International Journal of Modern Engineering Research (IJMER)

www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-1039-1045 ISSN: 2249-6645

Study of TCP Variants Compression on Congestion Window and Algorithms in Dynamic Environment

Nishant Chaurasia
M.Tech Scholar in CSE
Oriental institute of science and technology
Bhopal, India

Prof. Sanjay Sharma
Asst. Professor in CSE Department
Oriental institute of science and technology
Bhopal, India

***Abstract*: The most common transport protocol is the Transmission Control Protocol. Transmission Control Protocol (TCP) is reliable, connection oriented transport protocol which provides end-to-end data delivery in networks. TCP was primarily designed for wired networks and it shows performance degradation when used in wireless networks. The main concern in wireless networks with TCP is the packet loss which is mostly considered to have occurred due to congestion. The Transmission Control Protocol comes in many variants like TCP, Tahoe, Reno, New Reno, Vegas, Sack and so on. In a Mobile Ad Hoc Network or Dynamic Environment, temporary link failures and route changes happen frequently. With the assumption that all packet losses are due to congestion, TCP performs poorly in such environment. While there has been some research on improving TCP performance over Dynamic Environment, most of them require feedback from the network or the lower layer. In this paper we constitute a realistic Dynamic Environment by considering multistage error model in the design of wireless packet losses. Mobility in structure is also conceived to estimate the actual performance of TCP. Moreover the behaviour of TCP Tahoe, Reno, New Reno, Sack, Vegas, fack, lite and west-wood in Dynamic network is simulated to perceive the impact of wireless link on the behaviour of these TCP variants. Finally from the result of our simulation we conclude the best TCP variants for different circumstances and detecting or responding to out-of-order packet delivery events, which are the results of frequent route changes. In our simulation study, this approach had achieved best performance improvement, without requiring feedback from the network.**

Keywords: TCP Features, Advantages, TCP Variants, TCP algorithms and characteristics.

I. INTRODUCTION

Transmission Control Protocol (TCP) was originally defined in RFC 793 [1]. TCP is reliable, connection oriented transport protocol which provides end-to-end data delivery in networks. TCP was primarily designed for wired networks and it shows performance degradation when used in wireless networks. TCP is a window based reliable transport layer protocol that achieves its reliability through sequence numbers and acknowledgements [2]. TCP assumes that all the packet losses are due to congestion. When a packet is lost, TCP applies congestion avoidance mechanisms and slows its transmission rate. However, wireless networks are known to experience sporadic and usually temporary losses due to fading, shadowing, hand off that cannot be considered congestion. Packets can be lost due to hand offs as a mobile node moves out of range of a base station and into the range of another packet lost during such transitions also initiate TCPs congestion avoidance. TCP is based on the principle of “conservation of packets”, which means that in the case a connection works at the available capacity of bandwidth, the packet is not to be inserted into the network until the second packet doesn’t leave the network [3]. TCP implements the above principle by using the acknowledgements to time the outgoing packets, because the acknowledgement means that the specific packet has left the network.
TCP maintains the congestion window to represent the network capacity. The transmitter can send data up to the minimum value of congestion window and advertised window. Congestion control is the flow control imposed by the transmitter, while the advertised window is the flow control imposed by the receiver. The first control is based on the transmitter’s perception of network congestion, while the second is related to the size of available space in the buffer at the receiver for the given connection.
TCP operates in three phases:
1. Connection establishment
2. Data transfer
3. Connection termination

The basic implementations of TCP are based on Jacobson’s classical slow start algorithm for congestion avoidance and control [4- 5]. A number of solutions have been proposed to remove the problem of congestion. This paper performs analysis on the variants of TCP that have evolved for performance improvement in wireless networks.

II. TCP FEATURES AND ADVANTAGES

TCP provides a connection oriented, reliable, byte stream service. The term connection oriented means the two applications using TCP must establish a TCP connection with each other. It is a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction. TCP includes a flow-control mechanism for each of these byte streams that allow the receiver to limit how much data the sender can transmit. TCP also implements a congestion-control mechanism. TCP is a reliable connection oriented end-to-end protocol which has many mechanisms to provide reliable communication. But a small number of packets are lost due to congestion and buffer overflow. In such cases, TCP ensures reliability by using sequence numbers and time-out intervals. The packet of the particular sequence number is resent after the time-out timer runs out. TCP runs on the concept of “Conservation of Packets” [4]. The TCP provides different facilities as discussed below in the following list.

A. Stream Data Transfer
TCP transfers a continuous stream of bytes. TCP does this by grouping the bytes in TCP segments, which are passed to IP for transmission to the destination. TCP decides how to segment the data and forwards the data at its own convenience.

B. Reliability
TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments.

C. Flow Control
The receiving TCP, when sending an ACK back to the sender, also indicates to the sender the number of bytes it can receive beyond the last received TCP segment, without causing overrun and overflow in its internal buffers. This is sent in the ACK in the form of the highest sequence number it can receive without problems.

D. Multiplexing
To allow for many processes within a single host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the internet communication layer this forms a socket. A pair of sockets uniquely identifies each connection.

E. Logical Connection
The reliability and flow control mechanisms described above require that TCP initializes and maintains certain status information for each data stream. The combination of this status, including sockets, sequence numbers and window sizes, is called a logical connection. Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes.

F. Full Duplex
TCP provides for concurrent data streams in both directions.

III. TCP VARIENTS AND COMPARISION

A. TCP
TCP provides important features of flow control, reliability, congestion control and connection management. Originally, TCP designed for wired networks but it also performs well in wireless networks. In order to improve its performance TCP cuts down the size of its congestion window resulted in further performance degradation. This is a more serious problem in busty and highly mobile networks which have rapid topological changes (Henna, 2009)[9]. TCP provides division for sequenced data stream into packets, confirms the packets delivery with the possibility of losing the IP layer loses, retransmit, reorders, or packets duplication and monitoring the network band capacity to avoiding congestions. TCP protocol can provide over two end points connection, flow rate controlling with bidirectional link and data reliability.

B. TCP Tahoe
TCP Tahoe is the congestion control mechanism suggested by Van Jacobson. The actual TCP data transmission is clocked by the acknowledgements received. But at the start of the transmission, there would not be any acknowledgement. To overcome this, the Tahoe suggests a mechanism called “slow start”. According to this mechanism, the congestion window size is taken as 1 at the beginning of start or a restart of data transmission. After sufficient acknowledgements are received, the congestion window size is additionally increased. After congestion is achieved, the window size is multiplicatively decreased. This is called Additive Increase Multiplicative Decrease [5]. Whenever a packet is lost, the “go back n” method is used, and the entire pipe is emptied. This results in a high bandwidth delay.

C. TCP Reno
TCP Reno has all the advantages of Tahoe like the slow start mechanism and the time-out intervals. Also, it has some intelligent mechanisms to detect the packet losses previously. After each packet loss, the entire pipe is not emptied. It uses a Fast Retransmit mechanism in which when 3 duplicate acknowledgements are received, it is understood that there is packet loss. Hence even before the actual packet loss is detected, the packet is retransmitted. It has the disadvantage of reducing the window size more than required and hence cannot afford Fast Recovery [6]. If window size is reduced very much, then the normal course grained timeout.

D. TCP New Reno
The TCP New Reno [7] is more advanced than TCP Reno. It is able to detect multiple packet losses. It also enters the fast recovery mechanism like Reno, but it does not end up reducing the congestion window size. It waits till the acknowledgements of all the congested packets are received. The actual disadvantage of the New Reno is that it takes a whole Round Trip Time to detect a single packet loss.

E. TCP Vegas

The TCP Vegas [8] is a modified version of Reno. It works on the proactive measures to control congestion rather than reactive measures. It uses an algorithm to check for timeouts. It also overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss. It also uses a slightly modified slow start mechanism. It has the mechanism to detect congestion even before packet losses occur, but it also retains the other mechanisms of Reno and Tahoe. Overall, the Vegas has a new retransmission mechanism, a modified slow start algorithm and congestion avoidance scheme.

F. TCP Sack:

TCP with selective acknowledgment (Sack) permits the receiver of data to openly acknowledge the data in out of order which arrived to data sender. If Sack is used, the TCP sender does not resend the data Sacked through the period of loss recovery. Many of research proved that Sack technique enhance TCP throughput if multiple packet loss happen during same window (Ekiz, *et al*., 2011). Sack algorithm is a mutual between selective duplication resending strategy, has been suggested to overcoming the limits and with accumulative acknowledgment structure for TCP (Kettimuthu and All cock, 2004). TCP with Sack is behaving more easily to understand than other two algorithms, Tahoe and Reno.

TC Sack needs that packets not acknowledging accumulatively but must acknowledging in selective manner because of that every ACK includes a block that defines each segment if acknowledged. So, TCP sender has an image of the acknowledged segments and the segments that outstanding. Every time TCP sender go in fast recovery phase, it sets a mutable pipe that is determine the amount of data is still outstanding in the path of the network and fix the congestion window to half of the recent value. Whenever it accepts an acknowledgment it decreases the pipeline by one and for each it resends a segment it increases it by one. When the pipeline is going to less than congestion window size, it detects the segments which are still not received and resend them. If no segments in outstanding situation, then it will send new packets, therefore more than single segment losses can be able to send within single RTT. The major problematic with implementation of TCP Sack is that presently selective acknowledgement does not deliver via the receiver and to implementing TCP Sack it not very easy process, but it precise and complicated task.

G. TCP Fack

TCP with forward acknowledgement (Fack) is a different algorithm which works on upper options of TCP Sack. TCP Fack is use info providing via Sack to adding extra accurate control to the data injection in to the pipe of network within during recovery process. The basic concept of Fack mechanism is by considering the greatest sequence number of forward selective acknowledgement as a mark that completely previous segments which unselectively acknowledged were lost. This monitoring permits to improve the recovery process of packets losses meaningfully. Fack algorithm is taking a more violent methodology and considering unacknowledged holes among lost packets and Sack blocks. This methodology frequently outcomes improved TCP performance than the traditional approach, it is excessively violent if packets have been rearranged in the pipeline, due to these holes between blocks of Sack does not designate packets loss in this state (Sarolahti and Kuznetsov, 2002). The congestion window of TCP Fack is illustrated in figure 6, where a different behavior of the adjusting the window size. Fack presents a good technique to halving the size of window if the congestion occurred. If cwnd is instantly halved, TCP sender breaks transferring for a while and then restarts if the sufficient amount of data leaving the network. If the congestion happens, the window size must be halved depending on the multiplicative reduction of the exact cwnd. The sender recognizes the congestion state after it happened at least single RTT and if through that RTT in slow start phase, then the recent value of cwnd will duplicated than previous value if when congestion happened. So, in this state, the congestion window is firstly halved to determine the accurate cwnd which must be further reduced. However, TCP Fack offers congestion avoidance and fast retransmit mechanisms, but it aspects a lot of circumstances in recovery processes and also is not easy to implement Fack over applications (Tayade, 2011).

H. TCP Lite

TCP Lite is a service that provides a transport method that interrupts TCP in order to reduce the overhead involved in session management in which no data is transmitted or received. TCP Lite reduces or eliminates pure TCP protocol data units used in the set up and ACK while maintaining order, integrity, reliability and security of traditional TCP. TCP lite uses big window and protection against wrapped sequence number. Lite performs over TCP same as Reno. But when window increases it have some problems to
Maintain them.

I. TCP West-wood

The TCP Westwood (TCPW) is a sender-side-only modification to TCP New Reno that is intended to better handle large bandwidth-delay product paths, with potential packet loss due to transmission or other errors, and with dynamic load.TCP Westwood protocol relies on a simple modification of the TCP source protocol behaviour for a faster recovery. This is performed by setting both a slow start threshold and a congestion window values that result from the effective connection while congestion is experienced. Hence, TCPW attempts to make a more “informed” decision, in contrast with TCP Reno, which automatically halves the congestion window after three duplicate ACKs. Like TCP Reno, TCPW cannot distinguish between buffer overflow losses and random losses. However, in presence of random losses, TCP Reno overreacts and reduces the window by half. TCP West Wood cannot distinguish between buffer overflow and random losses. It does not provide fast recovery mechanism for data packet or ACK.

IV. VARIENTS SIMULATIONS

The plots of the congestion window for all TCP variants are following. The values of cwnd are taken from a simulation runtime of 200 seconds with the presence of packet error rate of 10-3. However for better understanding of the behaviour of congestion window of each TCP variants, the value of the congestion window is traced for an interval of every second.

Fig 3(a): Congestion window dynamics of TCP Tahoe

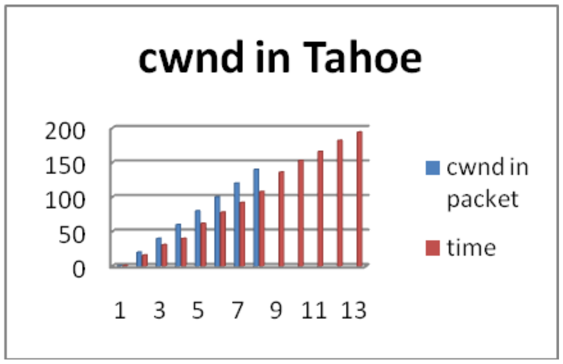


Fig 3(b): Congestion window dynamics of TCP Reno

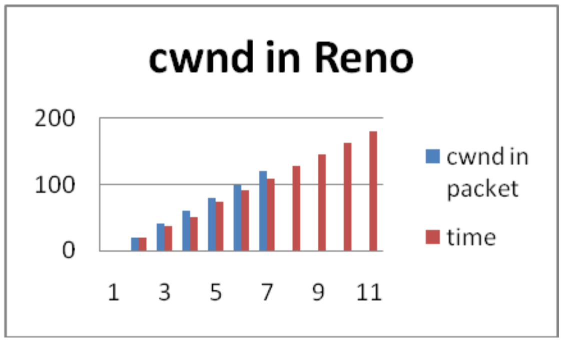


Fig 3(c): Congestion window dynamics of TCP New Reno

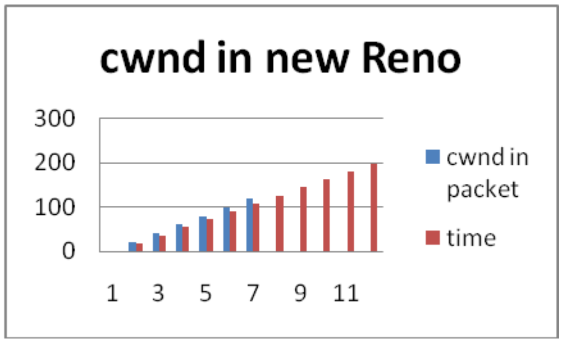


Fig 3(d): Congestion window dynamics of TCP Sack

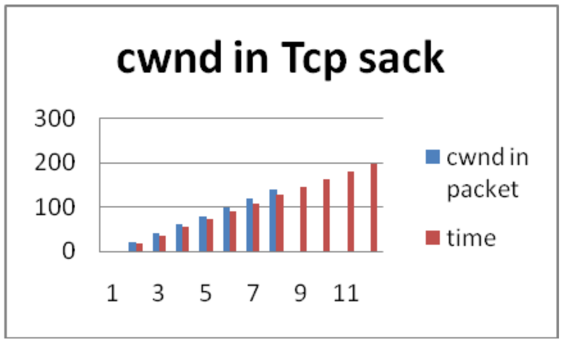


Fig 3(e): Congestion window dynamics of TCP Vegas

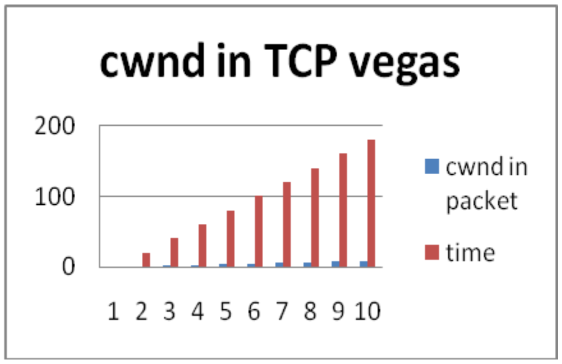


Fig 3(f): Congestion window dynamics of TCP fack

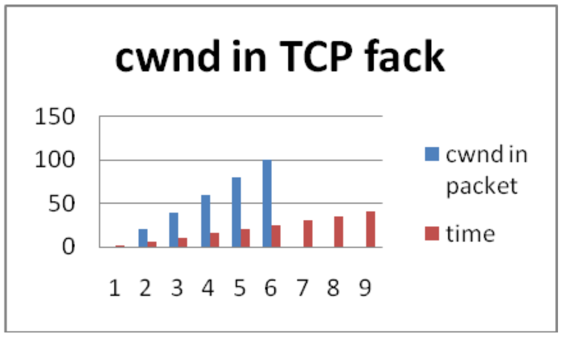


Fig 3(g): Congestion window dynamics of TCP lite

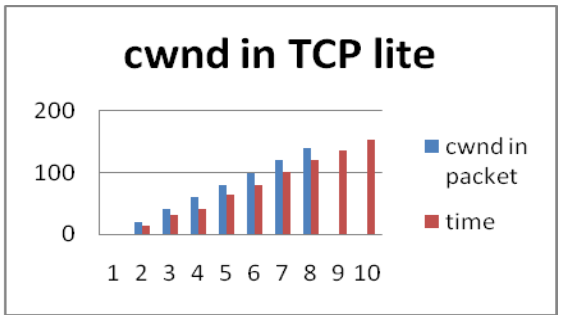
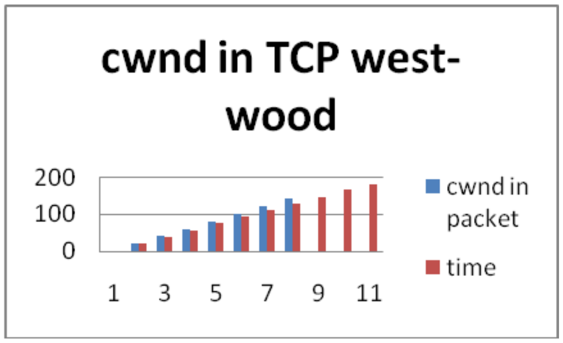


Fig 3(h): Congestion window dynamics of TCP west-wood



V. TCP ALGORITHMS AND CHARACTERSTICS

A. *Slow Start*

It operates by observing that the rate at which new packets should be injected into the network is the rate at which the acknowledgments are returned by the other end. Slow start adds another window to the sender's TCP: the congestion window, called "cwnd". When a new connection is established with a host on another network, the congestion window is initialized to one segment (i.e., the segment size announced by the other end, or the default, typically 536 or 512). Each time an ACK is received, the congestion window is increased by one segment. The sender can transmit up to the minimum of the congestion window and the advertised window. The congestion window is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver. The former is based on the sender's assessment of perceived network congestion; the latter is related to the amount of available buffer space at the receiver for this connection. The sender starts by transmitting one segment and waiting for its ACK. When that ACK is received, the congestion window is incremented from one to two, and two segments can be sent. When each of those two segments is acknowledged, the congestion window is increased to four. This provides an exponential growth, although it is not exactly exponential because the receiver may delay its ACKs, typically sending one ACK for every two segments that it receives. At some point the capacity of the internet can be reached, and an intermediate router will start discarding packets. This tells the sender that its congestion window has gotten too large. Early implementations performed slow start only if the other end was on a different network. Current implementations always perform slow start.

B. *Congestion Avoidance (Additional Increase)*

If the receiver window is large enough, the slow start mechanism described in the previous routers in between the hosts will start discarding packets. As mentioned earlier TCP interprets packet loss as a sign of congestion, and when this happens TCP invokes the Congestion Avoidance mechanism [10]. Even though slow start and congestion avoidance is two different mechanisms they are more easily described together.

In the joint description below a new TCP variable is introduced. This variable, ssthresh, is the slow start threshold which TCP uses to determine if slow start or congestion avoidance is to be conducted.

1. When establishing a new connection cwnd is initialized to 0 < cwnd <= min (4*MSS, Max (2*MSS, 4380bytes))
2. The sender side TCP sends a maximum of in (cwnd, rwnd) bytes
3. When congestion occurs Ssthresh <-min (min (cwnd, rwnd) / 2, 2* MSS). If congestion was due to a timeout slow start is conducted.
4. When new data is acknowledged by the other end cwnd is increased. The way in which TCP increases the cwnd depends On if we are doing slow start (cwnd<ssthresh) or congestion avoidance The increase of cwnd in slow start was described in the previous Section, and if we are doing congestion avoidance then cwnd <- cwnd + (1/cwnd).

C. *Fast Retransmit*

TCP may generate an immediate acknowledgment (a duplicate ACK) when an out- of-order segment is received. This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order, and to tell it what sequence number is expected. Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire.

D. *Fast Recovery*

After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows. The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start. The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached.

2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.

3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step.

This ACK should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost.

E. Selective Acknowledgment

Multiple packet losses from a window of data can have a catastrophic effect on TCP throughput. Multiple dropped segments generally cause TCP to lose its ACK-based clock; reducing overall throughput. The Selective Acknowledgment (SACK) [11] is a strategy which corrects this behaviour in the face of multiple dropped segments. With selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost. It allows the receiver to acknowledge discontinuous blocks of packets that were received correctly, in addition to the sequence number of the last contiguous byte received successively. The acknowledgement can specify a number of SACK blocks, where each SACK block is conveyed by the starting and ending sequence numbers of a contiguous range that the receiver correctly received.

F. Flow Control

In computer networking, flow control is the process of managing the data rate between two nodes to prevent a fast sender from outrunning a slow receiver. It provides mechanism for the receiver to control the transmission speed, so that it is not overwhelmed. Flow Control should be distinguished from congestion control, which is used for controlling the flow of data when congestion has occurred actually. In a connection between a client and a server, the client tells the server the number of bytes it is willing to receive at one time from the server; this is the client's receive window, which becomes the server's send window. Likewise, the server tells the client how many bytes of data it is willing to take from the client at one time; this is the server's receive window and the client's send window. Since the window size can be used in this manner to manage the rate at which data flows between the devices at the ends of the connection, it is the method by which TCP implements flow control, one of the “classical” jobs of the transport layer. Flow control is vitally important to TCP, as it is the method by which devices communicate their status to each other. By reducing or increasing window size, the server and client each ensure that the other device sends data just as fast as the recipient can deal with it. Flow control is a technique whose primary purpose is to properly match the transmission rate of sender to that of the receiver and the network. It is important for the transmission to be at a high enough rates to ensure good performance, but also to protect against overwhelming the network or receiving host. Congestion control is primarily concerned with a sustained overload of network intermediate devices such as IP routers. TCP uses the window field, briefly described previously, as the primary means for flow control. During the data transfer phase, the window field is used to adjust the rate of flow of the byte stream between communicating TCPs.

G. Retransmission Mechanism

Retransmission Mechanism [12] keeps track of when each segment was sent and it also calculates an estimate of the RTT by keeping track of how long it takes for the acknowledgment to get back. Whenever a duplicate acknowledgement is received it checks to see if the (current time segment transmission time)> RTT estimate; if it is then it immediately retransmits the segment without waiting for 3 duplicate acknowledgements or a coarse timeout [12]. Thus it gets around the problem faced by Reno of not being able to detect lost packets when it had a small window and it didn’t receive enough duplicate ACK’s. To catch any other segments that may have been lost prior to the retransmission, when a non duplicate acknowledgment is received, if it is the first or second one after a fresh acknowledgement then it again checks the timeout values and if the segment time since it was sent exceeds the timeout value then it re-transmits the segment without waiting for a duplicate acknowledgment [12]. Thus in this way Vegas can detect multiple packet losses.

H. Congestion Avoidance:

Congestion avoidance is the algorithm used by TCP to avoid losing packets, if packets are lost. TCP performs congestion avoidance [4,8,12] when cwnd is greater than ssthresh. In the congestion avoidance phase, the cwnd is increased by 1 full-sized segment every round-trip time (RTT). Congestion avoidance continues until congestion is detected.

Congestion can be detected in two ways:

1) Receipt of duplicate acknowledgment

2) Due to time timeout

VI. CONCLUSION

Transmission Control Protocol was designed initially for wired networks it results in performance degradation when used in wireless networks. Transmission Control Protocol is responsible for reliable transport and regulation of data flow from source to destination. The primary reasons for performance degradation are packet losses, link failures, hand offs and long round trip time.

In this research, the performance of five different clones of TCP (Tahoe, Reno, NewReno, Sack, Vegas, Fack, Lite, West-wood) have been analyzed in the presence of high bit error. We can say that as the distance between the source and destination increases the delay time also increases and hence the total throughput starts reducing so to improve the throughput we increase the window size. Therefore on the WAN when the distance increases between the source and destination the throughput starts to reduce and hence to increase and support the large number of users we increase the window size. The effect of mobility (with incorporated Handoff) on the performance of TCP has been simulated. From the analysis, it is seen that the throughput of TCP degrades with increasing the speed of the mobile station. We have also analyzed the behaviour of these five TCP clones in the presence of both high bit error and handoff. From the analysis, it is observed that the throughput of these five TCP clones seriously degrades because of heavy packet losses due to both handoffs and high bit error.

VII. FUTURE WORK

In future, the efficiency of the TCP agents can be studied after introducing some amount of mobility to the nodes and the second is to analyze behaviour of TCP in cellular mobile environment considering coverage, battery power, and other impacts that are found in cellular mobile environment.

ACKNOWLEDGMENT

I am deeply thankful to my Guide Prof. Sanjay Sharma of Computer Science and my all friends whose help, stimulating suggestions and encouragement helped me in all the time for my review.

REFERENCES

[1] Postel J.,“Transmission Control Protocol”, RFC793, Internet Request for Comments 793, Sept. 1981.

[2] Stevens, Richard W.,“TCP/IP Illustrated”, 2001.

[3] Barakat C., Altman E., Dabbous W.,“On TCP performance in a heterogeneous network: a survey”, 1999.

[4] W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994

[5] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," Computer Communication Review, vol. 26, pp. 5--21, July 1996.

[6] Jitendra Padhye , Victor Firoiu , Donald F. Towsley , James F. Kurose, “Modeling TCP Reno performance: a simple model and its empirical validation,” IEEE/ACM Transactions on Networking (TON), v.8 n.2, p.133-145, April 2000

[7] S.Floyd, T.Henderson “The New-Reno Modification to TCP‟s Fast Recovery Algorithm” RFC 2582, Apr 1999.

[8] Lawrence S. Brakmo and Larry L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet,"Ekiz, N., A.H. Rahman and P.D. Amer, 2011. Misbehaviors in TCP SACK generation. ACM SIGCOMM Computer Communication Review, 41(2): 16-23.

[9] Henna, S., 2009. A Throughput Analysis of TCP Variants in Mobile Wireless Networks, pp: 279-284.IEEE.

[10] W. Stevens, “TCP Slow Start, Congestion Avoidance Fast Retransmit Algorithm”**,** IETF RFC 2001, January 1997.

[11] Lawrence, S. Brakmo, Student Member IEEE and Larry L. Peterson“TCP Vegas end congestion avoidance on a Global Internet, October 1995.

[12] S.Floyd, T.Henderson **“**The New- Reno Modification to TCP’s fast Recovery Algorithm**”**RFC 2582, Apr 1999.

AUTHORS PROFILE

Nishant Chaurasia born in 1987 in Gwalior, India. He awarded B.E in Computer Science and Engineering from the Rajiv Gandhi Technical University,Bhopal, India in 2009. From 2009-present, He has been M.Tech student in Computer Science and Engineering, at the Rajiv Gandhi Technical University,Bhopal, India.

Prof. Sanjay Sharma is an Assistant Professor of Computer Science Department in Oriental Institute of Science and Technology, Bhopal, Madhya Pradesh, India. He obtained his M.Tech degree from RGPV University. His research interest is in the field of MANETs.