Congestion Control Analysis over Wireless Ad hoc Networks

Firdous Ul Rashid
M.Tech (CSE)
SRM University
NCR Campus

Jitendra Singh
A.P (CSE)
SRM University
NCR Campus

Atendra Panwar
M.Tech (CSE)
SRM University
NCR Campus

Manoj Kumar
M.Tech (CSE)
SRM University
NCR Campus

Abstract

Transmission Control Protocol (TCP), the mostly used transport protocol, performs well over wired networks. As much as wireless network is deployed, TCP should be modified to work for both wired and wireless networks. Since TCP is designed for congestion control in wired networks, it cannot clearly detect non-congestion related packet loss from wireless networks. TCP Congestion control plays the key role to ensure stability of the Internet along with fair and efficient allocation of the bandwidth. So, congestion control is currently a large area of research and concern in the network community. Many congestion control mechanisms are developed and refined by researcher aiming to overcome congestion. During the last decade, several congestion control mechanisms have been proposed to improve TCP congestion control. The purpose of this paper is to analyze and compare the different TCP variants namely Reno, New Reno, Vegas under the AODV routing protocol. TCP's robustness is as a result of its reactive behaviour in the face of congestion, and fact that reliability is ensured by retransmissions.

KeyWords: MANET, TCP Congestion Control Algorithms, Reno, New Reno, Vegas.

1. Introduction

A mobile ad hoc network (MANET) is generally defined as a network that has many free or autonomous nodes, often composed of mobile devices or other mobile pieces that can arrange themselves in various ways and operate without strict top-down network administration. In MANET each node acts as a router and these networks are scalable. To support connectivity between nodes MANET networks use different kinds of protocols such as AODV, DSR, and DSDV etc.

2. Transmission Control Protocol

TCP stands for Transmission Control Protocol and it is a sliding window protocol that provides handling for both timeouts and retransmissions. TCP is known as a connection-oriented protocol. TCP is responsible for ensuring that a message is divided into the packets that IP manages and for reassembling the packets back into the complete message at the other end. In the OSI communication model, TCP is in layer 4, the Transport layer.TCP establishes a full duplex virtual connection between two endpoints.

3. TCP Congestion Control Algorithms

3.1 Slow Start: Slow start reduces the burst affect when a host first transmits. It requires a host to start its transmissions slowly and then build up to the point where congestion starts to occur. The host does not initially know how many packets it can send, so it uses slow start as a way to gauge the network's capacity. A host starts a transmission by sending two packets to the receiver. When the receiver receives the segments, it returns ACKs (acknowledgements) as confirmation. The sender increments its window by two and sends four packets. This build up continues with the sender doubling the number of packets it sends until an ACK is not received, indicating that the flow has reached the network's ability to handle traffic or the receivers ability to handle incoming traffic. Slow start does not prevent congestion; it simply prevents a host from causing an immediate congestion state. If the host is sending a large file, it will eventually reach a state where it overloads the network and packets begin to drop. Slow start is critical in avoiding the congestion collapse problem.

3.2 Congestion Avoidance: Congestion can occur when data arrives on a big pipe (a fast LAN) and gets sent out a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs. Congestion avoidance is a way to deal with lost packets. The assumption of the algorithm is that packet loss caused by damage is very small (much less than 1%), therefore the loss of a packet signals congestion somewhere in the network between the source and destination. There are two indications of packet loss: a timeout occurring and the receipt of duplicate ACKs. Congestion avoidance and slow start are independent algorithms with different objectives. But when congestion occurs TCP must slow down its transmission rate of packets into the network, and then invoke slow start to get things going again. In practice they are implemented together. Congestion avoidance and slow start require that two variables be maintained for each connection: a congestion window, cwnd, and a slow start threshold size, ssthresh.In Congestion Avoidance the window grows linearly.

3.3 Fast Retransmit: Duplicate ACKs that were mentioned to be one way of detecting lost packets can also be caused by reordered packets. When receiving one duplicate ACK the sender cannot yet know whether the packet has been lost or just gotten out of order but after receiving several duplicate ACKs it is reasonable to assume that a packet loss has occurred. The purpose of fast retransmit mechanism is to speed up the retransmission process by allowing the sender to retransmit a packet as soon as it has enough evidence that a packet has been lost. This means that instead of waiting for the retransmit timer to expire, the sender can retransmit a packet immediately after receiving three duplicate ACKs.

3.4 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 1). 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.

4. TCP Variants

4.1 TCP Reno: TCP Reno is the most widely adopted Internet TCP protocol. It employs four Congestion control Algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery .When packet loss occurs in a congested link due to buffer overflow in the intermediate routers, either the sender receives three duplicate acknowledgments or the sender’s retransmission timeout (RTO timer expires). In case of three duplicate ACKs, the sender activates TCP fast retransmit and recovery algorithms and reduces its congestion window size to half. It then linearly increases congestion window, similar to the case of congestion avoidance. This increase in transmission rate is slower than in the case of slow start and helps relieve congestion. TCP Reno fast recovery algorithm improves TCP performance in case of a single packet loss within a window of data. However performance of TCP Reno suffers in case of multiple packet losses within a window of data.

4.2 New Reno: New Reno is a modification of TCP Reno. TCP New Reno enhances TCP throughput performance when multiple packets are dropped from a single window of data for TCP Reno connections that does not support the TCP SACK option. When multiple packets are dropped from a single window of data, the ACK for the retransmitted packet acknowledges some but not all of the packets transmitted before the fast retransmit. This is referred to as partial ACK. During fast recovery when a TCP sender receives partial ACK, the TCP sender concludes that the indicated packets was lost and retransmit that packet. The remaining three phases (slow start, congestion avoidance, and fast retransmit) are similar to TCP Reno. A problem occurs with New Reno when there are no packet losses but instead, packets are reordered by more than 3 packet sequence numbers. When this happens, New Reno mistakenly enters fast recovery, but when the reordered packet is delivered, ACK sequence-number progress occurs and from there until the end of fast recovery, every bit of sequence-number progress produces a duplicate and needless retransmission that is immediately ACKed.

4.3 Vegas: Vegas [8] is a TCP implementation which is a modification of RENO. It builds on the fact that proactive measure to encounter congestion is much more efficient than reactive ones. It tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggests a modified slow start algorithm which prevents it from congesting the network.

The three major changes induced by Vegas are:

 New Re-Transmission Mechanism: Vegas extend on the retransmission mechanism of RENO. It 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.
 Congestion avoidance: TCP Vegas is different from all the other implementation in its behavior during congestion avoidance. It does not use the loss of segment to signal that there is congestion. It determines congestion by a decrease in sending rate as compared to the expected rate, as result of large queues building up in the routers.
 Modified Slow-start: TCP Vegas differs from the other algorithms during its slow-start phase. The reason for this modification is that when a connection first starts it has no idea of the available bandwidth and it is possible that during exponential increase it over shoots the bandwidth by a big amount and thus induces congestion. To this end Vegas increases exponentially only every other RTT, between that it calculates the actual sending throughput to the expected and when the difference goes above a certain threshold it exits slow start and enters the congestion avoidance phase.

5 Simulation Parameters

Parameters	**Values**
Simulator used	NS2.3
No of Nodes	5,10,15,20
Simulation Time	190 sec
Traffic	FTP
TCP Variants	Reno,NewReno, Vegas
Packet Size	512
Window Size	15
Routing Protocol	AODV
Queue Size	50

6 Performance Metrics

6.1 Throughput: - It is defined as the ratio of the total number of bits received by the destination to the total simulation time. It is measured in bits per second/Kbps/Packets per second.

TP = received packets/simulation time (kbps)

6.2 End to End Delay: - It is defined as the time taken for a packet to be transmitted across network from source to destination. The lower the delay the better is the performance.

6.3 Packet Delivery Ratio: - It is defined as the ratio of the total number of packets received by the destination node to the number of packets send by the source node.

7 Simulation Results

In this section, we present the results of our ns-2 simulations of the three TCP variants namely Reno, New Reno and Vegas under AODV routing protocol. This wireless simulation has been done for 190 seconds.

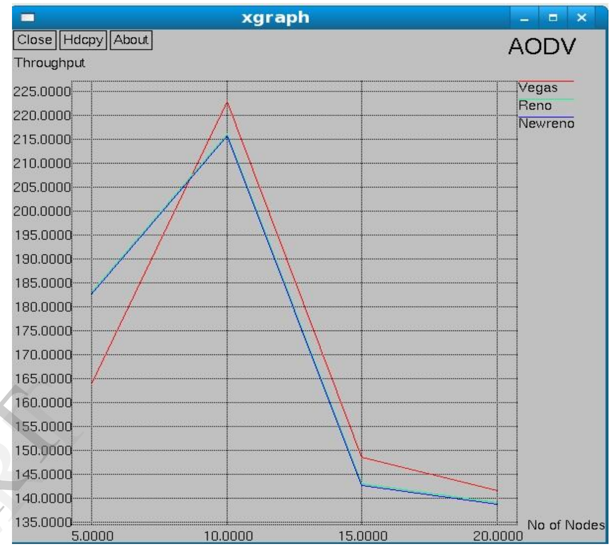

Fig1:No of Nodes vs Throughput

In fig1, we compare the TCP variants on the basis of Throughput and in the above graph we see that as the number of nodes increases the Throughput decreases in all the three TCP variants , but Vegas gives better Throughput as compared to Reno and New Reno

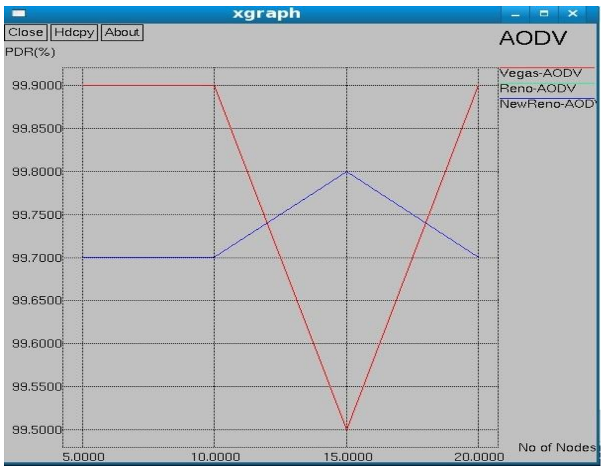

Fig2: No of Nodes vs PDR

In fig 2, the packet delivery ratio of Vegas increases as the node density increases and the PDR of TCP Reno and TCP New Reno decreases as the node density increases.

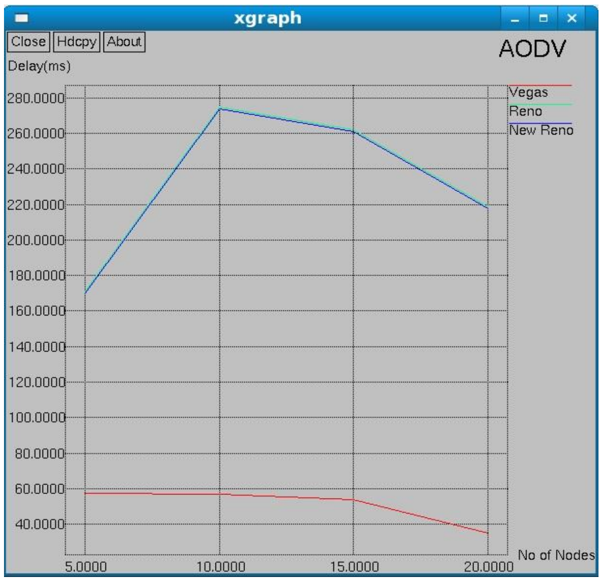

Fig3:No of Nodes vs Delay

Fig 3 prove the End to End Delay of TCP variants namely Reno, New Reno and Vegas under different node densities. From the experimented results the TCP Vegas shows less delay than Reno and New Reno.

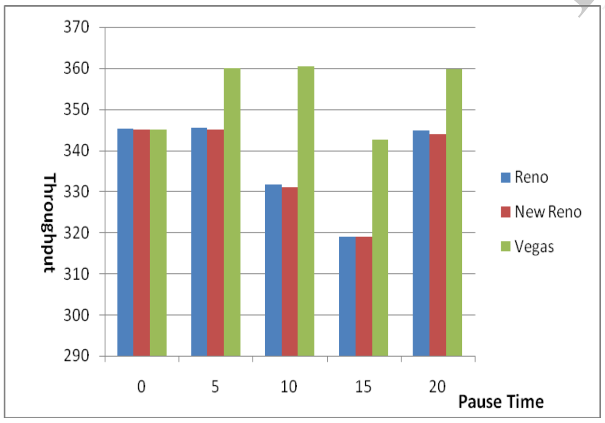

Fig 4: Pause Time vs Throughput

Fig 4 shows Throughput of different TCP variants under different Pause Time. It is observed that the TCP Vegas gives better Throughput than other TCP variants. Here the receiver receives maximum number of packets because of less delay as shown in fig 3, so we can conclude that here in TCP Vegas Throughput is better than Reno and New Reno.

8 Conclusion

We have successfully evaluated the three TCP variants using NS2 simulation tool in the Mobile Ad hoc Networks. The results are more significant and comparable. From the simulation results, we conclude that TCP Vegas is much better than TCP Reno and New Reno because the TCP Vegas provides good Throughput than other two TCP variants and the packet delivery ratio of Vegas is better than Reno and New Reno.

9 Acknowledgement

We are very thankful to Mr. Rakesh Kumar Yadav (AP CSE) SRM University Modinagar for their guidance , support , stimulating suggestions and encouragement with the help of which we were able to complete our research work.

10 References

[1] V. Jacobson. “Congestion Avoidance and Control”.SIGCOMM Symposium no Communication Architecture and protocols.IJCN, Vol (2) 1988.

[2]K.Fall, S.Floyd “Simulation Based Comparison of Tahoe, Reno and SACK TCP” International Journal of Advances in Engineering & Technology, Aug 2011.

[3] 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).

[4] M. Jehan, Dr Radhamani and T KalaKumari “VEGAS: Better Performance than other TCP congestion control algorithms on MANETs” international journal of computer Networks (IJCN), Volume (3): Issue (2):2011.

[5] Vishnu Kumar Sharma and Dr Sarita Singh Bhadauria “Performance Analysis on Mobile Agent Based Congestion Control using AODV routing protocol techniques with Hop by Hop algorithms for MANET” International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.3, No.2, April 2012.

[6] Poonam Tomar and Shweta Yadav “Enhanced Reliable TCP for congestion control with corruption control in MANETs” *Journal of Global Research in Computer Science* Volume 3, No. 4, April 2012.

[7] Prof. S.A. Jain, Mr. Abhishek Bande, Mr. Gaurav Deshmukh, Mr. Yogesh Rade, Mr. Mahesh Sandhanshiv “An Improvement In Congestion Control Using Multipath Routing in Manet” International Journal of Engineering Research and Applications (IJERA) Vol. 2, Issue 3, May-Jun 2012,

[8] Mandakini Tayade and Sanjeev Sharma “Review of different TCP variants in ah hoc networks” IJEST ISSN: 0975-5462.