Effectuation Of TCP Agents And Equivalence Of Outcome With Different TCP Variants

Madhvi A. Bera[1] Bhumika S. Zalavadia[2] Rashmi Agrawal[3]
[1, 2, 3]Dept. of Computer Engineering
[1, 2, 3]Atmiya Inst. of Technology & Science, Rajkot, India,

Abstract--- Transmission Control Protocol (TCP) consist a lot of rules that make control communication, there are a lot of version from TCP, their role to control the congestion then graphically examine slow start, congestion avoidance, fast retransmission and fast recovery . The performance for different TCP versions algorithm like TCP Tahoe, Reno, New Reno, SACK, FACK, and Vegas will be different depending on the fundamental functions such as slow start, congestion avoidance, fast retransmission and fast recovery.

I. Introduction

The transmission control protocol is one of the two protocols that make up the Internet connection suite. This protocol is combined with the Internet protocol to form the foundation of nearly all Internet traffic. These protocols are almost inseparable in use, relying completely on each other for proper functioning; as a result, the Internet connection suite is typically abbreviated as TCP/IP (transmission control protocol/Internet protocol). The transmission control protocol is responsible for the disassembling and reassembling of data, and the Internet protocol handles the routing and transmission.
Internet traffic is primarily made of up of small data bursts called packets. These packets contain information relating to the origin and destination of the data as well as some optional additional information. The packets are created and reassembled by the transmission control protocol and sent over the Internet by the Internet protocol.
Technically, TCP passes data in the shape of segments and each segment is divided as header and data section. Soon as TCP acknowledges the data from a stream, fragment it into various chunks and then further adds a header in order to create a TCP segment. But that isn’t all; TCP segment after that are encapsulated into an IP datagram etc. Always data section with the payload data carrying for the application is following by the header section. Well, TCP makes sure of keeping on track the individual segments of data transmission.
The implication of the difference is that packet losses are no longer mainly due to network congestion; they may well be due to some wireless specific reasons. As a matter of fact, in wireless LANs or cellular networks, most packet losses are due to high bit error rate in wireless channels and handoffs between two cells, while in mobile ad hoc networks, most packet losses are due to medium contention and route breakages, as well as radio channel errors. Therefore, although TCP performs well in wired networks, it will suffer from serious performance degradation in wireless networks if it misinterprets such non-congestion related losses as a sign of congestion and consequently invoke congestion control and avoidance procedures, as confirmed through analysis and extensive simulations carried out[21].
Moreover, during the entire TCP connection lifetime, it undergoes from different changes in terms of state being. These changes can be differentiated such as the following: LISTENING, (server side), SYN-SENT (made by clients), SYN-RECEIVED (made by servers), ESTABLISHED (port readiness for receiving and sending data), FIN-WAIT-1, FIN-WAIT-2 (point out that server application is in standing-by position to close etc) CLOSE-WAIT, LAST-ACK (point out that server is standing-by to begin connection termination from its side), TIME-WAIT and connection close stage. In addition to this, connection establishment is based on a three-way process of handshake. But server has to keep a port free for a client, going to connect to that server for connection purpose (passive open). After that, a client initiates an active open in order to establish a connection using following three-way handshake process: SYN, SYN-ACK and ACK**.**

II. TCP PERFORMANCE IN MANET

Even though TCP ensures reliable end-to-end message transmission over wired networks, a number of existing researches have showed that TCP performance can be substantially degraded in MANET. This section continues with a description of different types of constraints influencing TCP performance in MANET [21] [25].

A. Route Failure

In MANET, the mobility of the node is considered as the major reason for the route failure and the route reestablishment is instantly needed in case of route failure. However, it is likely that a new route establishment may experience longer duration than the RTO of the sender. In consequence of that, the TCP sender will unnecessary invoke congestion control mechanism.

B. Network Partitioning

A network partition takes place when a node departs from the network, resulting in an isolation of some parts of a mobile ad-hoc network. These fragmented portions are defined as partitions. In MANET, TCP considers network partitioning as one of the most imperative challenges which is mainly caused due to the mobility or energy-constrained (limited battery power) operation of nodes. When the source and the destination of a TCP connection lie in different parts of the network, all transmitting packets are found to be dropped by the network. As a result, the congestion control algorithm will be invoked instantly by the TCP sender

C. Hidden and Exposed Node Impact

Figure presents a typical hidden node condition where the node B is within the range of nodes A and C, but nodes A and C are not in each other's range. When node A is not aware that node B is currently busy receiving from node C, and therefore may start its own transmission, causing a collision.

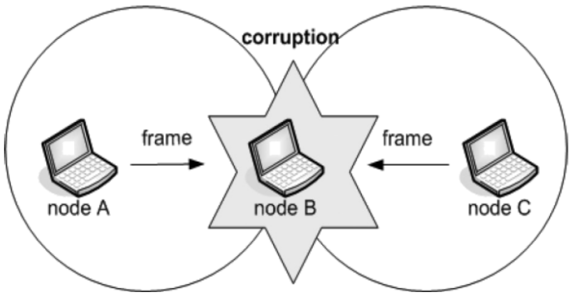


Fig. 1: Hidden node Impact

Figure The "exposed node" problem: This problem occurs when a node is prevented from sending packets to other nodes due to a neighboring transmitter. For example, node C wants to transmit to node D but mistakenly thinks that this will interfere with B's transmission to A, so C refrains from transmitting. The "exposed node" problem leads to loss of efficiency.

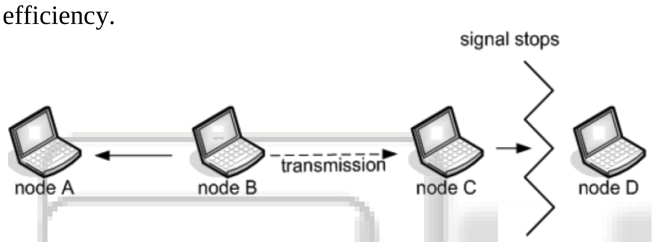


Fig. 2: Exposed Node Impact

D. *Simple TCP:*

It is well known that TCP is a connection-oriented transport protocol that is aimed at guaranteeing end-to-end reliable ordered delivery of data packets over wired networks. For this purpose, basic functionalities such as flow control, error control, and congestion control are indispensable. While these functions have a clean-cut definition of their own, in practice they are closely coupled with one another in TCP implementation [1].

In TCP, a sliding window protocol is used to implement flow control, in which three windows are used, namely, Congestion Window, advertised window, and Transmission Window. Congestion window indicates the maximum number of segments (Without causing confusion, the term segment and packet are used interchangeably henceforth) that the sender can transmit without congesting the network. As shown next in details on congestion control, this number is determined by the sender based on the feedback from the network. Advertised window, however, is specified by the receiver in the acknowledgements it. Advertised window indicates to the sender the amount of data the receiver is ready to receive in the future. Normally, it equals to the available buffer size at the receiver in order to prevent buffer overflow. Transmission window means the maximum number of segments that the sender can transmit at one time without receiving any ACKs from the receiver. Its lower edge indicates the highest numbered segment acknowledged by the receiver. Obviously, to avoid network congestion and receiver buffer overflow, the size of transmission window is determined as the minimum of the congestion window and the receiver's advertised window [7].

E. *TCP variants*

Transmission Control Protocol (TCP) consist a set of policy that make manage communication, there are a group of version from TCP, their role to control the overcrowding then graphically examine measured start, congestion avoidance, fast retransmission and fast recovery . The performance for different TCP versions algorithm like TCP Tahoe, Reno, New Reno, SACK, FACK, and Vegas will be dissimilar depending on the primary functions such as slow start, congestion avoidance, fast retransmission and fast recovery. Among them, TCP Reno is by far most widely deployed [2].

1) *Fast retransmission and fast recover*

As noted earlier, a packet can be assumed lost if three duplicate ACKs are received. In this case, TCP performs a fast retransmission of the packet. This mechanism allows TCP to avoid a lengthy timeout during which no data is transferred. At the same time, *ssthresh* is set to one half of the current congestion window, i.e., *cwnd*, and *cwnd* is set to *ssthresh* plus three segments. If the ACK is received approximately one round trip after the missing segment is retransmitted, fast recovery is entered. That is, instead of setting *cwnd* to one segment and starting with slow start, TCP sets *cwnd* to *ssthresh*, and then steps into congestion avoidance phase. However, only one packet loss can be recovered during fast retransmission and fast recovery. Additional packet losses in the same window may require that the RTO expire before retransmission [25].

2) *Selective acknowledgment*

Owing to the fact that fast retransmission and fast recovery can only handle one packet loss from one window of data, TCP may experience poor performance when multiple packets are lost in one window. To overcome this limitation, recently the selective acknowledgement option (SACK) is suggested as an addition to the standard TCP implementation.

The SACK extension adopts two TCP options. One is an enabling option, which may be sent to indicate that the SACK option can be used upon connection establishment. The other is the ACK option itself, which may be sent by TCP receiver over an established connection if SACK option is enabled through sending the first option. The SACK option contains up to four (or three, if SACK is used in conjunction with the Timestamp option used for RTTM [24]) SACK blocks, which specifies contiguous blocks of the received data. Each SACK block consists of two sequence numbers which delimit the range of data the receiver has received and queued. A receiver can add the SACK option to ACKs it sends back to a SACK-enabled sender. In the event of multiple losses within a window, the sender can infer which packets have been lost and should be retransmitted using the information provided in the SACK blocks. A SACK-enabled sender can retransmit multiple lost packets in one RTT instead of detecting only one lost packet in each RTT [23].

3) *Random Early Detection (RED)*

Random Early Detection (RED) is a router-based congestion control mechanism that seeks to detect incipient congestion and notify some TCP senders of congestion by controlling the average queue size at the router. To notify the TCP senders of congestion, the router may mark or drop packets, depending on whether the senders are cooperative. As a response, the senders should reduce their transmission rate. This is done in two algorithms. The first algorithm is to compute the average queue size by using exponential weighted moving average. If we denote by avg and q the average queue size and the current queue size, respectively, then avg = (1-wq)*avg + wq*q, where wq is the queue weight. The other algorithm is to compute the packet-marking or packet-dropping probability pa. If avg falls in between minth and maxth, the packet marking probability pb = maxp (avg - minth) / (maxth - minth) and the final marking probability pa = pb/(1 – count*pb), where maxp and count are design parameters, respectively, denoting the maximum value for pb and the number of packets having arrived since last packet marking or dropping. If avg exceeds maxth, pa = 1, which means that the router marks or drops each packet that arrives. Through control over the average queue size prior to queue overflow, RED succeeds in preventing heavy network congestion and global synchronization as well as improving fairness. Notice that numerous variants of RED have been proposed to improve various performance of the original RED [22] [24].

4) Explicit Congestion Notification (ECN)

Most of current Internet routers employ traditional “drop-tail” queue management. In other words, the routers drop packets only when the queue overflows, which could lead to the undesirable global synchronization problem as well as heavy network congestion. Recently, active queue management (AQM) mechanisms have been proposed since they can detect congestion before the queue overflows at the routers and inform TCP senders of the congestion, thereby avoiding some of these problems caused by the “drop-tail” policy. In the absence of Explicit Congestion Notification (ECN), however, the only choice that is available to AQM for indicating congestion to end systems is to drop packets at the routers. With ECN, AQM mechanisms have an alternative to allow routers to notify end systems of congestion in the network [23].

F. *Sliding window method*

In many cases, it is possible to limit the number of acknowledgements, in order to relieve traffic on the network, by fixing a sequence number at the end of which an acknowledgement is required. This number is in fact stored in the *window* field of the TCP/IP header.

This method is effectively called the "*sliding window method*" because to some extent a range of sequences is defined that does not need acknowledgements and which moves as acknowledgements are received [14].

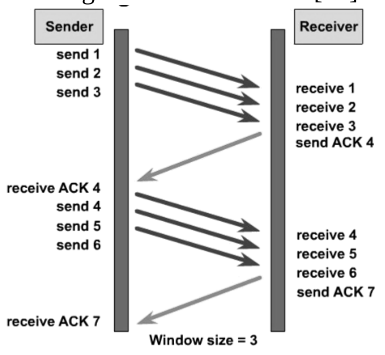
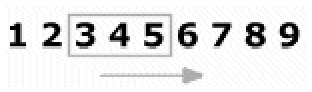
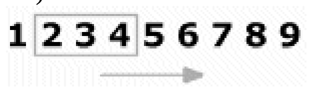


Fig. 3: Sliding window technique in TCP

In addition, the size of this window is not fixed. In fact, the server can include the size of the window which seems most suitable in its acknowledgements by storing it in the window field. So, when the acknowledgement indicates a request to increase the window, the client will move the right border of the window.



Conversely, in the case of a reduction, the client will not move the right border of the window towards the left but wait for the left border to advance (with the arrival of the acknowledgements).



1) Ending a connection

The client can request to end a connection in the same way as the server.

Ending a connection is done in the following way [16]:

- One of the machines sends a segment with the *FIN* flag set to 1, and the application puts itself in a waiting state, i.e. it finishes receiving the current segment and ignores the following ones.
- After receipt of this segment, the other machine sends an acknowledgement with the *FIN* flag set to 1 and continues to send the segments in progress. Following this, the machine informs the application that a *FIN* segment has been received, then sends a *FIN* segment to the other machine, which closes the connection.

2) TCP Tahoe

TCP Tahoe one of the various version of the TCP congestion control algorithm, Tahoe add some new improve on the TCP completion in the early stage. That enhance include slow start, congestion avoidance, and fast retransmission. This enhancement include the modify in the round-trip time based on location retransmission time out values[1].we can explain the TCP Tahoe as the fast retransmission algorithm outperforms when happened drop to the packet as the congestion. The sender must be coming up to get the retransmission timer to finish in the without fast retransmit enhance algorithm. The Tahoe algorithm will notice where they lose of the packet happened and then notice the full timeout instant. When Tahoe detected the loss of in the packet the TCP Tahoe performance will be slow, because this motivation transmission run will reduce quickly, which the fast transmission makes Tahoe do more considerably better than TCP completion in the loss and discovery the packet depending on the retransmission timer. It decide the important of Tahoe less than best performance on the very bandwidth relationship , this is because main of slow start ,and in this case may be happened multi losses within solitary window .that pass on to the sender will retransmit packet that have previously been delivered [1][5].

1) Determination of the available bandwidth.
2) Ensuring that equilibrium is maintained.
3) How to react to congestion

G. *Slow Start*

TCP packet transmissions are clocked by the incoming acknowledgements. However there is a problem when a connection first starts up cause to have acknowledgements you need to have data in the network and to put data in the network you need acknowledgements. To get around this circularity Tahoe suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called ‘slow-start’. The reason for this procedure is that an initial burst might overwhelm the network and the connection might never get started. A slow start suggests that the sender set the congestion window to 1 and then for each ACK received it increase the CWD by 1. so in the first round trip time(RTT) we send 1 packet, in the second we send 2 and in the third we send 4. Thus we increase exponentially until we lose a packet which is a sign of congestion. When we encounter congestion we decreases our sending rate and we reduce congestion window to one. And start over again. The important thing is that Tahoe detects packet losses by timeouts. In usual implementations, repeated interrupts are expensive so we have coarse grain time-outs which occasionally checks for time outs. Thus it might be some time before we notice a packet loss and then re-transmit that packet.

1) Congestion Avoidance

For congestion avoidance Tahoe uses ‘Additive Increase Multiplicative Decrease’. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as a threshold value. It then set CWD to one and starts slow start until it reaches the threshold value. After that it increments linearly until it encounters a packet loss. Thus it increase it window slowly as it approaches the bandwidth capacity.

2) TCP RENO

TCP instrument is like to TCP Reno Tahoe excluding that it supports improvements addition Tahoe Phase of quick revival algorithm called fast food (Hoe 1996). important development of TCP Reno, in difference to TCP Tahoe, to avoid the pathway of message "pipe" blank after fast retransmission, and thus avoids gradually begin to fill up again after a packet failure TCP Reno keeps information chart, with copy ACK is cheaper than the TCP Tahoe. In this way, TCP lowered its liability in the center, without relax for a epoch of delay timepiece between data and acknowledgments. This development important collision on bandwidth long delay the associates if the slow start lasts longer and larger Windows are required to accomplish best through position. If a particular packet is missing, a window of data, TCP Reno makes a device for quick revival, but when several packets are lost, Rhine profit are the same here than Tahoe. This means that when multiple packets are lost the same window, TCP Reno; about directly copy ' quick improvement and end until the new package can be sent.

The above discussion leads to the conclusion that a speedy recovery method introduced by TCP Reno manages several packages the losses within a window bad.

3) TCP SACK

It is an upgraded version (EV) of TCP Reno and New careful Acknowledgment Reno. The two most significant Problems, the acknowledgment of several lost packets and the communication of more than one lost packet per round trip the conditions can be determined over TCP SACK. In this protocol, the packages are accepted individually, but as cumulative. This application allows a TCP receiver to send SACK data and a sequence of options in TCP headers TCP ACK to the previous. The new issue is showing the first event and the last assigned non-contiguous data proceedings to the receiver. According to other TCP options used by the connection, a greatest of two or three blocks of data They may be reported by a single ACK. Recipients SACK The information in the TCP header when there are only two awards reply to the coming of an outside application package. A quantity template is to introduce new to the sender to maintain Sack length of this data and a new mechanism that is used for pipe track the number of packages presently in transfer (Example the "The data pipe network). SACK distribute as soon as New Reno. He enters the fast retransmit phase, when a loss of documented, and ends when all data is known, was excellent, when the fast retransmit Phase began.

SACK at rest makes no effort to distinguish between losses of by congestion and faults on the wireless association (which makes the loss of impact reduction) in performance. Also It seems that the accomplishment of presently designed SACK, there are still situation in which potentially is, if the packets are lost in very exact patterns [5].

4) TCP FACK

TCP SACK with assault Recognition as TCP FACK recognized. Use TCP FACK is approximately the same to the bag, but it will be a Rated modest enhancement, so that you do. Use the selection to bag a better approximation of the quantity of data in transfer. FACK TCP is a better way of reducing through half the window when blocking is detects. If cwnd instantaneously halve, the sender will stop program of a sure time, then carry on as soon as sufficient data has left he network. This rough Distribution of segments in one RTT can be avoided if the window is steadily abridged. When congestion occurs, the window should be halving after moribund in real terms cwnd multiplicative. While the sender is identified at least one RTT to blockage if during that RTT is ready to start slow, then cwnd current almost twice as high among CWnd shortage occurred. Therefore, in this case is first cwnd halved to estimate the suitable CWnd that require more decrease [26]

5) TCP New Reno

Hoe is the new version of the TCP Reno is identified as TCP New Reno. It is a modest special TCP Reno fast recovery algorithm. New Reno is capable when there are several packet losses.[1]

Reno and New Reno , both position to go to the fast retransmit when multiple duplicate packets received except otherwise later from third rapid recovery segment pending all exceptional data, when was documented early recovery. This means that in New Reno did not agree to part of the TCP ACK rapid recovery, but to be treated as a mark that the package now recognized by the package in the control scope is missing, and must be distributed. Therefore, when several packets are lost from one window of data in this New Reno without broadcasting time in the time to advance, Losses to retransmit packets per hour, round trip all packets that were lost through the window transferred. There is a speedy recovery when inject all data network and is still to be complete association launched by the speedy recovery was acceptable.[3]

The central matter is that TCP New Reno is clever to commerce with several packet losses in one window. Is discovery and come back lost more than a incomplete package Round-trip time; this insufficiency is more obvious than bandwidth wait. More importantly, there are situations where jobs can still happen when packets are lost in successive windows. Like all earlier versions of discussed earlier, TCP New Reno is in front, all is lost the packets are caused by congestion, and therefore could avoidable size of the congestion window of the segment, when errors happen.

6) TCP Vegas

Vegas algorithm predicts the start of overcrowding observe the dissimilarity amid predictable and genuine value. Vegas blocking window to adjust the transfer resource rates in an effort to keep a few packets in the buffer routers along the transmission line. TCP Vegas channel is the present value of the system clock for each segment sent. It is therefore able to know the RTT for each specific Packet sent. TCP Vegas have been following the current evolution(Lai and Yao 2002).

According on (Hasegawa, Kurata et al. 2000). TCP Vegas controls its window size by watch the RTT (Round Trip Time) of packet that the dispatcher sent home by. If RTT experiential to befall large, TCP Vegas recognize that the network is becoming congested and drowns the size of the window. If RTT become weak, moreover, decide the host sending TCP Vegas, the network of free from from the congestion and increase the window size again. Therefore, the window size in an ideal state is to give join to an appropriate value.

III. SIMULATION AND RESULTS

A. Simulation Parameters:

TCP parameters	Value
Slow start initial count	1
Receive buffer size (bytes)	8,760
Maximum ACK segment	2
Duplicate ACK threshold	3
Initial RTO (seconds)	1.0
Minimum RTO (seconds)	0.5
Maximum RTO (seconds)	64
RTT gain	0.125
Deviation gain	0.25

B. Simulation results:

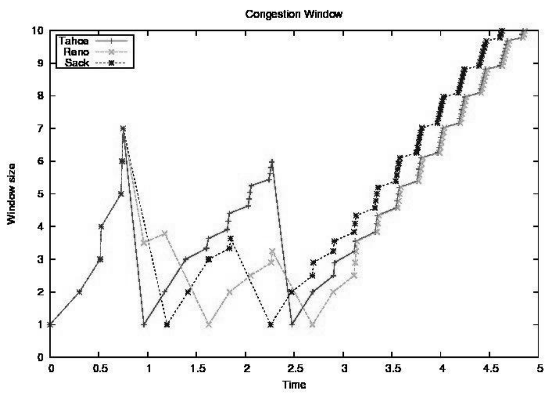


Fig 4: Plot of time Vs window size

Figure shows the plot of time vs. window size. We can see that the window size variation in TAHOE is maximum among all three TCP variants TCP TAHOE, TCP RENO and TCP SACK. Also the observation can be done that the RANO and TAHOE behaves same as the time of the network increases.

Figure shows the plot of congestion window size vs. time. It shows the the progression of the congestion window with time and the number of packets transmitted for all the three TCP variants namely TAHOE, RENO and SACK.

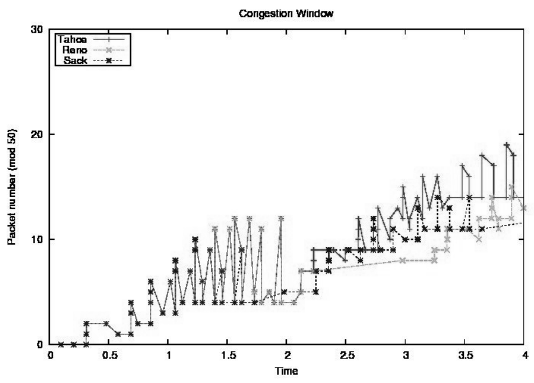


Fig. 5: Plot of congestion window size vs. time

C. Observation:

From the plots it can be observed that the time taken by the SACK version of TCP to overcome the congestion is less compared to others. This difference is seen clearly in the plot of window size. In the plot of packet numbers, the SACK overpowers the RENO but is not able to cope with the TAHOE version.

IV. CONCLUSION

In this paper we discussed the different version of TCP, which is TCP variants includes Tahoe, Reno, New Reno, SACK, and FACK. We explained the different using between both of them and the performance, efficiency for every one of TCP variants. We concluded that the main problem is traffic congestion Different variants of TCP. By increasing the flow of Traffic jams are reduced. Tahoe performance is slow occurs when a packet loss. Reno behaves so badly Tahoe when there is loss of multiple packets in the window. New Reno is the detection and allocate it has lost more than a limited number of Round Trip Time package We found the fundamental limitations imposed by the lake selective TCP acknowledgment. This difference is seen in the congestion window progression plot clearly.

REFRENCES

[1] V.Jacobson “Congestion Avoidance and Control” SIGCOMM Symposium no Cummunication Architecture and protocols.
[2] V.Jacobson “Modified TCP Congestion Control and Avoidance Alogrithms”. Technical Report 30, Apr 1990.
[3] S.Floyd, T.Henderson “The New Reno Modification to TCP’s Fast Recovery Algorithm” RFC 2582, Apr 1999.
[4] O. Ait-Hellal, E.Altman “Analysis of TCP Reno and TCP Vegas”.
[5] K.Fall, S.Floyd “Simulation Based Comparison of Tahoe, Reno and SACK TCP”.
[6] L.S.Brakmo, L.L. Peterson, “TCP Vegas: End to End

Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communication, vol. 13[1995], (1465-1490).

[7] A. Ahuja, S. Agarwal, J. P. Singh and R. Shorey, "Performance of TCP over different Routing Protocols in Mobile Ad-hoc Networks," IEEE Vehicular Technology Conference 2000, vol. 3, pp. 2315 - 2319, Tokyo.

[8] I. Ali, R. Gupta, S. Bansal, A. Misra, A. Razdan and R. Shorey, "Energy Efficiency and Throughput for TCP Traffic in Multi-Hop Wireless Networks," IEEE INFOCOM'02, New York, 2002.

[9] V. Anantharaman, S.-J. Park, K. Sundaresan, and R. Sivakumar, "TCP Performance over Mobile Ad-hoc Networks: A Quantitative Study," 'To appear in Wireless Communications and Mobile Computing Journal (WCMC), Special Issue on Performance Evaluation of Wireless Networks, 2003.

[10] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani and R. D. Gitlin, "AIRMAIL: a link-layer protocol for wireless networks," ACM Wireless Networks, Feb. 1995.

[11] H. Balakrishnan, V. Padmanabhan, S. Seshan and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," Proceedings of ACM SIGCOMM'96, Aug. 1996.

[12] H. Balakrishnan, S. Seshan and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," ACM Wireless Networks, Dec. 1995.

[13] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," Proc. 15th International Conf. On Distributed Computing systems (ICDCS), May 1995.

[14] P. Bhagwat, P. Bhattacharya, A. Krishna and S. K. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," IEEE INFOCOM'96, San Francisco, March 1996

[15] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," ACM computer communication review, vol. 27, no. 5, Oct. 1997

[16] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," IEEE JSAC. Vol.19 No.7, July 2001.

[17] M. C. Chan, R. Ramjee, "TCP/IP performance over 3G wireless links with rate and delay variation," MobiCom'02, Sep. 2002.

[18] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," IEEE Personal communications, 8 (1):34-39, February 2001.

[19] K. Chen, Y. Xue, K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks," IEEE ICC'03, Anchorage, Alaska, May, 2003.

[20] A. DeSimone, M. C. Chuah and O. C. Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs," Proc. Globecom '93, Dec. 1993.

[21] T. D. Dyer and R. V. Boppana, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks," ACM Mobihoc, October 2001.

[22] S. Floyd, K. Fall, ``Router mechanisms to Support end-to-end congestion control", LBL Technical report, February 1997.

[23] S. Floyd, V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transaction on Networking, vol. 1, no. 4, Aug. 1993.

[24] Z. Fu, P.Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", IEEE INFOCOM'03, San Francisco , March 2003.

[25] Z. Fu, X. Meng, and S. Lu, "How Bad TCP can Perform in Mobile Ad-Hoc Networks," IEEE Symposium on Computers and Communications, Italy, July 200233rd Hawaii International Conference on System Sciences (HICSS '00), January 2000.

[26] Mathis, M. and J. Mahdavi (1996). "Forward acknowledgement: Refining TCP congestion control." ACM SIGCOMM Computer Communication Review 26(4): 281-291.

[27] Hasegawa, G., K. Kurata, et al. (2000). Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet.