by using the spare bandwidth effectively instead of starving from that utilized by others. Rest of the paper is organized as follows: In Section 2, various TCP approaches addressing the issues very similar to those stated earlier are reviewed and analyzed. Section 3 discusses the design of Adaptive TCP. Section 4, evaluates Adaptive TCP using extensive simulations in ns-2. Finally, Section 5 concludes this paper.

2. Related Work

There have been many solutions proposed for improving the performance of TCP in wireless networks. This section briefly summarizes the existing research related to the work done in this paper. The related work is classified into three parts. The first part depicts the solutions to retain TCP's congestion control parameters for non-congestion FRR/RTOs, the second part presents the solutions to enhance performance using advance value for TCP's congestion control parameters after loss recovery based on sender side estimate and the third part presents the solutions those facilitate rapid growth for *cwnd* with the support from intermediate routers.

2.1. Solutions to Retain Value for *cwnd/ssthresh*

TCP schemes like TCP-*Freeze* [21], LLE-TCP (*Link Layer ARQ Exploitation*) [22], ILC-TCP (*Inter Layer Collaboration*) [23], *semi*-TCP [24] etc. have been proposed to prevent inappropriate reduction in *Loss Window* using an *explicit* notification. Some TCP schemes equipped with *end-to-end* LDA [6] [25] also advocate for retaining of *cwnd* after wireless loss recovery. Unfortunately, despite of accurate loss notification, the performance of the above schemes is inadequate due to the various reasons as follows:

a) With RTO, TCP sender resets *Loss Window* and restarts evolution of *cwnd* from the smallest value using a RTT dependent *ack-clocking* mechanism. In wireless networks with high *Bit Error Rate* (BER), non-congestion RTOs are inevitable and frequent. With frequent RTOs, before TCP sender raises size of *cwnd* to a reasonable value, it is compelled to resets it to a smallest value. Therefore, the effective size of *cwnd* over the data transfer phase is found inadequate to attain suitable bandwidth share [8] and hence retaining of the same leads to insignificant improvement.

b) When BER is small, retaining of *cwnd* can potentially aggravate congestion in the network and cause heavy packet loss, both for itself and for other traffic in the network, especially if the old estimate of the rate is stale (due to sudden change in the network congestion) [5] [26].

Moreover, these schemes require support from intermediate or end nodes and hence deployment of such TCP schemes in existing network setup is not always feasible. Several TCP schemes (e.g. *Eifel* [9], F-RTO [10] etc.) have been proposed to tackle TCP performance issues on account of spurious RTOs. Like LDA schemes, the common problem of the above schemes is that they paid little attention on adjusting the transmission rate appropriately when spurious FRRs/RTOs are detected. In fact with loss recovery, immediate controlling of the transmission rate using a fair mechanism is critical for improving TCP's performance.

2.2. Solutions with Advance Value for *cwnd/ssthresh*

TCP schemes with *implicit* estimation techniques (e.g. TCP Westwood (TCPW) [27], *Prairie* [28], JTCP [29] and TCP Veno [30]) exploit information during successive RTT measurements for either *Bandwidth Estimation* or *Loss Discrimination*, and accordingly adjust TCP's congestion control parameters after loss recovery to utilize the available bandwidth efficiently. In wireless networks, the sender side estimate for RTT and the *ack-arrival* time may fluctuate greatly due to various factors like transmission errors, channel contention and frequent route re-establishments [4] and it may not correctly infer to the network congestion all the time. Other factors such as TCP *coarse-grained* clocks, *ack-clustering* and *ack-compression*, *delayed acknowledgments* and route asymmetry between forward and reverse path also pose challenges to the accuracy for *Bandwidth Estimation* [31]. With this constraint, network estimate based on the statistics obtained for a period limited by RTT and uti-

lizing the same for adjusting TCP's congestion control parameters during loss recovery is indecorous. Research carried in [32] have shown that, the direct use of the dispersion of long packet trains for available bandwidth measure may cause misleading estimation and therefore undesirable performance.

TCPW and *Prairie* dynamically estimate the available bandwidth per RTT (which may be estimated incorrectly), and adjust the transmission rate by updating *ssthresh* and *Loss Window* to the larger values in proportion to the estimated available bandwidth. Unfortunately, updating the above parameters to large values based on the estimate during a small period of RTT does not guarantee for improvement of TCP's performance [32]. If the loss rate due to wireless errors is high, a TCP sender with large values for TCP state variables has high possibility to have frequent burst losses and long *go-back-N* retransmissions [5]. Besides, the inaccuracy for estimation the common problem is that, all above techniques paid no attention on revising the *Loss Window* when RTO's are triggered regardless of congestion [8]. This in turn, inappropriately decelerates *slow-start* process after RTO, due to minimum *Loss Window*.

2.3. Router Assisted Solutions with Rapid Growth of *cwnd*

TCP Fast TCP [16], *Quick-Start* [17], etc. advocate for superior growth in *cwnd* during *slow-start* process based on the approval from the intermediate routers. Since, the above proposals require support from all routers along the path, this could present a high bar to deployment in the *Internet* [32] Moreover, it can be difficult for the TCP sender to determine how much data will ultimately be transmitted and therefore to form a reasonable rate request. Above all, the above proposals are not expected to be of use in the case of a highly-utilized path [17]. Few TCP schemes described in [33] propose use of arbitrary high value for an *Initial Window* to enhance TCP's start-up performance. On a downside, the arbitrary large value has an adverse effect on the performance of long-lived TCP flows in a shared wireless environment [14] [15]. It is important to point out that they follow conventional FRR and hence inherit TCP's weakness for wireless loss recovery.

In summary, further efforts are essential in the direction to provide a judicious value for TCP's congestion control parameters with loss recovery that maximizes the bandwidth utilization by a long lived TCP connection when in isolation and at least protects network performance in a shared network environment. It is worth to mention here that the high speed scheme like TCP CUBIC [34] is primarily designed for large *Bandwidth Delay Product* (BDP) networks, not for the wireless networks with relatively small datarate. As reported in [35], random physical and MAC layer artifacts (e.g. *multi-path, interferences*) introduced packet losses can cause performance degradation for CUBIC. Hence, authors have not considered such TCP schemes either for related work or for performance comparison.

3. Proposed Scheme

In wireless networks, measuring accurately the available share of network bandwidth by a TCP sender is known to be difficult at transport layer. Unlike conventional TCP, which halves the *cwnd* during FRR and resets it after *Timeout*, the Adaptive TCP adjusts *Loss Window* and *ssthresh* to suitable values by exploiting the statistics for estimated RTT and by concluding the uninterrupted evolution of TCP's sending rate. First of all, the Adaptive TCP revises *Loss Window* based on the network congestion level estimated prior to the loss detection. It also updates *ssthresh* in accordance to the revised *Loss Window* particularly after FRR. The above mechanism is detailed in Section 3.1. Additionally under favorable network conditions, the Adaptive TCP advances the size of *ssthresh* as described in Section 3.2.

3.1. Reworking for the Size of *Loss Window*

The Adaptive TCP sender uses a *congestion estimator* (α), which reflects the degree of network congestion. The value of α is determined dynamically using Equation (1). As shown in **Figure 1**, it varies from 0 to 1 based on the estimated value of RTT prior to loss detection (RTT_n) and variations in estimated RTT (*i.e.* minimum value of estimated RTT ($RTT_{\min}$) and maximum value of RTT ($RTT_{\max}$)) from the time when the data transfer is initiated.

$$\alpha = 1 - \frac{RTT_n - RTT_{\min}}{RTT_{\max} - RTT_{\min}} \tag{1}$$

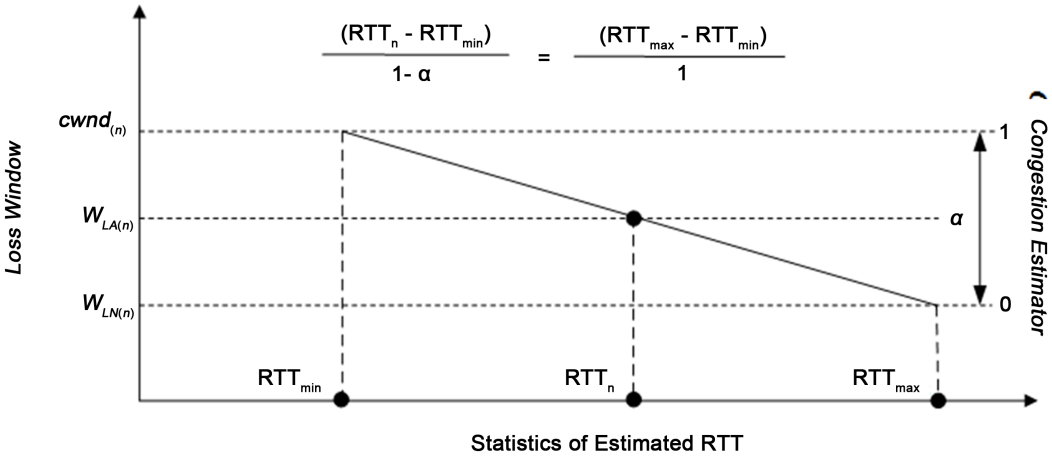

Figure 1. Congestion estimator (α) & loss window in adaptive TCP.

Adaptive TCP sender utilizes the above value of α and adjusts *Loss Window* after n^{th} loss recovery (say $W_{LA(n)}$), using Equation (2). When the estimated RTT_n is equal to RTT_{min}, it represents the congestion-free state of the network (*i.e.* α is set to 1) and like other LDA schemes, Adaptive TCP adjusts *Loss Window* to a size that corresponds to the maximum value of *cwnd* exercised prior to the loss detection. On the other side, when RTT_n approaches to RTT_{max}, it represents the maximum congestion in the network since inception of the data transfer (*i.e.* α is set to 0). In this situation, Adaptive TCP sets size of *Loss Window* using the congestion control mechanism of native TCP.

$$W_{LA(n)} = (\alpha) \times cwnd_{(n)} + (1-\alpha) \times W_{LN(n)} \tag{2}$$

Here, $cwnd_{(n)}$ represents the maximum size of *cwnd* that a sender exercised prior to the n^{th} loss detection. $W_{LN(n)}$ is the size of *Loss Window*, which a native TCP sender will have after n^{th} loss recovery action. Equation (2) itself concludes that the Adaptive TCP sender never advances the size of *Loss Window* beyond the size of $cwnd_{(n)}$. Thus, Adaptive TCP protects conservative approach of TCP NewReno, which is essential for maintaining *Fairness*. In a network condition other than extreme congestion, the Adaptive TCP sender acquires higher value for W_{LA} and hence requires less number of RTTs to attain usable network capacity after loss recovery.

It should be noted that, the proposed scheme also facilitates for higher size of *Loss Window* after *Timeout*. However, in case of serial timeouts (with exponential *back-off* for RTO), the Adaptive TCP sender follows the native approach and resets *Loss Window* to a minimum size. This is mainly due to irrelevance of RTT_n in determining network congestion after a significant delay of more than RTO. The Adaptive TCP regulates the size of *ssthresh* (say $ssthresh'_{(n)}$) after FRR and *Timeout* using Equation (3) and Equation (4), respectively.

$$ssthresh'_{(n)} = W_{LA(n)} \tag{3}$$

$$ssthresh'_{(n)} = \frac{cwnd_{(n)}}{2} \tag{4}$$

Note that, when TCP detects a non-congestion packet loss, the maximum *cwnd* exercised before the occurrence of loss, need not have converged to the optimal capacity of the network. In this situation, TCP falsely reduces *ssthresh* based on the incomplete result of probing. This reduces the upper limit for the *slow-start* process or initiates *congestion-avoidance* process to probe gently for the extra bandwidth, despite of having availability of sufficient bandwidth. The above prohibits the sender from immediate utilization of the bandwidth to its full usable capacity. In order to compensate this gap quickly, the Adaptive TCP monitors the uninterrupted evolution of the *cwnd* during a data transfer slot prior to the loss detection. Higher rate of growth in *cwnd* in comparison to its precursor data transfer slot indicates favorable network condition. In the next subsection, the second refinement of the algorithm is presented. Using the second refinement, the Adaptive TCP adopts the minimum value of *ssthresh* under favorable network condition and advances it further to support the larger increments in the *cwnd* per RTT, after loss recovery.

3.2. Advancement for *ssthresh*

Different from the other mechanisms, Adaptive TCP envisages data transfer into discrete slots; wherein each slot is confined to the interval between two successive loss events at TCP sender. During each slot, the TCP sender distinguishes connection progression either in congestive or non-congestive state. For that, Adaptive TCP computes the *Network Utilization Factor* ($K_{(n)}$), which characterizes the uninterrupted growth of *cwnd* during n^{th} data transfer slot between $(n-1)^{th}$ and n^{th} loss events (either FRR or RTO). On triggering of n^{th} loss event at time $T_{(n)}$, the Adaptive TCP sender computes ($K_{(n)}$) using Equation (5).

$$K_{(n)} = \frac{cwnd_{(n)} - W_{LA(n-1)}}{T_{(n)} - T_{(n-1)}} \tag{5}$$

Here, $W_{LA(n-1)}$ is the *Loss Window* after $(n-1)^{th}$ loss recovery event. In fact, the *Network Utilization Factor* reflects the *ack-arrival* rate between consecutive loss events and like TCPW, provides a base for bandwidth estimation. On a better side, the impact of abrupt RTT variations is averaged out here, as sender monitors the *cwnd* progression over an entire data transfer slot. The Adaptive TCP sender discovers the direction of network congestion after n^{th} data transfer slot by comparing $K_{(n)}$ with $K_{(n-1)}$. When $K_{(n)}$ is higher than $K_{(n-1)}$, it indicates lessening of congestion in the network. The authors believe this as favorable network condition and realize a scope for additional quick rise in *cwnd* after loss recovery. In this incident, the Adaptive TCP respects previous estimate of *ssthresh* ($ssthresh_{(n-1)}$) and advocates for using minimum size of *ssthresh* (referred as $ssthresh_{(adv)}$), which is computed using Equation (6).

$$ssthresh_{(adv)} = ssthresh_{(n-1)} + K_{(n-1)} \times \left(T_{(n)} - T_{(n-1)}\right) \tag{6}$$

As illustrated in **Figure 2**, the value for $ssthresh_{(adv)}$ is computed at point E, by extending growth of $ssthresh_{(n-1)}$ from point D using the *Network Utilization Factor* $K_{(n-1)}$. The Adaptive TCP sender, adjusts $ssthresh_{(n)}$ using the concluding size, which is determined as maximum of $ssthresh_{(adv)}$ and $ssthresh'_{(n)}$ suggested using Equation (3) or Equation (4), earlier. It must be noted that the above additional advancement is applicable only under favorable condition, *i.e.* when $K_{(n)} > K_{(n-1)}$. Therefore, any kind of over estimation for $ssthresh_{(n)}$ is ruled out. As the sender has witnessed higher transmission rate (lessening of congestion) during recent data transfer slot (n^{th}) compared to that observed during its precursor slot ($(n-1)^{th}$), any advancement in the size of $ssthresh_{(n)}$ to make up for $ssthresh_{(adv)}$ is legitimate.

The advancement in *ssthresh* is beneficial particularly with frequent transmission losses. Whenever, Adaptive TCP sender advances $ssthresh_{(n)}$ further (above the size of $W_{LA(n)}$ using the mechanism described in Section 3.1), it resumes transmission using *slow-start* process, even after FRR. This leads to rapid growth of *cwnd* and as a consequence immediate network utilization to its usable capacity. As shown in **Figure 2**, the size of *ssthresh* is advan-

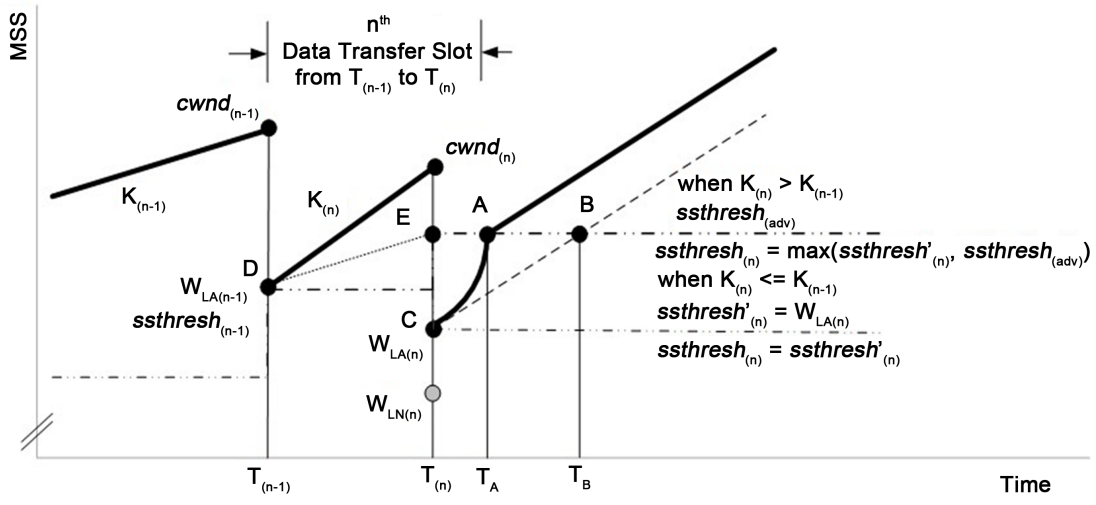

Figure 2. Advancement for *ssthresh*.

ced to a value corresponds to point E instead of that corresponds to point C. This facilitates exponential growth for *cwnd* up to point A at time T_A. Without this advancement, sender TCP increases *cwnd* to an identical size of $ssthresh_{(adv)}$ at time T_B. Since the advance value of *ssthresh* is derived judiciously, it rules out burstiness during *slow-start* process in a shared wireless environment and hence adverse effect on TCP performance.

Note that the proposal will not be energized when the network is congested, *i.e.* α is set to 0 with extreme congestion. And additional advancement for *ssthresh* is applicable only when $K_{(n)} > K_{(n-1)}$. Thereby, Adaptive TCP puts up a sincere effort for performance improvement along with aptitude for protecting the performance in a shared wireless environment. The following Section explores various facets of the Adaptive TCP and discusses the performance evaluation of Adaptive TCP in WLAN.

4. Performance Evaluation

The performance of the proposed Adaptive TCP scheme is evaluated in a series of *ns*-2 simulations. **Figure 3** shows the infrastructure based WLAN setup used to obtain the results. All wireless nodes are connected to the wired backbone through the designated wireless *Access Point* (AP). Latency D and datarate presented in **Figure 3**, emulate the presence of the wired backbone network. It is assumed that, the transmission over wired component of the network is free from corruption losses. The wireless channel between wireless nodes and designated AP is based on the IEEE 802.11 g standard with default physical layer parameters and operates at 11 Mbps. During simulations, wireless node initiates file transfer with the TCP receiver located in the wired backbone. The receiver issues a TCP-ack for every data packet received. TCP window size is measured in number of packets with uniform size of 1000 bytes per packet. Receiver's *advertised window* is always large to keep the sending window equal to *cwnd*. The *ns*-2 module of a TCP NewReno is extended to model the proposed transport layer functionality at the sender, with all other parameters set to default.

During all experiments, the simulation duration is kept as 500 sec (unless specified), which is long enough for examining steady state bandwidth (BW) utilization in the network. Transmission losses over an IEEE 802.11 link are modeled using a well-established 2 *State Markov Model* [36]. For demonstrating the effectiveness of proposed mechanism, performance of Adaptive TCP is evaluated using three distinct scenarios.

a) First set of simulations are performed to substantiate impact of adaptation on TCP's congestion control parameters immediately after loss recovery, *i.e. Loss Window* and *ssthresh*. Here, the behavior of the Adaptive TCP sender is analyzed in a dedicated erroneous WLAN and compared with the behavior of TCP NewReno and TCPW. Both are the well-recognized TCP variants, deployed in today's heterogeneous *Internet* [35].
b) In the second set of experiments, the proposed Adaptive TCP is subjected to different network conditions by varying wireless link errors on wireless network segment along with delay and datarate of the wireline network segment.
c) Third set of simulations examine the compatibility of proposed scheme in a terrestrial heterogeneous network environment, which is susceptible to both types of packet losses, *i.e.* transmission and congestion. The fairness of the proposed scheme is also evaluated.

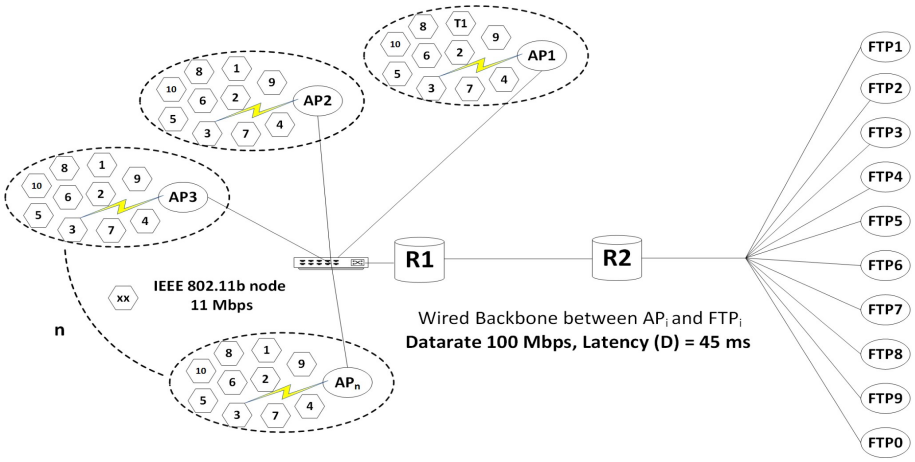

Figure 3. Network topology for simulations.

4.1. Functional Validation

During simulations, FTP data is transferred between T1 and FTP1 in presence of errors over an IEEE 802.11 link (**Figure 3**). The wireless sender T1 has instigated data transfer from 100 sec to 200 sec of simulation duration. **Figure 4** characterizes the behavior of Adaptive TCP with 1% of *Frame Error Rate* (FER), using α and the resulting *Loss Window*. It is seen that, the Adaptive TCP dynamically adjusted *Loss Window* in accordance to the computed value of α. When estimated value of RTT prior to the loss detection is identical to RTT_{max}, the computed value for α is 0. The above is inferred as sizeable congestion in the network and sender followed reduction in *Loss Window* as per the native scheme. However, when estimated RTT prior to the loss detection is below RTT_{max}, the computed value for α is above 0 (Equation (1)). In this situation, Adaptive TCP did not reduce *Loss Window* using a stagnant rule, rather adopted larger *Loss Window* following Equation (2). This facilitated the Adaptive TCP sender for resuming its transmissions using a much better opening of *cwnd* immediately after loss recovery, which is essential for utilizing network to its usable capacity quickly.

As mentioned earlier in Section 3.1, the Adaptive TCP sender has also estimated larger value of *ssthresh* along with *Loss Window*, particularly when loss is detected on arrival of 3 *DupAcks*. Above that, the authors also noticed an interesting conduct of the proposed scheme at 164.46 sec. As shown in **Figure 5**, the Adaptive TCP sender detected a non-congestion packet loss on arrival of 3rd *DupAck* at 164.23 sec and resumed transmission at 164.46 sec using a larger size of *Loss Window* (10 MSS, point C). At the same time, the Adaptive TCP sender has computed the minimum size for *ssthresh* as 12 MSS (point E). In fact the $ssthresh_{(adv)}$ is computed as 12.79 (which is truncated to an integer size of 12 MSS) by extending growth of $ssthresh_{(n-1)}$ from D to E using the *Network Utilization Factor* $K_{(n-1)}$. This additional rise for *ssthresh* above *Loss Window* is attributed to the

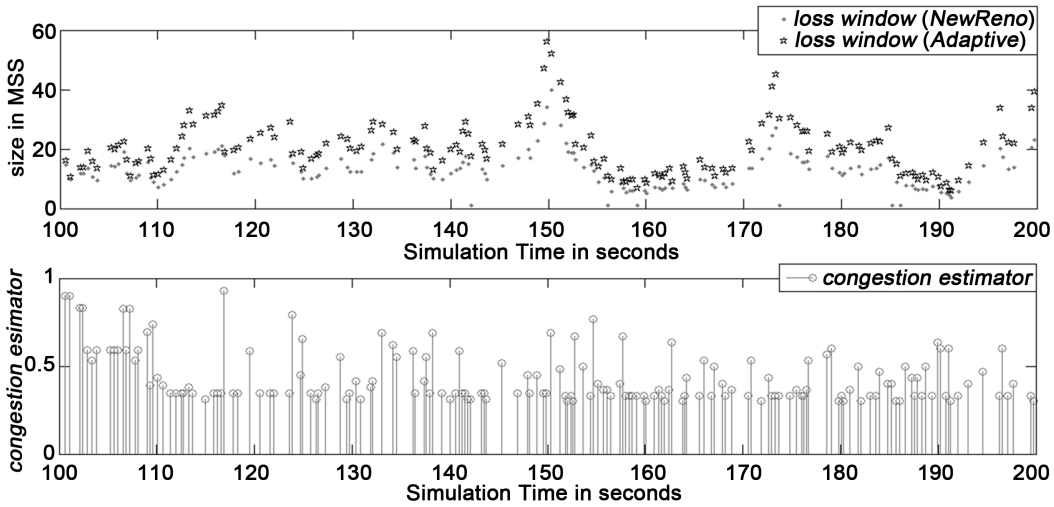

Figure 4. Congestion estimator (α) & loss window in Adaptive TCP (FER = 1%).

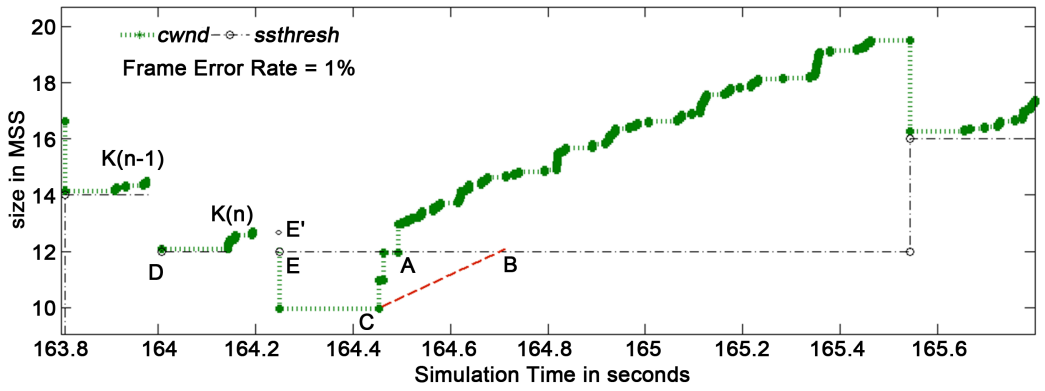

Figure 5. Advancement for *ssthresh* in Adaptive TCP (FER = 1%).

better network condition experienced by Adaptive TCP sender during data transfer in recent past (refer Section 3.2) compared to its precursor phase (*i.e.* $K_{(n)} > K_{(n-1)}$). The above elevated *cwnd* from 10 MSS to 12 MSS in a short time (from C to A in **Figure 5**) using the *slow-start* process after loss recovery and saved almost 2 RTTs in comparison with traditional *congestion-avoidance* process (that would have caused larger delay for identical growth in *cwnd* from C to B).

It is clearly visible from **Figure 6** that the along with *Loss Window*, Adaptive TCP has also estimated larger value of *ssthresh* to enrich sending rate due to the reasons as follow:

a) With the larger size of *Loss Window*, TCP sender resumed its transmission with higher value of sending rate.
b) The higher estimate of *ssthresh* stretched *slow-start* response after RTO to facilitate exponential growth in *cwnd* up to a much higher value.
c) Advanced value of *ssthresh* after FRR provided immediate growth of *cwnd* as per *slow-start* mechanism, which compensated for any inferior estimate of *Loss Window* as an outcome of conservative approach.

Figure 7, compares the *ssthresh* evolution between TCPW and Adaptive TCP. In a dedicated WLAN with transmission losses, a much higher estimate for *ssthresh* is observed using the Adaptive TCP, though it uses a conservative approach for advancement. This would in turn facilitate additional performance improvement over TCPW, as presented in the next Section.

4.2. Bandwidth Utilization

TCP's BW utilization in a WLAN is mainly compromised due to inappropriate setting of its congestion control parameters after loss recovery (Section 1). The Adaptive TCP sender prevents undue reduction in *Loss Window* and *ssthresh* based on adaptation. As a consequence, it obtains healthier transmission rate immediately after loss recovery. To demonstrate, the Adaptive TCP is evaluated with an objective to elucidate the impact of link errors

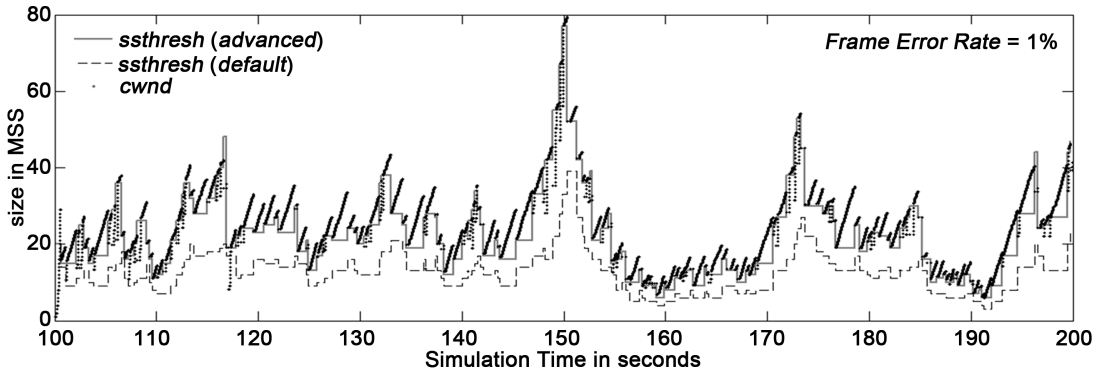

Figure 6. *ssthresh* and *cwnd* evolution in Adaptive TCP (FER = 1%).

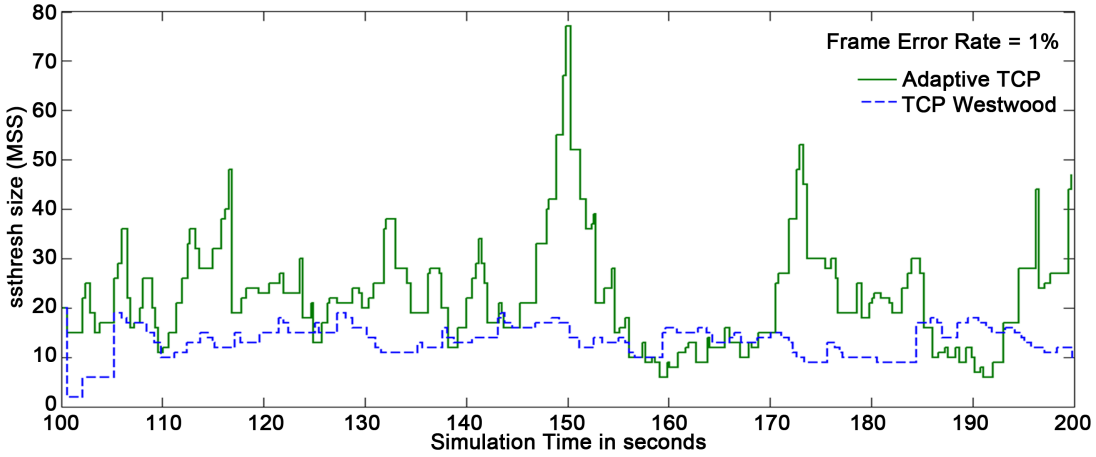

Figure 7. *ssthresh* comparison between TCPW and Adaptive TCP (FER = 1%).

on TCP's BW utilization in WLAN. In order to establish superiority of Adaptive TCP, its BW utilization is compared with the one achieved using the other wireless TCP variants deployed in the current *Internet*. For majority of experiments discussed in this section, a single TCP connection (unless specified) supporting FTP session between a wireless node T1 and wired node FTP1 (**Figure 3**) is analyzed.

4.2.1. Impact of Random Errors

In WLAN, IEEE 802.11 layer employs local retransmission over a *point-to-point* link for loss recovery, which may cause additional rise for estimated RTT in absence of network congestion [37]. During simulations in a dedicated WLAN, large number of link retransmissions is observed with increase in FER and as a consequence higher RTT estimate at sender is witnessed. The above adversely affected on TCP's responsiveness, in general.

Figure 8, represents comparison for BW utilization by TCP NewReno, Adaptive TCP and TCPW, as FER varies from 0% to 10%. It is observed that, the BW utilization by Adaptive TCP matches to that attained by other TCP variants, when error rate is 0%. However, all TCP schemes showed degradation for aggregate BW utilization with increase in FER. This is mainly due to needless shrinkage of *cwnd/ssthresh* at sender on account of frequent losses and sinking of TCP's responsiveness on account of additional rise for RTT in view of link retransmissions. Both of the above factors furnished significant rise to the unused network BW, referred as residual BW in the rest of the analysis. Using the proposed refinements, Adaptive TCP avoided compromise in the size of *Loss Window* and *ssthresh* particularly with wireless packet losses and hence its performance is benefited by healthier sending rate after loss recovery. As shown in **Figure 8**, the Adaptive TCP demonstrated much higher BW utilization relative to that acquired by other TCP schemes for an identical value of FER. It is clearly visible that the Adaptive TCP showed about 100% of throughput improvement over TCP NewReno and TCPW for the error rates of 0.5% and 1%. In **Figure 8**, the statistics for reduction in residual BW are presented in addition to the appraisal for BW utilization amongst TCP variants. As indicated, the Adaptive TCP exhibited significant reduction for the residual BW over other TCP schemes in comparison.

During the simulations in the stated network scenario, the maximum BW utilization is observed as 5.99 Mbps with 0% FER, which is considered as reference for deriving the above statistics of residual BW. To explain further, BW utilized by TCP NewReno, TCPW and Adaptive TCP is recorded as 2.92 Mbps, 3.81 Mbps and 4.78 Mbps respectively with 0.1% of FER. It means that the Adaptive TCP has less residual BW (1.21 Mbps) compared to that seen with TCPW (2.18 Mbps) and TCP NewReno (3.07 Mbps). This turned out as 66% and 47% of improvement for residual BW by Adaptive TCP over TCP NewReno and TCPW respectively. Since, the Adaptive TCP also uses RTT dependent growth in *cwnd*, the improvement for BW utilization diminishes with increase in FER due to increase in RTT (*slow convergence*) in view of link retransmissions. This in turn reflected into a reduced amount of improvement for residual BW with increase in FER.

4.2.2. Impact of Wireline Delay

The objective is to evaluate the effectiveness of the proposed scheme for different latency of wired backbone. The impact of the above variations (in addition to the errors in the wireless component) on the efficiency of the

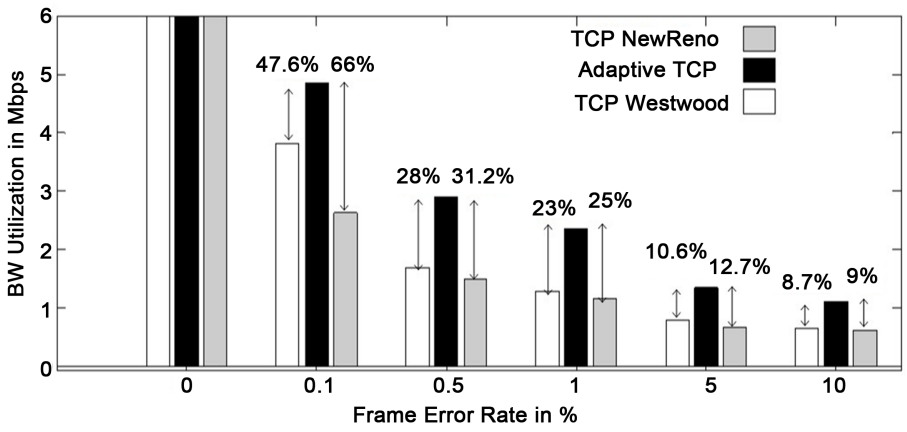

Figure 8. Impact of FER (single flow with default environment settings).

Adaptive TCP in WLAN is examined through simulations. As reported earlier in Section 4.2.1, BW utilization by the Adaptive TCP is at par with other TCP scheme; in WLAN environment with 0% FER. During simulations, the Adaptive TCP outperformed other TCP variants in terms of BW utilization, for all combinations of wireline delay (latency D varies from 15 ms to 75 ms) and FER (varies from 0.1% to 10%). To support the claim, the comparison for BW utilization by different TCP variants is presented in **Figure 9**. The analysis is presented for different values of wireline delay in presence of a fixed FER of 1%. The **Figure 9** additionally provides the statistics for reduction in residual BW by Adaptive TCP over other TCP schemes. It is obvious that, with the increase in wireline delay the RTT in the network also increases for the identical value of FER. Consequently, the TCP performance is additionally sacrificed on account of *slow convergence* to usable network capacity after loss recovery. All TCP schemes showed identical behavior and as seen from **Figure 9**, BW utilization by all TCP variants diminish with increase in the wireline delay in WLAN (with identical FER of 1%).

Among all chosen TCP variants, BW utilization by TCP NewReno is inferior and mainly accounted for undue reduction of *Loss Window* and *ssthresh* after wireless loss recovery. Even so, Adaptive TCP and TCPW showed much better performance. The Adaptive TCP has an upper edge for BW utilization mainly due to having the advance value for *ssthresh* under favorable network conditions. This facilitated for rapid growth in *cwnd* by permitting *slow-start*, even after FRR; which TCPW doesn't have. Moreover, Adaptive TCP has larger size of *Loss Window* after *Timeout* and consequently it requires less number of RTTs to complete *slow-start* process for an identical size of *ssthresh*.

4.2.3. Impact of Wireline Datarate

The Adaptive TCP is further evaluated with an objective to validate its effectiveness in presence of different datarate in the wired backbone. The impact of the above variations (in addition to the wireless transmission errors), is examined on the efficiency for BW utilization by Adaptive TCP. It should be noted that, in the stated network scenario, a single AP with 11 Mbps of link can deliver maximum of 6.6 Mbps of TCP traffic over the backbone [38]. In this context, with any value of wireline datarate below 6 Mbps, bottleneck in the network is transferred to the wired backbone and it limits the maximum throughput in the network. On the other side, with a larger datarate on wireline backbone, bottleneck is transferred to the wireless component. In this scenario, the maximum traffic delivered on the backbone is limited to the maximum datarate supported by AP. In order to demonstrate the efficiency of Adaptive TCP in both scenarios, the analysis is presented with two different values for wireline datarate; *i.e.* 5 Mbps & 100 Mbps, falling into each category. **Figure 10** shows comparison of BW utilization between Adaptive TCP, TCP NewReno and TCPW, for different datarate of the wired backbone (5 Mbps & 100 Mbps). The analysis is presented for the entire range of FER. A superior BW utilization by the Adaptive TCP is observed in presence of non-zero value of FER due to the proposed refinements. Importantly, irrespective of the location of bottleneck in WLAN, the Adaptive TCP performed better in presence of link errors and it also protected performance of the native TCP scheme in absence of link errors.

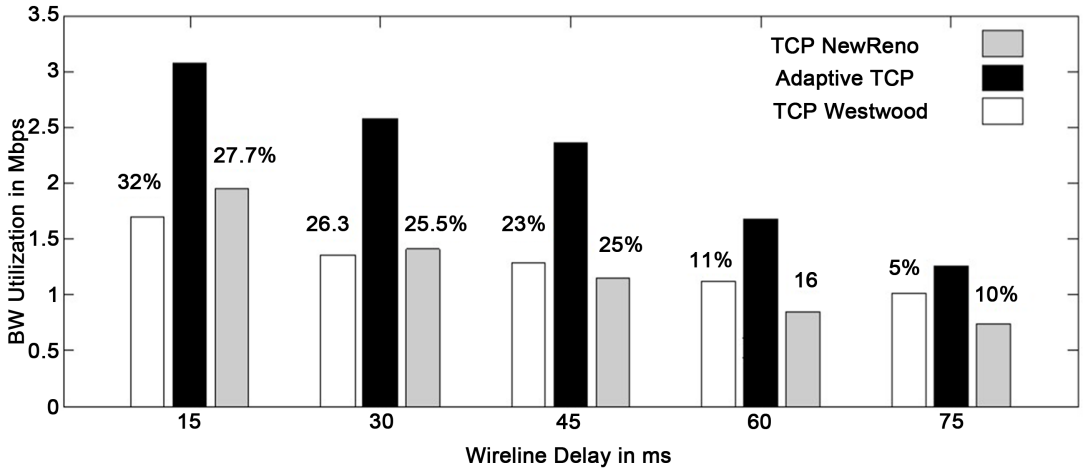

Figure 9. Impact of wireline delay (single flow, FER = 1%).

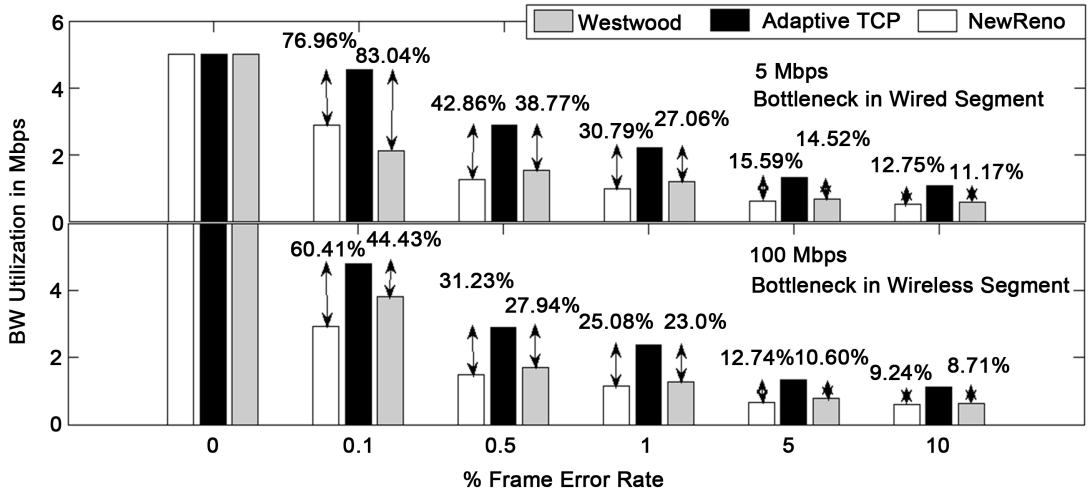

Figure 10. Impact of wireline datarate on BW utilization.

It is worth to mention that the BW utilization by TCPW is inferior to that achieved with TCP NewReno, when wired datarate is 5 Mbps and link transmission errors are reasonably low (0.1% FER). This shows inefficiency of TCPW in a WLAN with low capacity backbone. In fact, TCPW misjudged the available network bandwidth and over-estimated the *cwnd* that leads to more number of Go-Back-N TCP retransmissions with frequent losses [5]. With 0.1% FER, about 30% of additional TCP retransmissions for TCPW are observed in comparison to those seen with Adaptive TCP. The performance of Adaptive TCP is benefited mainly due to the conservative approach that reduces the Go-Back-N TCP retransmissions due to *cwnd* over-estimation. The statistics for %reduction in residual BW by Adaptive TCP over TCPW and TCP NewReno is also indicated in the same **Figure 10** (above respective bar). With 5 Mbps wireline datarate and 0% FER, the maximum BW availability for a single TCP flow is limited to 5 Mbps. In contrast, with increase in wired datarate the maximum BW availability is endorsed by utmost output of AP (during simulations 5.99 Mbps with 0% FER is observed). Since, the statistics for % of residual BW are derived with reference to the maximum BW availability, the effective value for residual BW is lessen when maximum BW availability is less. Therefore, for a specific FER, improvement for residual BW using Adaptive TCP is seen larger with 5 Mbps compared to that observed with 100 Mbps of wireline delay.

4.2.4. Impact of Self-Similar Traffic

The performance of the Adaptive TCP is evaluated further in presence of competing TCP flows in two distinct scenarios. In first scenario, the TCP traffic over the wireline backbone is realized using four active TCP flows, each from independent AP (refer **Figure 3**). In this scenario, TCP flows are competing on a wired backbone only. Here the objective is to validate effectiveness of the Adaptive TCP in presence of different level of congestion in the backbone and transmission errors on the wireless link. In second scenario, wireless home networks also have competing TCP flows and packet drops are accounted for the congestion in wireless component of the network in addition to the transmission errors. In both the scenarios, with the increase in number of active APs, a much higher utilization of backbone capacity is observed. In first scenario, it is observed that the operative TCP traffic delivered by respective AP is reduced with increase in the errors in IEEE 802.11 segment of network. Consequently, as seen from **Figure 11**, the overall network BW utilization also degraded with increase in FER. Due to the proposed refinements for preventing undue reduction of *Loss Window* and *ssthresh*, Adaptive TCP demonstrated relative improvement in BW utilization. It is worth to mention that the improvement in BW utilization by Adaptive TCP is lowered with increase in FER due to the inherited weakness (*slow convergence* to the usable capacity on account of additional rise in RTT in view of link retransmissions).

In second scenario, the simulations are performed with 4816 and 32 number of TCP flows; those are uniformly distributed amongst 4 active APs. In all cases, Adaptive TCP outperformed TCP NewReno and TCPW. **Figure 11** also shows statistics for % reduction in residual BW with variations in FER and total number of TCP flows. From the analysis of simulation results, the following inferences are drawn.

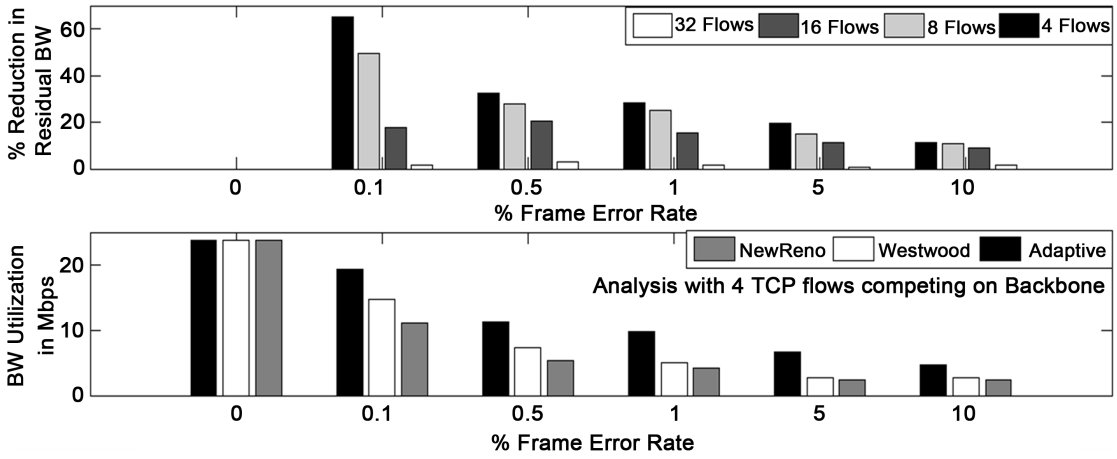

Figure 11. Analysis with multiple flows.

a) With increase in the FER, estimated RTT is increased at sender. Therefore, Adaptive TCP took significant time before raising *cwnd* to a large value. Moreover, with the frequent losses, the Adaptive TCP is required to revise its *Loss Window* before it acquired sufficiently large *cwnd*. Since, the revision for *Loss Window* is based on the conservative approach (*i.e.* the revised *Loss Window* can never be higher than the maximum *cwnd* seen before loss detection), the additional improvement for BW utilization using Adaptive TCP is lowered with increase in FER.

b) With the increase in the total number of TCP flows under the same AP, the effect of transmission errors on overall BW utilization is decreased, *i.e.* compromise for BW usage by one TCP flow due to wireless error is compensated by the other TCP flows in the vicinity. This caused less reduction in residual BW with increase in the number of TCP flows in the network for the identical FER. Subsequently, it has reduced the scope for % improvement in residual BW with the stated situation.

Fairness

The results are also analyzed to observe the impact of advancement on the fairness of Adaptive TCP. The fairness in this experiment is computed using the *Jain's Fairness Index* (JFI) [39], which is defined as per Equation (6).

$$JFI = \left[\sum_{i=1}^{n} x_i\right]^2 \Big/ n\sum_{i=1}^{n} x_i^2 \tag{6}$$

where, n is the number of TCP flows sharing the end-to-end path and xi denotes the good put achieved by it flow. The index ranges between 0 and 1. An index of 1 corresponds to the best fairness achievable between competing flows, whereas an index of $(1/n)$ denotes the case where a single flow acquires the entire bandwidth available. **Figure 12** shows the comparison for JFI. The results clearly indicate that the Adaptive TCP doesn't compromise with fairness for the chosen range of FER and competing TCP flows. It means that the throughput gain is realized by using the spare bandwidth effectively and not by starving the same from other TCP flows. It is believed as the most encouraging feature of the proposed scheme.

5. Conclusions

In this paper, a novel, sender side only approach is presented for improving the performance of TCP in WLAN. The performance improvement is achieved by revising TCP's congestion control parameters, *i.e. Loss Window* and *ssthresh*, after loss recovery. Our proposal has been evaluated by extensive simulations for its implementation over TCP NewReno and also compared with a well-known sibling TCP Westwood. Our analysis from the simulation results draws the following conclusions.

a) Using the computed value of *congestion estimator*, the Adaptive TCP sender prevents compromise of *Loss Window* and *ssthresh* parameters on account of non-congestion events. With this corrected parameters, the Adaptive TCP sender resumes transmissions after loss recovery with a healthier sending rate.

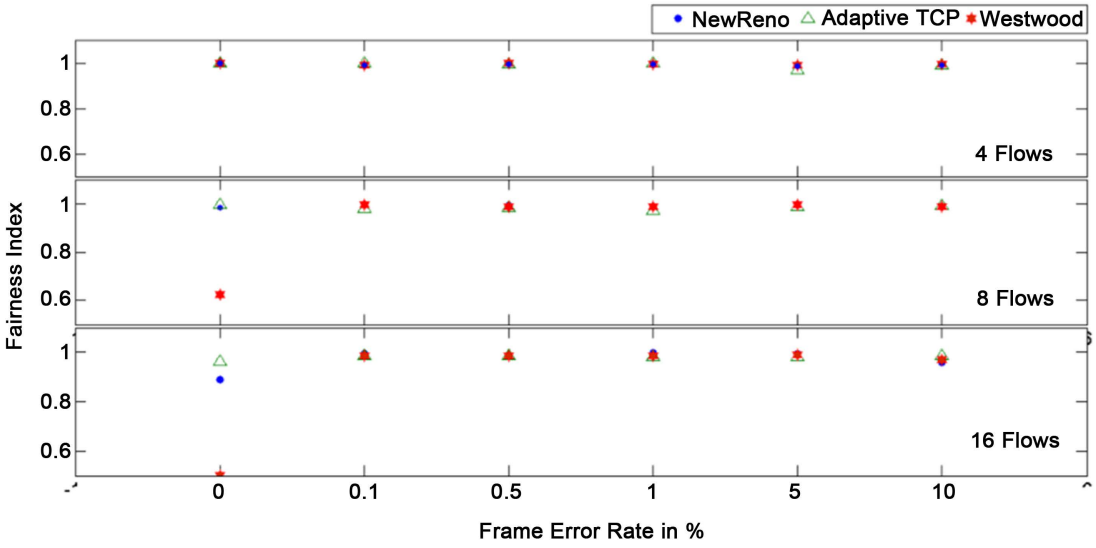

Figure 12. Impact on fairness.

b) Under favorable network conditions, Adaptive TCP sender adopts the minimum value of *ssthresh*, which is derived by extending previous *ssthresh* estimate using the previous value of *Network Utilization Factor*. This in turn creates a room for executing *slow-start* process even in absence of *Timeout* in order to achieve a higher sending rate. This is also beneficial after RTO, as it extends *slow-start* to a much higher value of *congestion window*.

c) Adaptive TCP demonstrated more than 100% of improvement over TCP NewReno in a non-congested network scenario with transmission losses. In terms of reduction in residual bandwidth, the achievement is above 60%.

d) As the changes are proposed only for preventing compromise in *Loss Window*, Adaptive TCP never advances the *Loss Window* beyond the size of *congestion window* seen prior to the loss detection. Due to this conservative approach, it protects TCP performance in a shared wireless environment, which is not guaranteed by other wireless TCP proposals.

e) The proposed scheme maintains the *Fairness Index* of native TCP scheme.

It must be noted that the Adaptive TCP refines TCP's congestion control mechanism coupled with loss recovery and does not interfere with TCP's *self-clocking* behavior. In this context, it does not take care of compromise in TCP's responsiveness due to unavoidable increase in RTT. The performance improvement may be investigated further to address the issues accounted for TCP's *slow responsiveness*, particularly in IEEE 802.11 wireless networks.

References

[1] Allman, M., Paxson, V. and Blanton, E. (2009) TCP Congestion Control. RFC5681. http://dx.doi.org/10.17487/rfc5681

[2] Cisco (2012) Cisco Visual Networking Index: Forecast and Methodology 2012-2017. White Paper.

[3] Foukalas, F., Gazis, V. and Alonistioti, N. (2008) Cross-Layer Design Proposals for Wireless Mobile Networks: A Survey and Taxonomy. *IEEE Communication Surveys and Tutorials*, **10**, 70-85. http://dx.doi.org/10.1109/COMST.2008.4483671

[4] Al-Jubari, A.M., Othman, M., Ali, B.M. and Hamid, N.A.W.A. (2011) TCP Performance in Multi-Hop Wireless Adhoc Networks: Challenges and Solution. *EURASIP Journal on Wireless Communications and Networking*, **198**, 1-20. http://dx.doi.org/10.1186/1687-1499-2011-198

[5] Sreekumari, P. and Lee, M. (2013) TCP NRT: A New TCP Algorithm for Differentiating Non-Congestion Retransmission Timeouts over Multihop Wireless Networks. *EURASIP Journal on Wireless Communications and Networking*, **2013**, 172. http://dx.doi.org/10.1186/1687-1499-2013-172

[6] Lim, C.H. and Jang, J.W. (2008) Robust End-to-End Loss Differentiation Scheme for TCP over Wired/Wireless Net-

works. *IET Communication*, **2**, 284-291. http://dx.doi.org/10.1049/iet-com:20060172

[7] Lohier, S., Doudane, Y.G. and Pujolle, G. (2007) Cross-Layer Loss Differentiation Algorithms to Improve TCP Performance in WLANs. *Springer US Journal of Telecommunication Systems*, **36**, 61-72.

[8] Sreekumari, P. and Chung, S.-H. (2011) TCP NCE: A Unified Solution for Non-Congestion Events to Improve the Performance of TCP over Wireless Networks. *EURASIP Journal on Wireless Communications and Networking*, **23**, 1-20. http://dx.doi.org/10.1186/1687-1499-2011-23

[9] Ludwig, R. and Gurtov, A. (2005) The Eifel Response Algorithm for TCP. RFC4015.

[10] Sarolahti, P. and Kojo, M. (2005) Forward RTO-Recovery (FRTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and the Stream Control Transmission Protocol (SCTP). RFC4138.

[11] Ho, C.Y., Chen, Y.C., Chan, Y.C. and Ho, C.Y. (2008) Fast Retransmit and Fast Recovery Schemes of Transport Protocols: A Survey And Taxonomy. *Computer Networks*, **52**, 1308-1327. http://dx.doi.org/10.1016/j.comnet.2007.12.012

[12] El-Ocla, H. (2010) TCP CERL: Congestion Control Enhancement over Wireless Networks. *Wireless Networks*, **16**, 183-198. http://dx.doi.org/10.1007/s11276-008-0123-4

[13] Ciullo, D., Mellia, M. and Meo, M. (2009) Two Schemes to Reduce Latency in Short Lived TCP Flows. *IEEE Communications Letters*, **13**, 806-808. http://dx.doi.org/10.1109/LCOMM.2009.091149

[14] Chu, J., Dukkipati, N., Cheng, Y. and Mathis, M. (2013) Increasing TCPs Initial Window. IETF, Experimental RFC 6928. http://dx.doi.org/10.17487/rfc6928

[15] Lu, X., Zhang, K., Foh, C.H. and Fu, C.P. (2014) SSthreshless Start: A Sender-Side TCP Intelligence for Long Fat Network. http://arxiv.org/abs/1401.7146

[16] Padmamabhan, V.N. and Katz, R.H. (1998) TCP Fast Start: A Technique for Speeding up Web Transfers. *Proceedings of IEEE GLOBECOM*'98, Sydney, 8-12 November 1998. http://www.cs.berkeley.edu/~padmanab/papers/gi98-unabridged.ps

[17] Floyd, S., Allman, M., Jain, A. and Sarolahti, P. (2007) Quick-Start for TCP and IP. RFC 4782, January 2007. http://dx.doi.org/10.17487/rfc4782

[18] Ros, D. and Welzl, M. (2013) Less-Than-Best-Effort Service: A Survey of End-to-End Approaches. *IEEE Communication Surveys and Tutorials*, **15**, 898-908. http://dx.doi.org/10.1109/SURV.2012.060912.00176

[19] Sardar, B. and Saha, D. (2006) A Survey of TCP Enhancements for Last-Hop Wireless Networks. *IEEE Communications Surveys & Tutorials*, **8**, 20-34.

[20] Touch, J. (2012) Automating the Initial Window in TCP. Work in Progress, July 2012.

[21] Goff, T., Moronski, J., Phatak, D.S. and Gupta, V. (2000) Freeze-TCP: A True End-To-End TCP Enhancement Mechanism for Mobile Environments. *Proceedings of the* 19*th Annual Joint Conference of the IEEE Computer and Communications Societies*, Tel Aviv, 26-30 March 2000, 1537-1545.

[22] Kliazovich, D., Redana, S. and Granelli, F. (2012) Cross-Layer Error Recovery in Wireless Access Networks: The ARQ Proxy Approach. *International Journal of Communication Systems*, **25**, 461-477.

[23] Chinta, M., Helal, A. and Lee, C. (2003) ILC-TCP: An Interlayer Collaboration Protocol for TCP Performance Improvement in Mobile and Wireless Environments. *Proceedings of the* 2003 *IEEE Wireless Communications and Networking*, New Orleans, 20-20 March 2003, 1004-1010. http://dx.doi.org/10.1109/WCNC.2003.1200509

[24] Cai, Y.G., Jiang, S.M., Guan, Q.S. and Yu, F. (2013) Decoupling Congestion Control from TCP (Semi-TCP) for Multi-Hop Wireless Networks. *EURASIP Journal on Wireless Communications and Networking*, **2013**, 149. http://dx.doi.org/10.1186/1687-1499-2013-149

[25] Cen, S., Cosman, P.C. and Voelker, G.M. (2003) End-to-End Differentiation of Congestion and Wireless Losses. *IEEE/ACM Transactions on Networking*, **11**, 703-717. http://dx.doi.org/10.1109/TNET.2003.818187

[26] Park, M.-Y., Chung, S.-H. and Ahn, C.-W. (2012) TCPs Dynamic Adjustment of Transmission Rate to Packet Losses in Wireless Networks. *EURASIP Journal on Wireless Communications and Networking*, **2012**, 304.

[27] Mascolo, S., Grieco, L.A., Ferorelli, R., Camarda, P. and Piscitelli, G. (2004) Performance Evaluation of Westwood+ TCP Congestion Control. *Journal of Performance Evaluation-Internet Performance Symposium*, **55**, 93-111.

[28] Parvez, N. and Hossain, E. (2005) TCP Prairie: A Sender-Only TCP Modification Based on Adaptive Bandwidth Estimation in Wired-Wireless Networks. *Computer Communication*, **28**, 246-256. http://dx.doi.org/10.1016/j.comcom.2004.08.001

[29] Wu, E.H.K. and Chen, M.-Z. (2004) JTCP: Jitter-Based TCP for Heterogeneous Wireless Networks. *IEEE Journal on Selected Areas in Communications*, **22**, 757-766. http://dx.doi.org/10.1109/JSAC.2004.825999

[30] Fu, C.P. and Liew, S.C. (2003) TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks. *IEEE Journal on Selected Areas in Communications*, **21**, 216-228.

[31] Leung, K.C., Li, V.O.K. and Yang, D.Q. (2007) An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges. *IEEE Transactions on Parallel and Distributed Systems*, **18**, 522-535. http://dx.doi.org/10.1109/TPDS.2007.1011

[32] Scharf, M. (2011) Comparison of End-to-End and Network Supported Fast Startup Congestion Control Schemes. *Computer Networks*, **55**, 1921-1940.

[33] Kodama, S., Shimamura, M. and Iida, K. (2011) Initial CWND Determination Method for Fast Startup TCP Algorithms. *Proceedings of the* 19*th International Workshop on Quality of Service* (*IWQoS*), San Jose, 6-7 June 2011, 1-3.

[34] Poojary, S. and Sharma, V. (2011) Analytical Model for Congestion Control and Throughput with TCP CUBIC Connections. *Proceedings of the* 2011 *IEEE Global Telecommunications Conference* (*GLOBECOM* 2011), Houston, 5-9 December 2011, 1-6.

[35] Wang, J.Y., Wen, J.T., Han, Y.X., Zhang, J., Li, C. and Xiong, Z. (2013) CUBIC-FIT: A High Performance and TCP CUBIC Friendly Congestion Control Algorithm. *IEEE Communications Letters*, **17**, 1664-1667. http://dx.doi.org/10.1109/LCOMM.2013.060513.130664

[36] Dunaytsev, R., Moltchanov, D., Koucheryavy, Y. and Harju, J. (2011) Modeling TCP SACK Performance over Wireless Channels with Completely Reliable ARQ/FEC. *International Journal of Communication Systems*, **24**, 1533-1564.

[37] Dalal, P., Kothari, N. and Dasgupta, K. (2011) Improving TCP Performance over Wireless Network with Frequent Disconnections. *International Journal of Computer Networks and Communication*, **3**, 169-184. http://dx.doi.org/10.5121/ijcnc.2011.3611

[38] Choi, S., Park, K. and Kim, C.K. (2006) Performance Impact of Interlayer Dependence in Infrastructure WLANs. *IEEE Transactions on Mobile Computing*, **5**, 829-845. http://dx.doi.org/10.1109/TMC.2006.102

[39] Jain, R., Chiu, D. and Hawe, W. (1984) A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems. DEC Technical Report DEC-TR-301.