International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.1, No.5, September 2012 DOI:10.15693/ijaist/2012.v1i5.41-45

A New Comparison of Mobility Models in MANET using TCP Protocols

S.Allwin Devaraj
Assistant Professor
Department of ECE
Francis Xavier Engineering College, Tirunelveli

Abstract – **In Mobile Ad-hoc Network (MANET), mobile nodes act as routers themselves, keeping route information to reach other mobile nodes, and helps to forward data packets from one mobile to another. The performance of the different mobility models such as Random Walk (RW), Random Waypoint (RWP), Manhattan Mobility (MM), Reference Point Group Mobility (RPGM) are analyzed in TCP (Transmission Control Protocol) Environment using NS2 software under Linux platform. Depending on the type of TCP implementation the behavior was different, due to the activation/missing of the following congestion control algorithms:"Slow-Start", "Congestion Avoidance", "Fast Recovery" and "Fast Retransmit". The performance of TCP flows in an ad-hoc network is analyzed using four different mobility models with AODV as the routing protocol and TCP as the transport protocol.**

Keywords – **MANET, Mobility Models, Slow Start, Congestion Avoidance, Additive Increase Multiplicative Decrease (AIMD), TCP New Reno**

I. INTRODUCTION

The Transmission Control Protocol is originally first designed for low speed and point to point wired networks and it is a connection oriented protocol. New complexities and issues were introduced to TCP, as soon as the internet is evolved, and gigabit Ethernet and wireless mobile connectivity became the latest trend[]. So TCP shows very poor performance in MANETs etc...

A Mobile Ad Hoc Network (MANET) is considered an autonomous collection of wireless mobile nodes that are capable of communicating with each other without the use of a network infrastructure or any centralized administration [1], [4]. Due to the host mobility, the network topology may change rapidly and unpredictably over time. The network is decentralized no administrator is required to manage network. It is self organizing and enables communication in situations where there is no time to set up the necessary infrastructure or situations where the need for a communication network is temporally required. MANETs have a wide range of applications form military to search and rescue operations during disaster [2]. The interest of the scientific and industrial community in the area of telecommunications has recently shifted to more challenging scenarios in which a group of mobile units equipped with radio transceivers communicate without any fixed infrastructure [1].

A. Transmission Control Protocol

Transmission Control Protocol is the Internet's most widely used transport control protocol. TCP's strength lies in the adaptive nature of its congestion avoidance and control algorithm and its retransmission mechanism, first proposed by V. Jacobson [6], [11] as a part of TCP Tahoe. It was further refined in Reno and New Reno versions of TCP. The major control mechanisms of TCP are its congestion avoidance and congestion control mechanism. They are discussed in detail below:

1) AIMD Mechanism of TCP

TCP maintain a new state variable for each connection, called congestion window, which is used by the source to limit how much data it is allowed to have in transmit at a given time. The congestion window is congestion controls counterpart to flow control advertised window. TCP is modified such that the maximum number of bytes of unacknowledged data allowed is now the minimum of the congestion window and the advertised window. TCP's effective window is revised as,

Max window = MIN (congestion window, advertised window)
Effective window = max window- (last byte sent- last byte Acknowledged)

That is, max window replaces advertised window in the calculation of effective window. Thus, a TCP source is allowed to send no faster than the slowest component the network or the destination host can accommodate. The problem, of course, is how TCP comes to learn an appropriate value for congestion window. Unlike the advertised window, which is sent by the receiving side of the congestion, there is no one to send a suitable congestion window to the sending side of TCP. The answer is that the TCP source sets the congestion window based on the level of congestion perceives to exit in the network. This involves decreasing the congestion window when the level of congestion goes down and the mechanism is commonly called additive increase/multiplicative decrease. The source determines if the network is congested and that it should decrease the congestion window based on the observation that the packets are not delivered and a timeout results, is that a packet was dropped due to congestion. It is rare that a packet is dropped because of an error during transmission.

Therefore, TCP interprets timeouts as a sign of congestion and reduces the rate at which it is transmitting.

Specifically, each time a timeout occurs, the source sets congestion window for each timeout corresponds to the “multiplicative decrease” part of AIMD.

Although congestion window is defined in terms of bytes, it is easiest to understand multiplicative decrease if we think in terms of whole packets. For example, suppose the congestion window is currently set to 16 packets. If a loss is detected, congestion window is set to 8. Normally, a loss is detected when a timeout occurs. Additional losses cause congestion window to be reduced to 4, then 2, and finally to 1 packet. A congestion control strategy that only decrease the window size is obviously too conservative. We also need to be able to increase the congestion window to take advantage of newly available capacity in the network. This is the “additive increase” part of AIMD, and it works as follows. Every time the source successfully sends a congestion window worth of packets that is, each packet sent out during the last RTT has been Acknowledged- it adds the equivalent of one packet to congestion window. TCP does not wait for an entire window’s worth of ACKs to add one packet’s worth to the congestion window, but instead increments congestion window by a little for each ACK that arrives. The order of packet transmission is x+2x+3x+4x+… where x=1

2) *SLOW START mechanism of TCP*

TCP’s reaction to a missing acknowledgement is quit drastic, but necessary to get rid of congestion fast enough. The behavior of TCP shows after the detection of congestion is called slow start. The sender always calculates a congestion window for a receiver. Start size of the congestion window is one segment (TCP packet). Now the sender sends one packet and waits for acknowledgement. If this acknowledgement arrives, the sender increases the congestion window one by one, now sending two packets (congestion window= 2). After arrival of the two corresponding acknowledgements, the sender again adds 2 to the congestion window; one for each of the acknowledgements .Now the congestion window equals 4. This scheme doubles the congestion window every time the acknowledgements come back, which takes one round trip time (RTT). This is called the exponential growth of the congestion window in slow start mechanism. It is too dangerous to double the congestion window each time because the step might become too large. Therefore, the exponential growth stops at the congestion threshold .As soon as the congestion window reaches the congestion threshold, further increase of the transmission rate is only linear by adding 1 to the congestion window each time the acknowledgements come back. The order of packet transmission is $x^0+x^1+x^2+x^3+x^4$… where x=2,3,4….

B. *TCP Reno*

This Reno retains the basic principle of Tahoe, such as slow starts and the coarse grain re-transmit timer. 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 [12],[14].

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 has 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 suggest an algorithm called **‘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 a algorithm which we call **Fast-Re-Transmit.** The basic algorithm is presented as under:

1) Each time we receive 3 duplicate ACK’s we take that to mean that the segment was lost and we re-transmit the segment immediately and enter Fast- Recovery.

2) Set Ssthresh to half the current window size and also set CWD to the same value.

3) For each duplicate ACK receive increase CWD by one. If the increase CWD is greater than the amount of data in the pipe then transmit a new segment else wait. If there are ‘w’ segments in the window and one is lost, then we will receive (w-1) duplicate ACK’s. Since CWD is reduced to W/2, therefore half a window of data is acknowledged before we can send a new segment. Once we retransmit a segment, we would have to wait for atlease one RTT before we would receive a fresh acknowledgement. Whenever we receive a fresh ACK we reduce the CWND to SSthresh. If we had previously received (w-1) duplicate ACK’s then at this point we should have exactly w/2 segments in the pipe which is equal to what we set the CWND to be at the end of fast recovery. Thus we don’t empty the pipe, we just reduce the flow.

C. *TCP New Reno*

New RENO is a slight modification over TCP-RENO [6],[7],[9]. 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 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.

II. MOBILITY MODELS

A. Introduction

The results of some experiments are explained here and these are used to measure the proposed metrics assuming several typical mobility models: Random Walk (RW), Random WayPoint (RWP), Manhattan Mobility (MM), and Reference Point Group Mobility (RPGM).These typical mobility models and their experimental results are explained in below.

B. Random Walk

This is one of the simplest mobility models and is often used in simulation experiments for MANET. In this model, at every unit of experimental time, each mobile node randomly determines a movement direction from all directions, and randomly determines a movement speed from 0 to V m/s. It is known that in the long term, this model offers very low mobility similar to vibrating in the same position, because mobile nodes randomly change movement direction.

C. Random WayPoint

This is one of the most popular mobility models for MANET researchers. In this model, each node remains stationary for a pause time S sec. Then, it selects a random destination in the entire area and moves to the destination at a speed determined randomly between 0 and V m/s. After reaching the destination, it again pauses, and then, repeats this behaviour. It is known that in this model, mobile nodes tend to gather at the centres of the area, and the movement speed tends to converge to zero (very low).

D. Manhattan Mobility

This model emulates the node movement on streets where nodes only travel on the pathways in the map. On each street, the mobile nodes move along the lanes in both directions. At each intersection, the mobile nodes choose their directions and speed (0 to V m/s) randomly. The MM model showed several interesting features due to its restricted mobility and this model forms many small partitions but rarely forms large ones when the node density is not very high compare to the RW model.

E. Reference Point Group Mobility Model

This model is used to model group mobility. Each group has a logical “center” called a reference point and group members (nodes). Each reference point moves according to the RWP model with V m/s (maximum speed) and S sec (pause time). In each group, nodes are uniformly distributed within a certain radius R from the reference point. To achieve this, each node moves according to the RW model with V m/s (maximum speed) within that range. Specifically, a node’s movement vector is composed by adding the movement vector based on the RW model for the node to that based on the RWP model for the reference point.

III. PERFORMANCE ANALYSIS AND RESULTS

A. *Simulation Model and Parameters*

The simulation platform is ns-2 [15] which is a discrete event simulator. In our simulation, mobile 100 mobile nodes move in a 2500 meter x 2500 meter rectangular region for 100 seconds simulation time. We assume each node moves independently with the same average speed. All nodes have the same transmission range of 250 meters.

B. *Performance Metrics*

We evaluate mainly the performance according to the following metrics.

Packet Delivery Ratio:

Packet Delivery Ratio (PDR) is calculated by dividing the number of packets received by the destination through the number of packets originated by the source.

Delay:

The term the average delay is a data packet experiences to cross from source to destination. This delay includes all possible delays caused by buffering during route discovery delay, queuing at the interface queues and retransmission delays at the MAC, propagation and transfer times.

C. Simulation Results

1. Random Walk Mobility Model

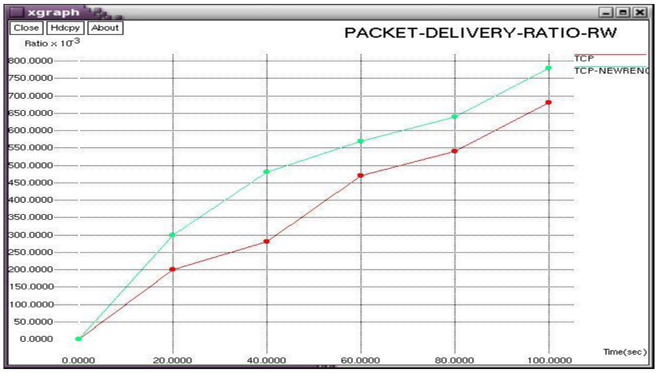

Figure 1 Packet Delivery Ratio

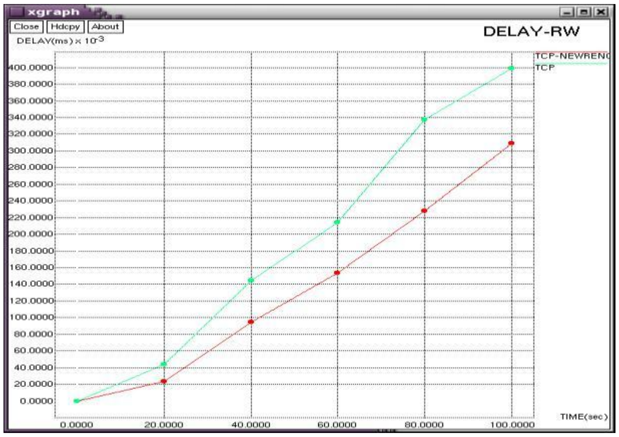

Figure 2 Delay

Figure 1 shows the results of packet delivery ratio for RW mobility model. From the simulation results, it is seen that TCP New Reno achieves more delivery ratio than TCP.

Figure 2 shows the results of delay for RW mobility model. From the simulation results, it is seen that TCP New Reno achieves less delay than TCP.

2. Random Way Point Mobility Model

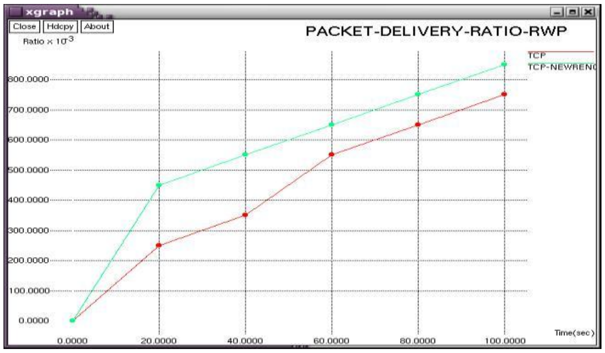

Figure 3 Packet Delivery Ratio (RWP)

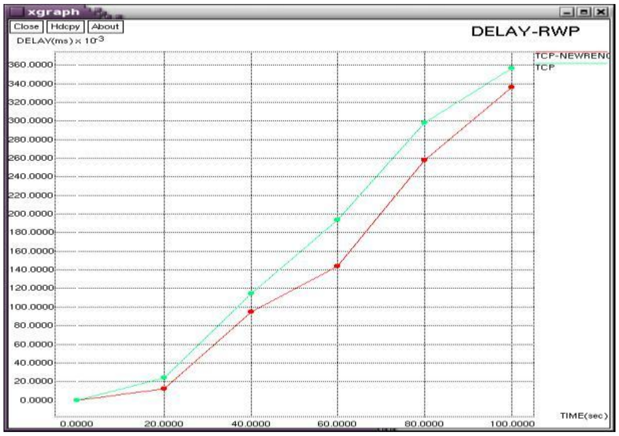

Figure 4 Delay (RWP)

Figure 3 shows the results of packet delivery ratio for RWP mobility model. From the simulation results, it is seen that TCP New Reno achieves more delivery ratio than TCP.

Figure 4 shows the results of delay for RWP mobility model. From the simulation results, it is seen that TCP New Reno achieves less delay than TCP in RWP mobility model.

3. Manhattan Mobility Model

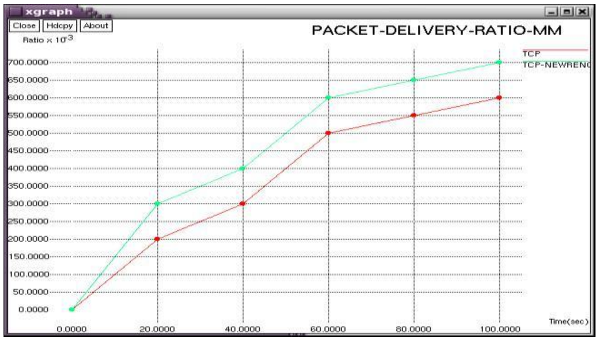

Figure 5 Packet Delivery Ratio (MM)

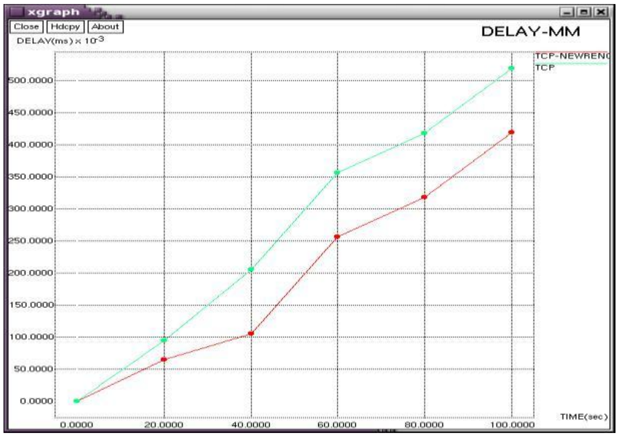

Figure 6 Delay (MM)

Figure 5 shows the results of packet delivery ratio for MM mobility model. From the simulation results, it is seen that TCP New Reno achieves more delivery ratio than TCP.

Figure 6 shows the results of delay for MM mobility model. From the simulation results, it is seen that TCP New Reno achieves less delay than TCP in TCP environment.

4. *Reference Group Point Mobility Model*

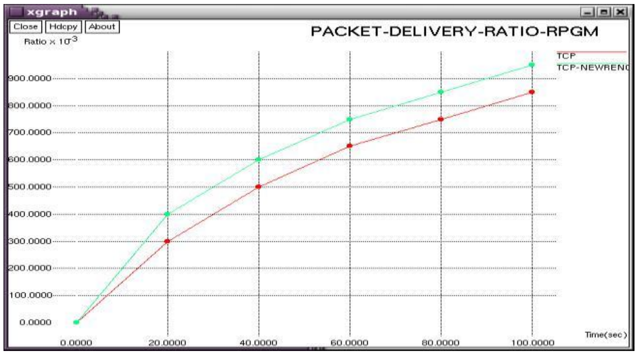

Figure 7 Packet Delivery Ratio (RPGM)

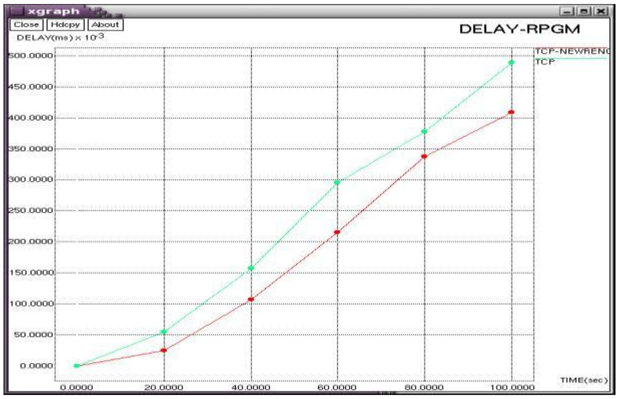

Figure 8 Delay (RPGM)

Figure 7 shows the results of packet delivery ratio for RPGM mobility model. From the simulation results, it is seen that TCP New Reno achieves more delivery ratio than TCP.

Figure 8 shows the results of delay for RPGM mobility model. From the simulation results, it is seen that TCP New Reno achieves less delay than TCP.

IV. CONCLUSION

It is seen from the simulation analysis TCP shows very poor performance over the networks. The reason for the performance degradation is that the congestion control mechanism in TCP cannot distinguish between the packet loss caused by wireless link error and that caused by network congestion, thus, reacting to the loss by reducing its congestion window (cwnd). The results of experiments are reported that measured the proposed metrics assuming four typical mobility models: RW, RWP, MM and RPGM. In this contribution, the performance of TCP and New Reno has been analyzed. Simulation analysis shows that the TCP New Reno was the dominant protocol on packet delivery ratio and delay with TCP.

REFERENCES

[1] Hahner. J, Dudkowski. D, Marron. P and Rothermel. K, “Quantifying Network Partitioning in Mobile Ad Hoc Networks”, *Proc. Int’l Conf. Mobile Data Management*, pp. 174-181, 2007.

[2] Jardosh. K, Belding Royer. E, Almeroth. K and Suri. S , “Towards Realistic Mobility Models for Mobile Ad Hoc Networks*”, Proc. Mobicom*, pp. 217-219, 2003.

[3] Kwak. B, Song. N and Miller. L, “A Mobility Measure for Mobile Ad-Hoc Networks”, *IEEE Comm. Letters*, vol. 7, No. 8, pp. 379-381, 2003.

[4] Hara, T, “Quantifying Impact of Mobility on Data Availability in Mobile Ad Hoc Networks”, *IEEE Transaction on mobile computing*, vol.9, No.2, pp 241-258, 2010.

[5] Ng. J and Zhang. Y, “A Mobility Model with Group Partitioning for Wireless Ad Hoc Networks”, *Proc. Int’l Conf.Information Technology and Applications*,vol.2, No.5, pp. 289-294, 2005.

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

[7] S.U Lar, X.Liao and S.Guo,“Modeling TCP NewReno Slow Start and Congestion- Avoidance using Simulation Approach” *IJCSNS International Journal of Computer Science and Network Security*, VOL.11 No.1, January 2011.

[8] Yuvaraju B N, N Chiplunkar, “Scenario Based Performance Analysis of Variants of TCP Using NS2 – Simulator” *International Journal of Computer Applications* (0975 – 8887) Volume 4, No.9, August 2010.

[9] M Tekala and R Szabo, “Modeling Scalable TCP friendliness to NewReno TCP” IJCSNS *International Journal of Computer Science and Network Security*, VOL.7 No.3, March 2007.

[10] http://www.isi.edu/nsnam/ns NS–2, Network Simulator, March 2005.

[11] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control”, *RFC 2581*, IETF, April 1999.

[12] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Warland, “Analysis and comparison of TCP Reno and Vegas,” in Proceedings of IEEE INFOCOM’99, March 1999.

[13] Jitendra Padhye, VictorFiroiu, “Modeling TCP Reno Performance A simple model & its Empirical Validation” *IEEE/ACM Transactions* Volume (8)-02 April 2000.

[14] Habibullah Jamal and Kiran Sultan, “Performance Analysis of TCP Congestion Control Algorithms”, *International Journal of Computers and Communications*, Volume (02) - 01, 2008.