Int. J. Communications, Network and System Sciences, 2009, 3, 169-247
doi:10.4236/ijcns.2009.23024 Published Online June 2009 (http://www.SciRP.org/journal/ijcns/).

On Approaches to Congestion Control over Wireless Networks

David Q. LIU, Williana Jean BAPTISTE
Department of Computer Science,
Indiana University - Purdue University Fort Wayne, Fort Wayne, IN, USA
Email: {*liud, jeanw*}*@ipfw.edu*
Received January 25, 2008; *revised March* 29, 2009; *accepted April* 6, 2009

ABSTRACT

Congestion control in wireless networks has been extensively investigated over the years and several schemes and techniques have been developed, all with the aim of improving performance in wireless network. With the rapid expansion and implementation of wireless technology it is essential that the congestion control problem be solved. This paper presents a survey of five congestion control schemes which are different in slow start threshold calculation, bandwidth estimation, and congestion window manipulation. A comprehensive comparison of these approaches is given in relation to assumptions, bandwidth estimation, congestion window size manipulation, performance evaluation, fairness and friendliness and improved throughput.

Keywords: Transmission Control Protocol (TCP), Wireless Networks, Bandwidth Estimation, Congestion Window, Slow Start Threshold

1. Introduction

Congestion control in a TCP/IP-based internet is complex and challenging [1] and over the years a lot of effort and resources have been dedicated to the research in this area. TCP provides only end-to-end flow control and relies on packet loss as an indicator of congestion [1−3]. On the other hand, IP is a connectionless stateless protocol and has no provision for any mechanism to detect or control congestion.

TCP limits a sender's transmission rate relative to the network congestion such that if there is little congestion on the path between sender and receiver then the transmission rate will increase, otherwise if there is congestion, the transmission rate will decrease. TCP employs a window-based scheme to control the transmission rate and the size of the window directly limits the transmission rate. With TCP, congestion is avoided by changing the window size which greatly impacts the transmission rate.

Generally with most TCP versions used in the Internet today, if there is little or no congestion, the window size increases by some factor to the predefined size called the slow start threshold, *ssthresh.* After attaining the *ssthresh* size, the window size increases linearly. If a packet is lost or congestion is detected, the window size *is* decreased significantly to allow the network to recover from congestion.

The widely used standard TCP congestion control approach worked well for wired networks since loss of a packet was in most instances due to the congestion in the network. But with the rapid explosion of wireless networks [1,2,4], there is a significant increase in the number of combined wired and wireless networks and congestion control mechanisms previously used for wired networks do not perform well in the wireless links and wireless networks. The main reason for this decrease in performance of the widely used TCP congestion control mechanisms is that for wireless networks packet loss is caused frequently by several factors other than congestion such as noisy channels or fading radio signals, interference, host mobility and disconnection due to limited coverage [1,5,6].

The current TCP mechanisms can not distinguish congestion due to wireless fading channels or bandwidth reduction and therefore make unnecessary reduction in

the congestion window size(*cwnd*) and cause severe performance degradation [1,2,6,7].

Significant efforts and resources have been utilized in researching and developing techniques that would enhance performance in the wireless portion of wired-wireless networks. Two studies [6,7] show that accurate estimation of the available bandwidth for *ssthresh calculation and setting* greatly improves performance. Another study [4] indicates that effectively manipulating the size of the window is essential in improving performance. A combination of both bandwidth calculation and window manipulation is proposed in [2,5] to improve performance.

This paper reviews five approaches to TCP congestion control and review their implementations based on four techniques of managing the send window namely slow start, dynamic window sizing, fast retransmit and fast recovery. It is structured as follows; Section 2 describes five approaches to TCP congestion control for wireless networks including characteristics, algorithms and assumptions. In Section 3, these techniques are compared and contrasted for similarities and differences according to the areas of bandwidth estimation, congestion window calculation, performance, fairness and related results. Concluding remarks are stated in Section 4.

2. Overview of Congestion Control Techniques

In this section, several congestion control techniques over wireless networks are described. The Table 1 lists the terms used in these techniques.

2.1. TCP Enhancement for Transmission in Variable Bandwidth Wireless Environment

Since network bandwidth changes constantly especially in wireless networks, TCP must frequently probe the extra bandwidth of a network to optimally use the available bandwidth by adequately setting the slow start threshold. A scheme is proposed in [4] that dynamically sets the slow start threshold and manipulates the window size in both the slow start phase and the congestion avoidance phase. The slow start threshold is calculated by combining the expected rate with the actual rate to obtain an appropriate rate.

2.1.1. Slow Start Threshold Estimation

The *ssthresh* estimation calculates an appropriate *ssthresh* by combining the expected rate with the actual rate and is defined as follows:

<expected rate>=$cwnd/rtt_{min}$;

<actual rate>=$cwnd/rtt$;

AR=<expected rate>×β+<actual rate>×(1-β);

2.1.2. Congestion Window Estimation

The congestion window is calculated based on the degree of variation of rtt. For three consecutive increases in rtt the congestion window is defined as follows:

if $rtt_{va} < ½$, $cwnd_{next}=cwnd_{cur}+1$else $cwnd_{next} = cwnd_{cur}$

2.1.2.1. Slow Start Phase

TCP enters this phase when a connection is initiated or on timeout.

when timeout {

cwnd = 1; *ssthresh* = AR * rtt_{min}/seg_size;

if *ssthresh* < 2, *ssthresh* = 2;

when an ACK is received {

If *cwnd* < *ssthresh*, *ssthresh* = AR * rtt_{min}/seg_size

else cwnd = ssthresh, enter congestion avoidance phase

2.1.2.2. Congestion Avoidance Phase

For three consecutive increases of *rtt*,

if $var_{rtt} < ½$, $cwnd_{next} = cwnd_{cur} + 1$

For three consecutive decreases in rtt

if (rtt_{va}<1/3), $cwnd_{next} = cwnd_{cur} +1$

else if (1/3<= rtt_{va} <= 2/3) $cwnd_{next} = cwnd_{cur} +3$

else if (rtt_{va} > 2/3) $cwnd_{next} = cwnd_{cur} + 5$

2.1.2.3. Fast Retransmission Phase

This phase is entered when three duplicate ACKs are received. The sender immediately sends the out of sequence packet without waiting for the timer to expire.

ssthresh = AR * rtt_{min} / seg_size.

cwnd = *ssthresh.*

2.1.2.4. Fast Recovery Phase

TCP enters this phase after the fast retransmission phase. In order to reduce transmission, TCP sets *cwnd* = *ssthresh* and enters the congestion avoidance phase.

2.1.2.5. Retransmission Timeout Phase

TCP sets *cwnd* = 1 and enters the slow start phase.

Table 1. Congestion control terms.

Term	Meaning
ACK	acknowledgement
AR	appropriate rate
B_m	measured bandwidth
B_s	smoothed bandwidth
BWE	bandwidth estimation
cwnd	congestion window size
rtt	round trip time
rtt_{acr}	archived rtt
rtt_{var}	variation of rtt
seg_size	segment size
ssthresh	slow start threshold

2.2. Modified TCP Congestion Control Algorithm (Constant TCP)

Modified sender's TCP congestion control [2] uses a constant congestion window. The foundation of the proposal is based on the following assumption: if a TCP sender transmits packets at a rate greater than its fair share then some packets from the previous round would be in the network when the next round of packets are transmitted. If the load of the network is expressed in terms of queue length over some fixed time interval then L load at instant *i* is

$L_i = N + L_{i-1}$,

where N is the average amount of the new arriving traffic and L_{i-1} is the amount of traffic left after the last time interval. If the sender is transmitting packet with its fair share then $L_{i-1} = 0$ *and* $L_i = N$.

In this scheme, a TCP connection is divided into a number of slots such that the fair share of a connection of the network bandwidth remains unchanged for a slot period during the lifetime of the connection. Changes in the available bandwidth are due to connections leaving and joining the network at that time the current slot ends and a new slot starts. The bandwidth calculation algorithm similar to [4] is used to estimate the available fair share for a slot. A new slot triggers the recalculation of the congestion window, which is set according to the available connection bandwidth. In this proposal the modified TCP sender goes through three phases during the lifetime of the connection. They are the start-up, the window recalculation phase and the constant window phase.

2.2.1. Startup Phase

When a connection is initiated, the sender uses the slow start phase for *k rtt* rounds gathering data such as the minimum *rtt* and the network bandwidth in order to calculate the starting *cwnd*.

2.2.2. Window Recalculation Phase

TCP enters this phase when there is a change in the available fair share triggered by a change in *rtt* values. The rtt values are archived as rtt_{acr} for future use and

$cwnd = \text{BWE} * rtt_{\min} / seg\_size$

2.2.3. Constant Window Phase

In this phase the *cwnd* calculated from the start up phase is kept constant regardless of the number of ACKs or DUPACKs received or timeouts, but changes in the *rtt* will trigger a window recalculation by tracking the *rtt* estimates from received segments. If $|rtt_{acr} - rtt_{var}| / rtt_{arc} > \beta$ then a window recalculation phase is entered.

2.3. Two Phase Congestion Control (TCP-TP)

The sender-side control congestion scheme TP-TCP [7] measures the network capacity and uses it to set the congestion window size and the transmission rate in order to optimally utilize the available bandwidth. TCP-TP includes the fair convergence phase and the congestion avoidance phase. In addition, the receiver delays ACKs to reduce the number of packets in the network.

2.3.1. Bandwidth Estimation

The TCP-TP sender measures the bandwidth during the lifetime of the connection and uses this information to calculate the congestion window to optimally utilize the available bandwidth. The bandwidth is measured using the following equation:

B_m=<bytes received between two successive ACKs>/ <time interval between two successive ACKs> This measured bandwidth B_m reflects the network environment at that point in time.

Due to the constant changes of bandwidth in the network, the scheme uses a smoothed bandwidth B_s instead of the measured bandwidth B_m. The smoothed bandwidth is the sum of the previously smoothed bandwidth and the current measured bandwidth according to the following equation:

$$B_{s(i)} = \alpha B_{s(i-1)} + (1-\alpha) B_m$$

2.3.2. Window Calculation

Initially,

ssthresh is calculated as follows:

ssthresh = <*network bandwidth*> * *rtt*

$cwnd = B_s * rtt_{min}$

2.3.2.1. Congestion Control Avoidance Phase

When *cwnd* = *ssthresh*, TCP-TP enters this phase and works like the standard TCP except that the cwnd increases and decreases by N at once for every N packets rtt.

2.3.2.2. Fair Convergence Phase

In this phase TCP-TP measures the network bandwidth and calculates and sets the *cwnd* and *ssthresh.* After each ACK is received, *cwnd* increases by β bytes where

$$\beta = \gamma\, rtt^2\, (ssthresh\text{-}cwnd)$$

2.4. TCP-Westwood Bandwidth Estimation

The sender of the modified TCP-Westwood (TCPW) [5] continuously measures the round trip time of the returning ACKs to calculate a minimum slow start threshold and congestion window which effectively utilizes the bandwidth at the time of congestion. This scheme TCPW uses a two-phase approach to control congestion namely, bandwidth estimation phase and the congestion control phase.

TCPW assumes that on receipt of three DUPACKs, the network capacity has been reached or packets have been dropped due to sporadic loss in wireless networks.

2.4.1. Bandwidth Estimation Phase

In this phase the sender continuously probes the network connection and calculates the available bandwidth and tracks the rtt from the returning ACKs. After a congestion episode the calculated bandwidth is used to determine the *ssthresh* and *cwnd.*

Before congestion episode the TCP sender increases the *cwnd* to determine network capacity and the bandwidth estimation is calculated as follows

$$Bwe= d_k /t_k - t_{k-1}$$

where

d_k: data transferred at time k

t_k: the time ACK was received at source for transmission of data k

t_{k-1}: the time ACK was received at source for transmission of previous data k-1.

Averaging the sample measurements accounts for the low frequency components of the available bandwidth and a low pass filter is used on the estimated bandwidth.

2.4.2. Calculation of Congestion Window

The *cwnd* is calculated using the *ssthresh* after 3 DUPACKs are received or timeout expiration.

2.4.3. Slow Start Phase

At the beginning of a connection:

cwnd =1

ssthresh = BWE x rtt_{min}/*seg_size*

cwnd increases by 1 for each new ACK receipt

until *cwnd* = *ssthresh*

If (timeout expires)

ssthresh = BWE x rtt_{min}/*seg_size*;

if (*ssthresh* < 2*) ssthresh*=2*;*

cwnd =1

2.4.4. Congestion Avoidance Phase

During this phase the sender probes for extra bandwidth and exponentially increases the *cwnd*. Once three DUPACKs are received, the network is at its capacity and this scheme uses the following algorithm for setting the *cwnd* and *ssthresh*:

If (n DUPACKs are received)

ssthresh = BWE x rtt_{min}/*seg_size*;
if (*cwnd*>*ssthresh*) *cwnd*=*ssthresh*

2.5. Enhanced Bandwidth Estimation (TIBET)

A bandwidth estimation scheme, Time Intervals based Bandwidth Estimation Technique (TIBET) [6], modifies the sender side of the TCP congestion control procedure. TIBET is based on the principle that if more information is available, the better is the estimation of the available bandwidth to a connection, leading to better and fair utilization of network resources.

If *n* packets (L_1,L_2,L_3…L_n) are transmitted within a time interval of *T*, then the average bandwidth BWE is given by

$$BWE=1/T *\sum L_i \qquad \text{where i =1 to n}$$

which can be rewritten as

$BWE=L_{mean}/(T/n)$, where L_{mean} is the average packet length in bits and T/n is the average interarrival time. Thus average used bandwidth over a time period is equal to the average packet length in bits transmitted during that time period/average inter arrival time. This scheme also proposes the low-pass filtering of either the packets lengths and their inter departure times.

2.5.1. Bandwidth Estimation Phase

Below is the pseudo code for estimation of bandwidth based on transmitted packets.

```
if(packet is sent)
   sample_length[k] = (packet-size * 8);
   sample_interval[k] = now –last_sending_time;
   avg_packet_length[k] = α * avg_packet _length [k-1]
                          +(1-α)* sample_length[k];
   avg_interval[k]=α * avg_interval[k-1]
                   + (1-α)* sample_interval[k];
  BWE[k] =avg_packet_length[k] / avg_interval[k};
```

where packet size is the *segment size* in bytes, *now* is the current time, *last_sending_time* is the time of the previous packet transmitted, α is the low-pass filter, and *BWE* is the estimated value of the used bandwidth.

A second alternative for calculating the average bandwidth is also proposed for received ACKs. The algorithm used for the ACKs is similar to the stated above except for the calculation of:

```
sample_length[k] = (acked * packet_size * 8)
sample_interval[k] = now – last_acked_time;
```

where last_acked_time is the time the last ACK was received, and acked is the number of segments acknowledged by the last ACK.

2.5.2. Calculation of Congestion Window

The *cwnd* is set to 1 after 3 DUPACKs are received or timer expires, and then the slow start phase is entered. The cwnd grows exponentially as usual until cwnd = ssthresh and at that time, the congestion avoidance phases is entered.

2.5.3. Slow Start Phase

At this phase, the *cwnd* and the *ssthresh* are set as follows:

ssthresh=BWE * rtt_{min}
cwnd=1

2.5.4. Congestion Avoidance Phase

During this phase the sender probes for extra bandwidth and exponentially increases *cwnd* to *ssthresh*. Once the

ssthresh has been reached, the *cwnd* increases by one for each ACK received. If three DUPACKs are received, the network has reached its capacity.

If ($cwnd = ssthresh$) $rtt_{min}=(1-\beta)\ x)\ rtt_{min}$

At this time, slow start phase is entered.

3. Comparison of Various TCP Congestion Control Techniques

The congestion control schemes presented in Section 2 are compared with respect to assumptions, bandwidth estimation, window size manipulation, slow start phase, retransmission phases, congestion avoidance phase and results.

3.1. Comparison of Assumptions

All schemes comply with true end-to-end TCP design principle and do not require the interception of packets by intermediate nodes. Further, for all schemes included in this survey the modifications were made only to the sender side of the traditional TCP congestion control algorithm. Each scheme is based on specific assumptions. For example, the constant TCP assumes that most indications of congestion by the current TCP variants used in the Internet does not necessitate a reduction in the transmission rate of the connection, as such, the *cwnd* size should remain constant until some other factors indicate that true congestion has occurred.

3.2. Comparison of Bandwidth Estimations

Most approaches [2,5−7] described in Section 2 state that the bandwidth estimation algorithm in Reno is inaccurate and causes the under utilization of available bandwidths by TCP entities and propose alternate bandwidth measurements that would optimally utilize the available bandwidth and improve transmission rate. The bandwidth is estimated using the average rate of returning ACKs [2,5]. This estimation more accurately reflects the TCP entity's fair share. The estimated bandwidth [5] is then used to set the *cwnd* and *ssthresh* after congestion episode or timeout expiration.

Another improved bandwidth estimation [7] is the rate of bytes received during immediate successive inter-arrival ACKs. Paper [6] proposes using a low-pass filter rate of average packet length in bytes for inter-arrival times for transmitted packets. This algorithm estimates the used bandwidth by measuring the inter-arrival samples and not the bandwidth samples compared with the algorithm used by [2,5] which directly samples the bandwidth. All bandwidth estimations were smoothed by using a low pass filter to account for the rapidly fluctuating network environment.

None of the papers reviewed compared their bandwidth estimation algorithms with regards to enhancing performance. All assume that their modified bandwidth estimation algorithm would more accurately estimate the available bandwidth resulting in optimal use of the connection's fair share.

3.3. Comparison of Congestion Window Size Manipulation Techniques

Since rate of transmission is indirectly [1,2] related to the congestion window size, effectively manipulating the window size will improve transmission rates because in wireless networks window size is unnecessarily reduced due to loss prone nature of the wireless links and not as a result of congestion. Several approaches manipulate the congestion window size and set the slow start threshold in order to maintain a high transmission rate comparable to the available bandwidth. One of them [2] maintains a constant congestion window and does not react by decreasing the window size when DUACKs are received and timeout expires. Instead, responding only when the network environment becomes sufficiently degraded through monitoring rtt values. Another scheme [4] proposes increasing (decreasing) the congestion window only when the change of three *rtt* values is greater (less) than some predefined factor.

Changing the window size by some fix factor is stated in [7] and that factor is calculated using the current available bandwidth. Other techniques reset the congestion window to the either the previously calculated slow start threshold [5] or to the newly calculated slow start threshold [4] which takes the current network environment into consideration once the network capacity is reached.

All these techniques aim to limit the unnecessary reduction in window size in wireless links thereby improving overall throughput.

3.4.1. Slow Start Phase

In this phase, available bandwidth is probed and the congestion window is increased by some factor. Various approaches are proposed for this phase. Three conditions would cause TCP entities to enter this phase and include starting up a connection, receipt of 3 DUPACKS and timer expiration. For all the proposed approaches at the start of a connection, the *cwnd* is set to either one [4−7] or a fixed value obtained after probing the bandwidth for a fixed number of round trip times [4].

As each new ACK is received, the *cwnd* is increased by one and information such as bandwidth and *rtt* measurements are collected in order to calculate *ssthresh* until the *cwnd* reaches the *sshtresh* value. One approach [2] has no need for a slow start phase since the congestion window once calculated is kept constant irrespective of detection of congestion.

3.4.2. Retransmission Timeout Phase

TCP enters this phase when the timer set on a packet transmitted expires before an ACK is received for that

packet. Several techniques are used to decrease the transmission rate in order to alleviate congestion in the network.

One of these techniques is setting cwnd to one [4,5,7] whilst the ssthresh is set to the value of the bandwidth estimated at that time multiplied by the rttmin [5]. The ssthresh is set to <actual rate> * rttmin/seg_size, which would allow for faster recovery.

Another scheme [2] does not enter this phase since most times the timeout is not an indication of congestion in wireless and there is no need to drastically reduce the transmission rate to reduce congestion.

3.4.3. Congestion Avoidance Phase

Once the congestion window equals the *ssthresh* this phase is entered and *cwnd* is increased or decreased by various functions. Various techniques are employed in this phase. One such technique is monitoring the *rtt* measurements and recalculating *cwnd* when some preset condition is met [2,4].

In [4], if there are three consecutive *rtt* value increases or decreases, then the *cwnd* is decreased or increased by a factor. However, in [2], *cwnd* is kept constant until the measured *rtt* function is greater than some fixed value, and at that time *cwnd* is recalculated and set to *BWE * rttmin/seg_size*. Another technique used in this phase is decreasing and increasing *cwnd* by a fix number of bytes [7].

3.5. Comparison of Performance Evaluation

Most congestion control schemes used NS2 simulations to evaluate their performance except for [7] where experiments were performed by modification of Linus Kernel 2.6.7. Commonly used performance metrics were employed in both the simulations and experiments and include error rate, bandwidth, link capacity, number of connections and rtt length, fairness and friendliness. Table 2 shows metrics used in each scheme.

The percentage increase in throughput compared to the standard TCP congestion control technique implemented in the Internet is presented in Table 3. Various levels in improvement in throughput were observed for all schemes, ranging from 10% to 550%.

Due to the lack of commonality amongst the compared metrics it was generally impossible to compare techniques against each other to determine the best algorithm. However, all techniques compared their improve performance against the TCPW and the results are presented in Figure 1−Figure 4. Overall, the TCP constant, TCP-TP and TCP-EVBWE all out perform TCPW in various network scenarios.

3.6. Comparison of Fairness and Friendliness

Fairness and friendliness are important metrics in evaluating the performance of a scheme. Fairness means that all similar connections have the same opportunity to transfer data and that one connection would not aggressively consume resources at the expense of other connections such that connections with longer round trip times are not at a disadvantage. Friendliness is that connections of different schemes are able to co-exist [8].

Table 2. Shows metrics used to evaluate performance.

Congestion	Error	Number of	Rtt	Band-
TCP-ETVBWE[4]	Yes	No	No	Yes
constant_TCP[2]	Yes	Yes	No	No
TCP_TP[7]	No	No	Yes	No
TCPW[5]	No	Yes	Yes	Yes
TIBET[6]	Yes	Yes	Yes	No

Table 3. Throughput increase for each proposed congestion control scheme.

Congestion Control Schemes	Throughput Increase%
TCP-ETVBWE[4]	10
Constant_TCP[2]	10-20
TCP_TP[7]	~10
TCPW[5]	394-550
TIBET[6]	50

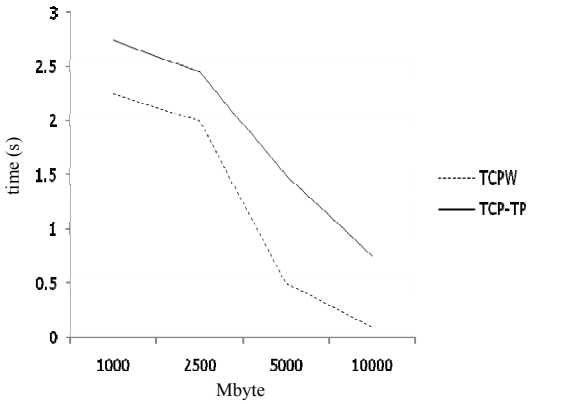

Figure 1. Throughput (Mbps) comparison in wireless networks with varying packet round trip times.

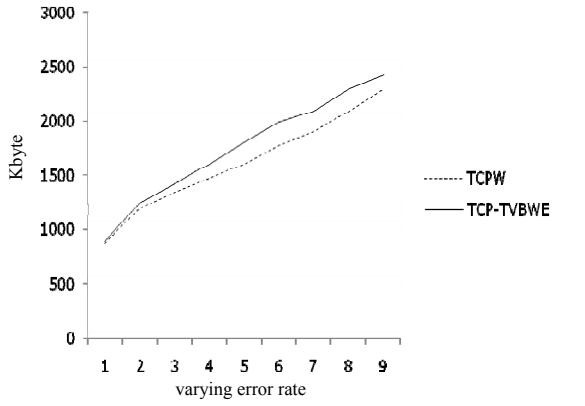

Figure 2. Throughput (Kbps) variation with varying error rates.

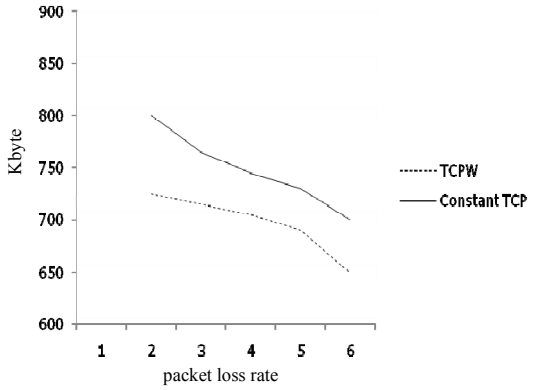

Figure 3. Throughput (Kbps) variation with packet loss rates in wireless links(%).

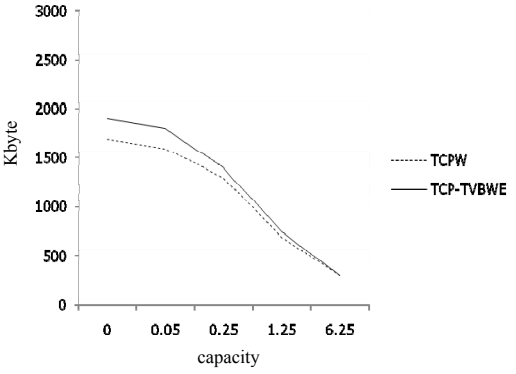

Figure 4. Throughput (Kbps) variation with capacity (Mbps) wireless links.

Table 4. Fairness and friendliness rank (0-3).

Congestion Control Schemes	Fairness	Friendliness
TCP-ETVBEW[4]	0	0
constant TCP[2]	0	0
TCP_TP[7]	3	3
TCPW[5]	2	2
TIBET[6]	3	3

0-not evaluated.

Some of these schemes were evaluated for fairness and friendliness and the results are ranked based on the extent of fairness and friendliness reported. TCP-TP and TIBET are both fair and friendly schemes while TCP constant and TCP-ETVBEW were not evaluated for fairness and friendliness [1,2]. It is difficult to see how TCP constant would be fair or friendly to other TCP entities since the *cwnd* remains constant even when the network environment changes and the TCP entity would continue to transmit at a high rate thereby consuming resources of other entities. The ranking of the proposed schemes in terms of fairness and friendliness is presented in Table 4.

4. Conclusions

Five sender side modification schemes to the standard TCP congestion control algorithm are surveyed in this paper. Their characteristics, algorithms and assumptions were presented. A comparison of their assumption, bandwidth estimation, window size manipulation, slow start phase, retransmission phases, congestion avoidance phase and performance evaluation methods is conducted. The need for an efficient method to optimally utilize available bandwidth is essential in the wireless links of combined wired and wireless networks. Some schemes propose efficient estimation techniques of available bandwidth in a dynamic internet environment while others schemes effectively manipulate the congestion window and set the slow start threshold. With simulations and experiment each of these schemes shows an improvement in throughput between 10−50%.

One of our future research projects is to evaluate these schemes against each other by comparing them in various network scenarios such varying bit error rates, number of connections, link capacity, and bandwidth.

5. References

[1] S. Schmid and R. Wattenhofer, "A TCP with guaranteed performance in networks with dynamic congestion and random wireless losses", In Proceedings of the 2nd Annual International Wireless Internet Conference (WICON'06), August 2006.

[2] R. Roy, S. Das, A. Ghosh, and A. Mukherjee, "Modified TCP congestion control algorithm for throughput enhancement in wired-cum-wireless networks" In Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2007.

[3] W. Stallings, "High-speed networks and internets performance and quality of service," Prentice Hall Inc., 2002.

[4] N. Wang, C. Chiou, and Y. Huang, "TCP enhancement for transmission in a variable bandwidth wireless environment," Proceedings of IWCMC, pp. 37−42, August 2007.

[5] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (ACM SIGMOBILE), pp. 287−297, 2001.

[6] A. Capone, L. Fratta, and F. Martignon, "Enhanced bandwisth estimation algorithms in TCP congestion control scheme," in Proceedings of the IFIP Conference on Network Control and Engineering of QoS, Security and Mobility, pp. 469−480, 2002.

[7] J. Lee, H. Cha, and R. Ha, "A two-phase TCP congestion control for reducing bias over heterogeneous networks," in Proceedings of International Conference on Information Networking, Convergence in Broadband and Mobile Networking, (ICOIN), pp. 9−108, 2005.

[8] A. Ghosh, S. Das, R. Roy, and A. Mukherjee, "Constant congestion window approach for TCP-effect on fairness," in Proceedings of 3rd Swedish National Computer Networking Workshop, September 2005.

