Comparative Analysis of Different TCP Variants in Mobile Ad-Hoc Network

Hrituparna Paul
Research Scholar
Dept. of Comp. Sc.
Assam University

Anish Kumar Saha
Asstt. Professor
Dept. of Comp Sc & Engg.
N.I.T Agartala

Partha Pratim Deb
M.Tech CSE
Netaji Subhash Engg College
West Bengal, India

Partha Sarathi Bhattacharjee
Research Scholar
Dept. of Comp. Sc.
Assam University, Silchar

ABSTRACT
Mobile Ad-Hoc Network is an autonomous group of mobile users that communicate using wireless links with no support from any pre-existing infrastructure network and used as a highly reliable end-to-end protocol for transporting applications. But in wireless networks suffers from significant throughput degradation and delays. It uses Congestion Control and Avoidance algorithms which degrades end-to-end performance in wireless system. In this paper we have analyzed the performance of tcp algorithms with AODV, DSR and TORA for throughput. The effect of throughput on the TCP variants New Reno, Reno and Tahoe with different node scenarios was studied.

Keyword
TCP Variants, Throughput.

1. INTRODUCTION
Mobile Ad Hoc Networks (MANETs) are wireless mobile nodes or an autonomous group of mobile users that cooperatively form a network without infrastructure. This network allows devices to create a network on demand without prior coordination or configuration and nodes within a MANET are involved in routing and forwarding information between neighbors.

There is a direct communication among neighboring devices in MANETs but communication between non-neighboring devices requires a routing algorithm. A lot of work has been done on routing protocols since they are critical to the functioning of ad-hoc networks [1], [2], [3] Within the two categories of routing protocols described in literature: Proactive and Reactive, it is more suited for highly mobile ad hoc networks due to its ability to cope with rapidly changing network topologies. Because there is no coordination or configuration prior to setup of a MANET, there are several challenges and these challenges include routing packets in an environment where the topology is changing frequently and the task of locating a node and maintain a path to it becomes increasingly in the face of node mobility.

Transport Control Protocol /Internet Protocol (TCP/IP) is a connection oriented protocol of the transport layer. It provides features like flow control, reliability and congestion control. It has been very effective in data transmission delivery and have also developed variants to possess the possibility to increase performance and multiple packet loss recovery.

Today, the TCP is extensively tuned to provide high-quality performance in the conventional wired network. In fact, the TCP is responsible for providing reliable data transport in the Internet. However, it cannot offer reliable service while using e-mail, internet search and file transmission in a MANET. This protocol is a standard networking protocol on the internet and is the most widely used transport protocol for data services like file transfer, e-mail and WWW browser. It is primarily designed for wire-line networks, faces performance degradation when applied to the ad hoc scenario. In addition, various routing protocols behave differently over the variants of TCP. It is essential to understand the performance of different MANET routing protocols under TCP variants. In this paper, we have done a performance analysis of MANET Routing Protocols over different TCP Variants.

The paper is organised as follows. Section 1 provides Introduction. Section 2 describes the Standard TCP congestion control algorithms. Section 3 describes the various TCP variants. Section 4 presents the simulation setup of our work. Finally Section 5 gives the future scope of our work concludes the paper.

2. TCP CONGESTION CONTROL ALGORITHMS
The four algorithms, Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery [4][5] are described below:

2.1. Slow Start [9]
Slow Start, a requirement for TCP software implementations is a mechanism used by the sender to control the transmission rate, and also known as sender based flow control. This is accomplished through the return rate of acknowledgements from the receiver. In other words, the rate of acknowledgements returned by the receiver determines the rate at which the sender can transmit data.

When a TCP connection first begins, the Slow Start algorithm initializes a congestion window to one segment, which is the maximum segment size (MSS) initialized by the receiver during the connection establishment phase and when acknowledgements are returned by the receiver, the congestion window increases by one segment for each acknowledgement returned. In this way, the sender can transmit the minimum of the congestion window and the advertised window of the receiver, which is simply called the transmission window.

At some point the congestion window may become too large for the network or network conditions may change such that packets may be dropped. Packets lost will trigger a timeout at the sender and when this happens, the sender goes into congestion avoidance mode.

2.2. Congestion Avoidance [9]
During data transfer phase of a TCP connection the Slow Start algorithm is used. There may be a point during Slow Start that the network is forced to drop one or more packets due to

overload or congestion. When this happens, Congestion Avoidance is used to slow the transmission rate and Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow down and stay slow.

In the Congestion Avoidance [9] algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked.

As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

2.3. Fast Retransmit and Fast Recovery [9]

Whenever a packet segment is transmitted, TCP sets a timer each time and thus it ensures the reliability. TCP retransmits the packet, if it does not obtain any acknowledgement within the fixed time-out interval and the reason for not getting any ACKs within a specific duration is due to either the packet loss or the network congestion. Therefore the TCP sender implements the fast retransmit algorithm for identifying and also repairing the loss. This fast retransmit phase is applied mainly based on the incoming duplicate ACKs and as TCP is not able to understand whether a packet loss or an out-of-order segment causes the generation of the duplicate ACK, it waits for more duplicate ACKs to be received. Because in case of out-of order segment, one or two duplicate ACKs will be received before the reordered segment is processed and on the other hand, if there are at least three duplicate ACKs in a row, it can be assumed that a segment has been lost. In that case, the sender will retransmit the missing data packets for a retransmission timer to expire without waiting.

After the missing segment is retransmitted, the TCP will initiate the fast recovery mechanism until a non-duplicate ACK arrives. The fast recovery algorithm is an improvement of congestion control mechanism that ensures higher throughput even during moderate congestion and the receiver yields the duplicate ACK only when another segment is reached to it. Therefore this segment is kept in the receiver's buffer and does not consume any network resources. This means that data flow is still running in the network, and TCP is reluctant to reduce the flow immediately by moving into the slow start phase. Thus, in that case in fast recovery algorithm, congestion avoidance phase is again invoked instead of slow start phase as soon as the fast retransmission mechanism is completed

3. TCP VARIANTS

3.1. TCP Tahoe

Tahoe [5] refers to the TCP congestion control algorithm which was suggested by Van Jacobson in his paper. TCP is based on a principle of conservation of packets, i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out as well. It implements this principle by using the acknowledgements to clock outgoing packets because an acknowledgement means that a packet was taken off the wire by the receiver. It also maintains a congestion window CWD to reflect the network capacity**.** It suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called slow-start. Reason for this procedure is that an initial burst might overwhelm the network and the connection might never get started.

The congestion window size is multiplicatively increased that is it becomes double for each transmission until it encounters congestion. 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. Sender is notified that congestion has occurred based on the packet loss.

3.2. TCP Reno

This RENO retains the basic principle of Tahoe, such as slow starts and the coarse grain retransmit timer [8]. However it adds some intelligence over it so that lost packets are detected earlier and the pipeline is not emptied every time a packet is lost. Reno requires that we receive immediate acknowledgement whenever a segment is received. The logic behind this is that whenever we receive a duplicate acknowledgment, then his duplicate acknowledgment could have been received if the next segment in sequence expected, has been delayed in the network and the segments reached there out of order or else that the packet is lost. If we receive a number of duplicate acknowledgements then that means that sufficient time have passed and even if the segment had taken a longer path, it should have gotten to the receiver by now. There is a very high probability that it was lost. So Reno suggests Fast Re- Transmit. Whenever we receive 3 duplicate ACK's we take it as a sign that the segment was lost, so we re-transmit the segment without waiting for timeout.

Thus we manage to re-transmit the segment with the pipe almost full. Another modification that RENO makes is in that after a packet loss, it does not reduce the congestion window to 1. Since this empties the pipe. It enters into an algorithm which we call Fast-Recovery.

3.3. TCP New Reno

New RENO is a slight modification over TCP-RENO. It is able to detect multiple packet losses and thus is much more efficient that RENO in the event of multiple packet losses. Like RENO, New-RENO [7] also enters into fast retransmit when it receives multiple duplicate packets, however it differs from RENO in that it doesn't exit fast recovery until all the data which was out standing at the time it entered fast recovery is acknowledged. The fast recovery phase proceeds as in Reno, however when a fresh ACK is received then there are two cases

- If it ACK's all the segments which were outstanding when we entered fast recovery then it exits fast recovery and sets CWD to threshold value and continues congestion avoidance like Tahoe.

- If the ACK is a partial ACK then it deduces that the next segment in line was lost and it re-transmits that segment and sets the number of duplicate ACKS received to zero. It exits Fast recovery when all the data in the window is acknowledged.

4. SIMULATION

We have evaluated the performance of different variants of TCP using OPNET simulator. Here AODV, DSR and TORA are simulated with different TCP algorithms with different scenarios (three, five node scenarios).

In three node scenarios the throughput performance between AODV and DSR at approximately 25 seconds, AODV has better throughput performance over DSR and comparing AODV with TORA at the approximation of 95 sec, AODV provides better throughput performance, in all the three graphs.

Total time consumed by each protocol for sending the data through New Reno, Reno and Tahoe is relatively smaller for AODV followed by DSR and than TORA. That is TORA has worst throughput performance in each of the three cases for MANET in 3 nodes. The cause behind the TORA less performance is considered as the reality of TORA working for route recreation, maintenance and erasure, if dropping of the route occurs, that requires more time and have bad impact in the data performance.

In five nodes scenario five nodes working as clients to establish connection with a fixed node working as source, and to transfer a file of the same size over each connection.

Comparing all the three graphs with each other we observe that as the number of the nodes are increased approximately 75% (3 to 5), throughput has been decreased for every TCP variant and each desired protocols.

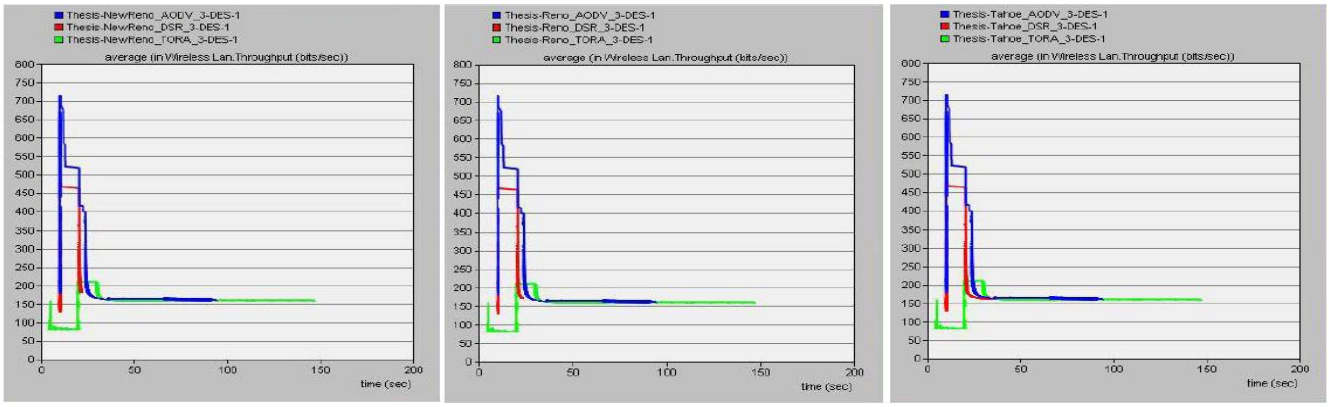

Fig:1 Throughput comparison in three nodes scenario

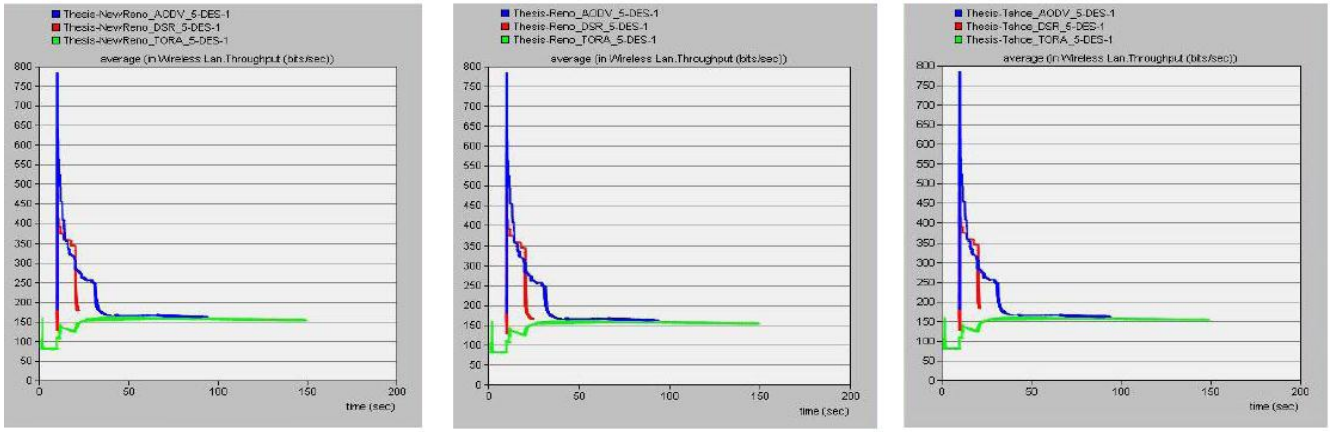

Fig:2 Throughput comparison in five nodes scenario

5. CONCLUSION

It has been concluded that throughput performance of DSR and TORA are minutely affected with increase in the number of nodes and due to mobility. In general, MANET could have dynamic number of nodes connectivity in mobility, so it's important to realize that when the number of nodes is higher, DSR and TORA would be avoided. AODV has better throughput performance shown in all three figures as compared to DSR and TORA and is the best solution for MANET. However simulation results for AODV with respect to New Reno, Reno and Tahoe depict that throughput is the same in all the cases, so our proposed solution in this case will be New Reno as it offers multiple packet loss recovery.

For future work we propose to study the performance of these TCP variants for different routing protocols such as OLSR, DSDV. We would also like to expand the range of analysis by considering other new TCP's like HSTCP, TCP Westwood, TCP Veno, TCP Vegas etc.

6. REFERENCES

[1] C.E. Perkins and E.M. Royer, "Ad-hoc On-Demand distance vector routing", Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, pp. 90-100, February 1999.

[2] D. Johnson, D. Maltz and Y. Hu., "The dynamic source routing protocol for mobile ad hoc networks", IETF MANET Working Group, Internet Draft, 2003.

[3] M.K.J. Kumar and R.S. Rajesh, "Performance analysis of MANET routing protocols in different mobility models", IJCSNS International Journal of Computer Science and Network Security, vol. 9 No.2, pp 22-29, Feb 2009

[4] K.Kathiravan, Dr. S. Thamarai Selvi, A.Selvam "Tcp Performance Analysis For Mobile Ad Hoc Network Using Ondemand Routing Protocols.

[5] JACOBSON, V. Congestion avoidance and control. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988), ACM.

[6] Laxmi Subedi, Mohamadreza Najiminaini, and Ljiljana Trajkovi Performance Evaluation of TCP Tahoe, Reno, Reno with SACK, and NewReno using OPNET Modeler.

[7] S.Floyd, T.Henderson "The New- Reno Modification to TCP's Fast Recovery Algorithm" RFC 2582, Apr 1999.

[8] O. Ait-Hellal, E.Altman "Analysis of TCP Reno and TCP Vegas".

[9] Suhas Waghmare et. al "Comparative Analysis of different TCP variants in a wireless environment", 978-1-4244-8679-3/11 ©2011 IEEE