International Journal of Engineering Research and Development
***e**-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com*
Volume 3, Issue 8 (September 2012), PP. 27-35

Enhanced Recovery scheme for TCP NewReno in MANET

Aparna Shrivastva[1], Asish Khare[2], Sunita Tiwari[3]
[1]*Radharaman Institute of Science & Tecnology, Bhopal, India.*
[2]*Radharaman Institute Of Science & Technology, India.*
[3]*Krishna Engineering College, Ghaziabad, India.*

Abstract—A mobile ad hoc network is a collection of mobile nodes without any infrastructure. Wireless links have issues like low bandwidth, node mobility multi-hop, channel competition and interference from other channels. Since TCP/IP is most popular and most widely used networking protocol on the Internet, therefore its use over ad hoc network is obvious. In multi hop wireless network the principle problem of TCP lies in its congestion control. TCP reacts to all packet losses as if they were cause by congestion i.e. a long time out and dropping congestion window. In wireless links there can be losses due to link failure so TCP can reveal poor performance. Mobile ad hoc networks need routing algorithm for communication among non-neighbouring nodes. Various routing algorithms for MANETs have been proposed in literature. We have used NS-2 simulation to analyze the performance of TCP NewReno over a set of routing protocols including DSR, DSDV and AODV. We aim to prove that the choice of ad hoc routing protocol to be implemented within the network affects TCP behaviour within MANET. We are also proposing a change in the recovery strategy of TCP NewReno for faster recovery and avoiding frequent coarse timeout and compared its performance with TCP NewReno over the abovementioned set of routing protocols in MANET.

Keywords— NewReno, Modified TCP NewReno, MANET, ad hoc routing, NS-2 Simulator.

I. INTRODUCTION

The wireless networks popularity is increasing very rapidly. Wireless ad hoc networks are very easy to implement and cost effective networks as they do not require any pre-existing infrastructure and base stations. Ad hoc networks are dynamic in nature and nodes can move freely and are self organize. Ad hoc network are potential technology for future.

In wireless network route changes are frequent due to unrestricted topology changes. Due to lack of fixed infrastructure various issues like interferences, unreliable wireless medium, asymmetric propagation properties of wireless channel, hidden and exposed terminal phenomena, transmission rate limitation and blindly invoking congestion control of transport layer etc are inherent in MANET. Among all above mentioned problems we have analysed the impact of transport layer over various routing protocols.

TCP is a standard transport layer protocol for reliable end to end delivery of data packet in traditional wired networks. TCP is independent of lower layer protocols. TCP under performs in wireless ad hoc networks due to rapid topology changes, limited battery power and node mobility. In order to be used in wireless ad hoc networks, TCP should face various challenges, such as packet losses due to congestion, high bit error rate (BER), mobility, delay and so forth. Various TCP versions such as Tahoe, Reno, NewReno, Vegas, and Westwood are enhancements of TCP and they perform differently depending on the routing protocols.

Routing a packet between a pair of nodes in wireless ad hoc network is a challenging task as nodes moves randomly. A path that is considered optimal at some point in time may not be good enough after some time. The conventional routing protocols as DSDV (Destination Sequenced Distance Vector) are proactive and maintain routes in all the nodes in the network. Proactive protocols react to any change in the topology even if no traffic is affected by the change. They also require periodic control messages for route maintenance in every node in the network. If mobility is high in networks, most scarce resources like bandwidth and battery power will be consumed in this process. On the other hand reactive protocols such as DSR(Destination Source Routing) and AODV (Ad hoc On Demand Distance Vector) determines route only when the explicitly require to route the packets. This will avoid unnecessary updating of every possible route in the network. But reactive protocols causes another problem called broadcast storm problem.

In this work we have performed a simulation study of performance of TCP NewReno over a set of ad hoc routing protocols DSR, DSDV and AODV. We will prove that the choice of ad hoc routing protocol to be implemented within the network affects TCP behaviour within MANET. Also, we are proposing a change in the recovery strategy of TCP NewReno for faster recovery and avoiding frequent coarse timeout. We have compared the performance of Modified TCP NewReno with TCP NewReno in light of abovementioned set of routing protocols.

Rest of the paper is organized as follows. Overview of routing protocol is provided in section 2 and overview of TCP variants is presented in section 3. Section 4 provides the the performance analysis of TCP NewReno over different routing protocols and in section 5 proposed modification in TCP New Reno is presented. In section 6 we discuss about simulation environment and results. Section 6 present conclusions.

II. OVERVIEW OF AD HOC ROUTING PROTOCOLS

In this section we are presenting a brief overview of the three routing protocols DSDV (Proactive), DSR (Reactive) and AODV (Reactive). In wireless ad hoc networks there must be some way of finding a route between two nodes. This is done with an ad hoc routing protocol. Often the routing protocol operates below the network layer, but still has knowledge about it. In literature various routing protocols for MANET exists each with their own strength and weaknesses.

A. Destination Sequenced Distance Vector Routing (DSDV)

DSDV is a table driven routing protocol. DSDV is known as proactive routing protocol. DSDV requires each node to maintain a routing table (Destination-address, Metric, and Sequence-number) for the next hop to reach a destination node and number of hops to reach destination. It periodically broadcast updates to the network and if a node does not receive a periodic update from its neighbor for a while link is assumed to be broken. Each route in DSDV is tagged by a sequence number to avoid formation of routing loops. It chooses the route for forwarding which have highest sequence number and when two routes have same sequence number than one with lower metric is chosen.

B. Destination Sequence Routing (DSR)

DSR is also distance vector routing and uses sequence number to avoid route looping. It belongs to the class of reactive protocol and allows nodes to dynamically discover route. It does not maintain routes to all possible destinations but establish them dynamically as and when the need arise. The overhead of route table maintenance is therefore low as each node does not need to contain up to date information for a complete path to a destination because the complete route a packet must follow to reach its destination is included in its header by the source [1]. Source routing ensures the loop freedom. To discover the route to destination from source network is flooded with route request packets (RREQ). A RREQ packet is rebroadcast by all intermediate nodes. The destination node replies to the earliest request to the source node. Route discovery mechanism of DSR is shown in Fig. 1 a) and b).

C. Ad Hoc On-Demand Distance Vector (AODV)

AODV is also a reactive protocol based on DSDV and DSR. In AODV each node maintains a routing protocol table one entry per destination like DSDV. Each entry records the next hop to the destination and its hop count. To indicate that it is an updated route AODV maintains sequence-number generated by destination node. Route discovery of AODV is shown in following figure 2 a) and b). Unlike DSR AODV does not maintain or record the nodes it has passed through, it only maintains the hop count.

The main difference between AODV and DSR is that in DSR each packet sent from source contains the full destination route whereas AODV packets only contain destination information. Hence AODV requires less memory.

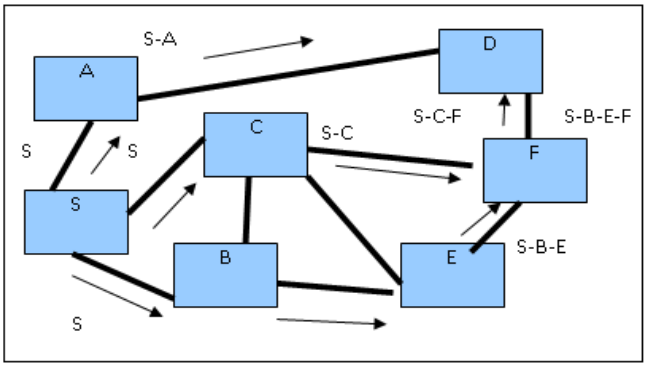


Fig. 1 a): Sending procedure of a request packet in DSR

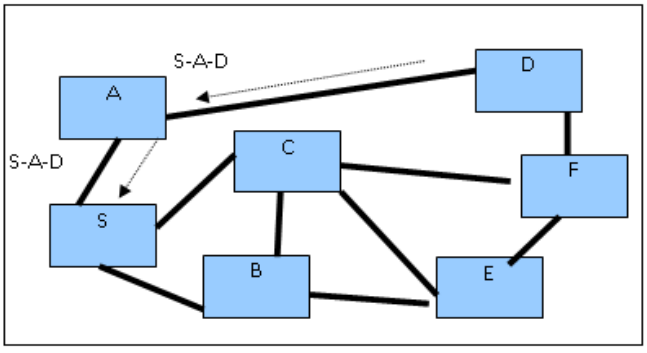


Fig. 1 b): Replying procedure of route request in DSR.

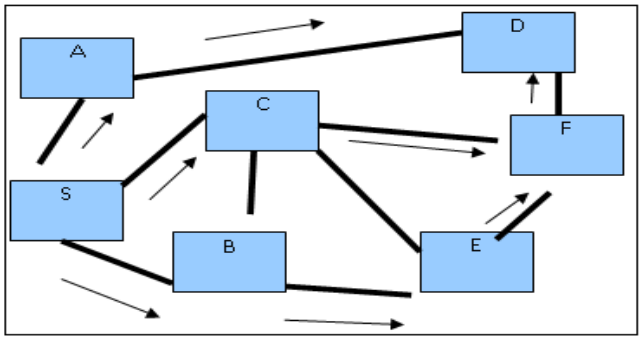


Fig. 2 a): Sending procedure of a request packet in AODV

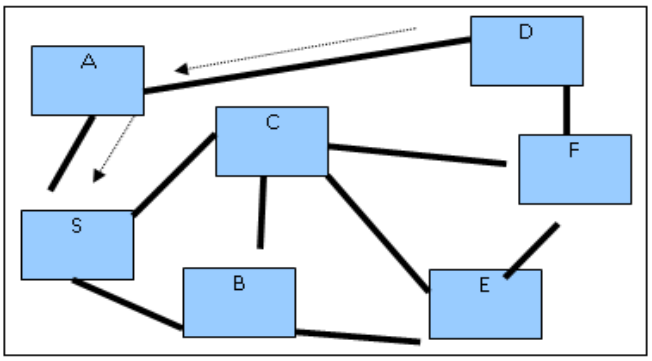


Fig. 2 b): Replying procedure of route request in AODV.

III. OVERVIEW OF TCP AND ITS VARIANTS

The TCP protocol provides reliability, flow control, congestion avoidance, fairness, and in-order delivery. TCP performs degrades in MANET as it was mainly developed for wired networks. One of the reasons for this degradation is that TCP cannot differentiate between losses due to congestion and losses due to link failures. TCP always interprets that packet loss is due to congestion in network and it reduces the transmission rate. Reduction in transmission rate hampers the link utilization and network performance in form of poor throughput. However main reason of packet loss in wireless network is not always congestion as link failures are also common. We will first discuss some TCP concepts and variants of TCP.

The strength of TCP lies in it's the adaptive nature of its congestion avoidance and control algorithm and its retransmission algorithm, first proposed as TCP Tahoe [1,3,4]. It was then refined in Reno and NewReno versions. Congestion avoidance algorithm of TCP Vegas was fundamentally different from that of TCP Tahoe. Congestion control and congestion avoidance concepts of TCP are discussed below.

Slow Start: when TCP starts transmission it starts with a slow rate as it needs to examine the bandwidth available. If TCP starts transmission at fast speed initially then the throughput of TCP connection will drastically affected because the intermediate nodes have to queue or drop the packets from buffer. There is a parameter cwnd in TCP which denotes the congestion window and its initial value is set as less than or equal to double of the Maximum Segment Size (MSS) but not greater than this. The size of cwnd is increased by one segment every time it receives an acknowledgement. Thus when first acknowledgment (ACK) arrives at the sender the size of cwnd becomes two and two segments are sent. When acknowledgment of these two arrives size becomes four and four segments are sent. Therefore increase in size of cwnd is exponential. Slow start continues to increase exponentially until its value less than or equal to slow start threshold denoted by ssthresh.

Congestion Avoidance: If packets loss starts then to avoid the loss of packet TCP uses congestion avoidance. Congestion avoidance is performed when the value of cwnd becomes greater than ssthresh. In congestion avoidance phase the value of cwnd is increased by 1 during every round trip Time (RTT). Congestion avoidance policy is used until the congestion is detected. TCP detect congestion by two methods.

Detection of congestion by timeouts: In this case value of ssthresh is updated by max (Flight Size/2, 2*MSS), where Flight Size is the amount of outstanding data in network and then cwnd value is set to 1 segment. When the packets are retransmitted TCP again performs slow start and after this same approach is followed for cwnd increase. After this again it goes into congestion avoidance phase.

Detection of congestion due to duplicate acknowledgments: if three duplicate acknowledgments (4 identical acknowledgements) arrive at sender then TCP assumes packet loss and it runs 'fast retransmit' and 'fast recovery' algorithm. TCP sender follows following steps on receiving three duplicate acknowledgements-

- It sets value of ssthresh =max(Flight Size/2, 2*MSS)

- Retransmit the lost packet and set cwnd= ssthresh + 3*MSS (fast retransmit).
- For every additional duplicate ACK received by sender it increments cwnd by one MSS.
- If sender window and new value of cwnd allows then transmits the new segment.
- If acknowledgment if new segment arrives it reset cwnd to ssthresh for fast recovery.

A. TCP Variants

Here we present a brief characterization of TCP Tahoe, TCP Reno, and TCP NewReno. TCP Tahoe [5] introduced congestion avoidance, where dropped packets are used as an indication of congestion, and slow start, where the initial window size keeps doubling until congestion is detected.

TCP Reno comes with an improvement over TCP Tahoe in which after a packet loss is indicated it does not reduce the size of congestion window to one it rather enters Fast Retransmit phase. Reno introduces mechanism of fast recovery after fast retransmit. During fast recovery it uses additive increase/multiplicative decrease at all time. Fast recovery helps in saving time by not waiting for timeout in order for re-transmission to begin. This avoids the need to go to slow start again after fast re-transmit. Therefore Reno shows performance improvement in case one packet is lost within a window of data except the case of multiple losses from the same window.

TCP-NewReno is an improvement of Reno [6], that is, advanced the fast transmit, where three duplicate acknowledgments signal a retransmission without a timeout with fast recovery. The fast recovery means that once a certain threshold of ACKs is received, the window size is decreased by half, rather than starting over with slow start. Only during timeout does it go back into slow start. NewReno increases the adoption of the TCP selective acknowledgements (SACK) modification. It can respond with the interpretation of partial ACKs as indications of packet losses. Because the timeout timer is renewed when ACKs are received, NewReno is able to maintain high throughput TCP-NewReno is a good solution to multiple losses in a single window. If some of the packets transmitted before the fast retransmit lose, the ACK's for the fast-retransmitted packet will acknowledge some but not all of the packets transmitted before the fast retransmit. This ACK packet is called a partial acknowledgment packet. TCP-NewReno will retransmit the indicated packet without delay when receiving the partial acknowledgment packet.

In [2], the authors conclude that TCP- NewReno can work well when there are only multiple losses/corruptions in a single window and the retransmission are usually successful. But when retransmission losses occur due to a high BER in lower layer, the lost packets which are transmitted after the fast retransmit can-not be recovered quickly by TCP-NewReno, and they also cannot be fast-retransmitted because of no enough duplicated ACK. Due to reduced size of congestion window.

Therefore successive timeout is inevitable. That leads to under-utilization of network resources. In order-to explain that clearly, we give an example (see Fig. 4).

IV. TCP NEWRENO PERFORMANCE COMPARISON OVER VARIOUS AD HOC ROUTING PROTOCOLS

In this work, we intend to study the effect of ad hoc routing protocols on TCP performance within MANETs. We consider in our study different types of ad hoc routing protocols having different characteristics. As shown in figure 3 a-d, on increasing number of nodes performance of TCP-NewReno gets affected.

For evaluating performance of TCP-NewReno over different routing protocols we have used following performance metrics-

1. **Delay** is defined as the time a data packet received by an intermediate hop minus the time data packet is forwarded by previous hop.
2. **The end to end delay** is defined as the time a data packet is received by the destination minus the time the data packet is generated by the source.
3. **Throughput** is measured at the sender as number of Kbits sent per second.
4. **The packet delivery ratio** defined as the number of received data packets divided by the number of generated data packets
5. **Congestion window** is measured at the sender as number of packets allowed on flight in the network.

We have used ns-2 simulation to study the performance of tcp-newreno over three different routing protocols. The ns network simulator from U.C. Berkeley/LBNL is an object-oriented discrete event simulator targeted at networking research and available as public domain. Ns-2 is well suited for packets switched networks and wireless networks including ad hoc, local and satellite, and is used mostly for small scale simulations of queuing and routing algorithms, transport protocols, congestion control, and some multicast related work. Ns-2 provides good support for TCP, routing over wireless networks. It also comes with a network emulator called NAM. It has utility to introduce live traffic and it comes with a rich suite of algorithms and models. The simulation parameters are shown in table I.

Table I: Simulation Parameters:

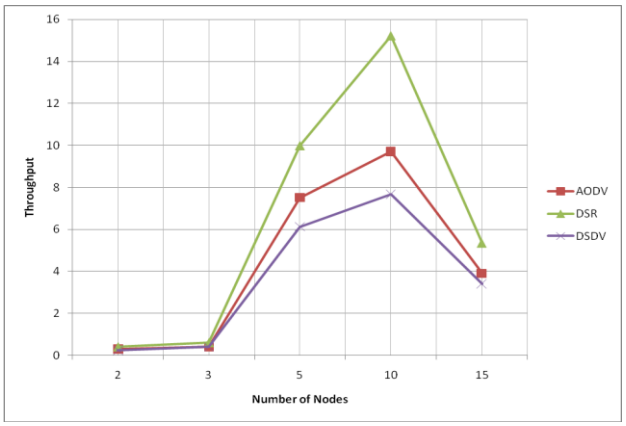
Parameter	**Value**
Simulation Time	300s
Topology Size	1200m*800m
Number of mobile nodes	2, 3, 5, 10, 15

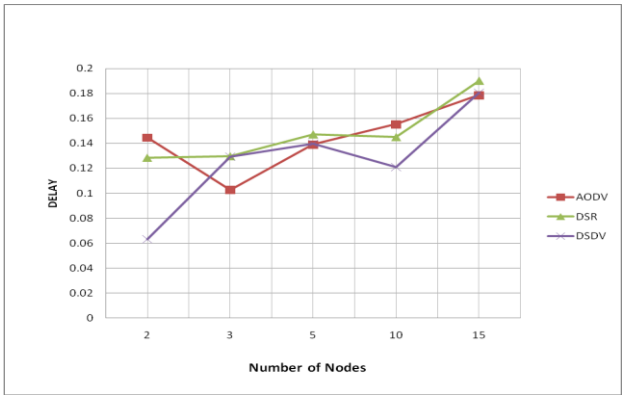
Packet Size	512 bytes
Routing Protocol	AODV, DSR, DSDV
MAC Protocol	IEEE802.11
Channel Type	Wireless Channel
Propagation Model	Two ray Ground Model
Pause Time of Mobile Nodes	0s
Speed of Mobile Nodes	0, 5, 10, 15, 20, 25 m/s
Traffic Model	Random
Motion Model	Random

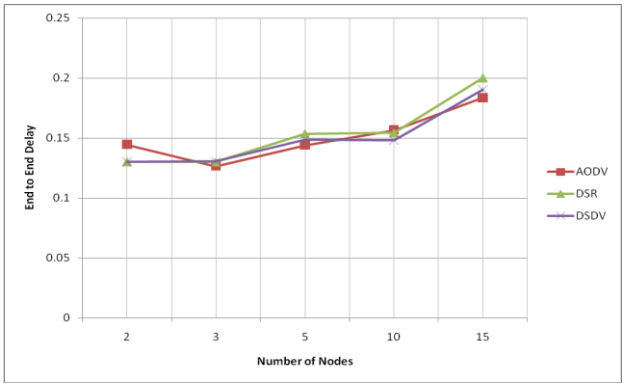
In order to ease the process of extracting data for performance study, the NS-2 Trace files can be generated. The trace files would capture information that could be used in performance study, e.g. the amount of packets transferred from source to destination, the delay in packets, packet loss etc. Unfortunately ns-2 does not provide tools to represent results. To simulate a certain network scenario, NS-2 users need to write a simulation script, save it and invoke the NS2 interpreter. After that NS- 2 will simply store the results in form of trace files. That means we need to write our own scripts to analyse the trace files. We have used different "awk scripts" to analyse the trace file.

A. Performance of TCP NewReno over different Routing Protocol

Now we present the performance of TCP NewReno over different routing protocols using the abovementioned performance metrics.







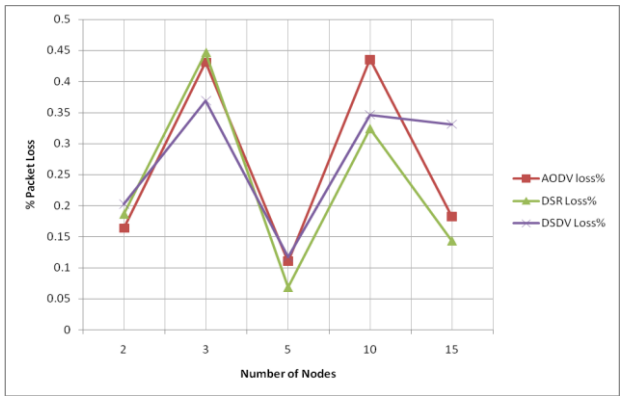


***Fig. 3* a)** Throughput b) Delay c) End to End Delay d) % packet loss

- ***Throughput:*** The figure 3 a) shows the average throughput Vs number of nodes in network. As we can see throughput decreases with increasing number of nodes. With increasing number of nodes number of connections also increases and hence the congestion also increases which in turn result in packet losses. Here DSDV perform worst among all three routing protocol. Since DSDV is table driven protocol routing table at each node maintains exhaustive information about network topology. Adaption of network topology is associated with large overheads. The on demand protocol on the other hand searches for the route when it is needed so they outperform DSDV irrespective of number of nodes. DSR performs better the AODV when number of nodes is less as it maintains the route in route cache, but as soon as number of nodes increases and in turn number of connections increases then the stale route problem of DSR comes active and makes the performance worse.
- ***Delay:*** Figure 3 b) shows the average delay Vs number of nodes. Delay also increases as number of nodes increases. With increasing number of nodes number of connections increases and that means traffic also increases. Therefore packets need to wait in queue at intermediate nodes which increases the average delay. When the route breaks occur, TCP halves it congestion window and starts the slow start procedure after the TCP timeout expiry period, tending to the increased delay. Here delay of DSDV is always low as it is a proactive protocol and discovers route proactively. Performance of reactive protocols DSR and AODV are comparable.
- ***End to End Delay:*** Figure 3 c) shows the average end to end delay of the three routing protocol and end to end delay also increases with increasing number of nodes. Here also DSDV performs better than the reactive protocols but performance of other two protocols AODV and DSR is comparable.
- ***Percentage Packet Loss:*** Packet loss rate is shown in figure 3 d). As number of connections increases the % packet loss decreases as loss probability decreases with number of connections increasing. That means as the number of packet sent are high then % packet loss will be low.

Our study concludes that the reactive routing protocols results in higher throughput and low packet loss ratio as compared to proactive protocols. On the other hand proactive protocol performs better in terms of delay and end-to-end delay.

V. MODIFICATION IN TCP NEW RENO

It was proved by our simulation study that TCP performance is highly influenced by the dynamic nature of MANET. New-Reno suffers from the fact that it takes one RTT to detect each packet loss [6,7]. When the ACK for the first retransmitted segment is received only then can we deduce which other segment was lost. The lost packets should be retransmitted immediately (Fast Retransmit/Fast Recovery) after identifying the packet loss. Also, the choice of ad hoc routing protocol to be implemented within the network affects TCP behaviour within this network. Packet loss can dramatically degrade the performance of any network protocol thus making routing in mobile ad hoc networks extremely challenging.

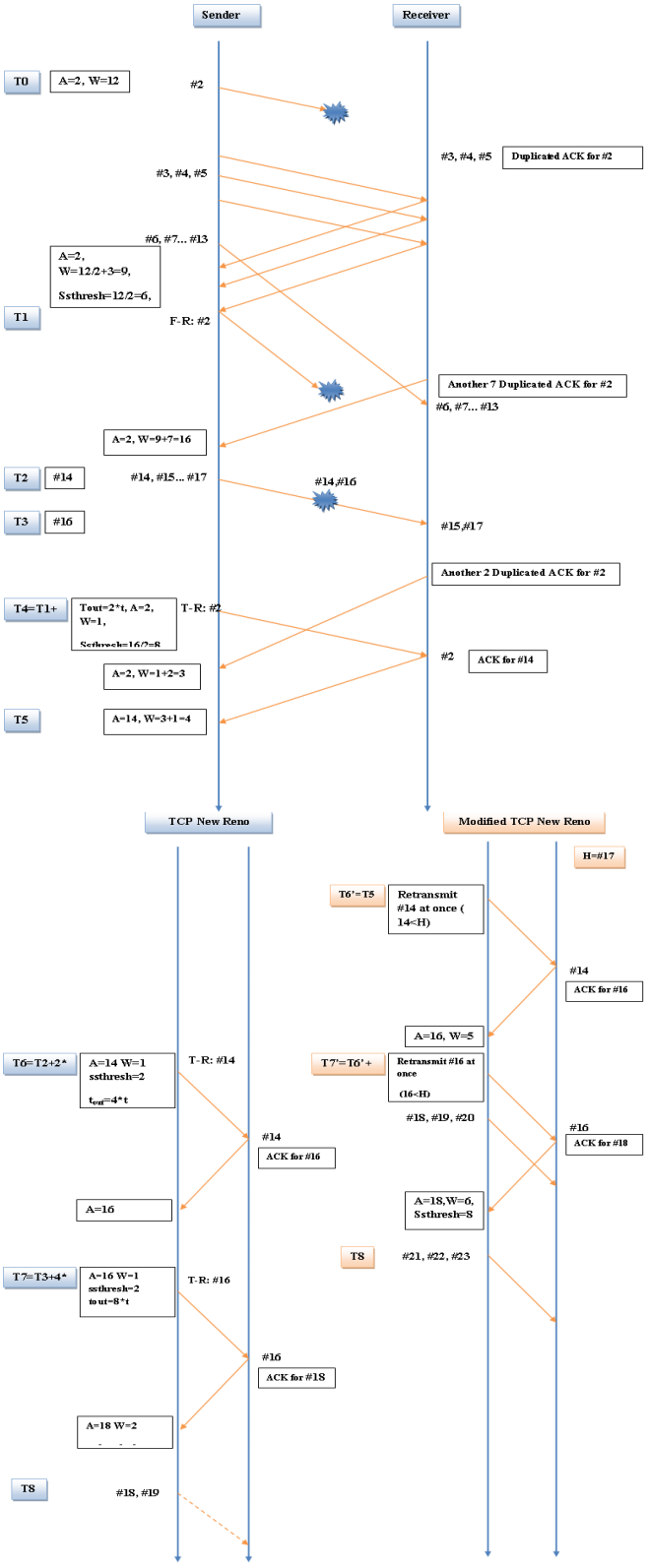


Fig. 4 Working of Modified NewReno

From the above discussions it is clear that it is very important to rapidly handle the packet losses of TCP in MANET. But TCP is very conservative in handling packet losses. In this modification to TCP-NewReno, when the retransmitted packet is acknowledged, the packet sent before it should also be acknowledged if no loss happened. So unacknowledged packets are considered lost and only they are retransmitted. This may lead to unnecessary retransmission in case of false triggered retransmission (due to underestimated RTT) and which in turn may lead to little drop in throughput. Here we need a variable to store highest sequence number before retransmit in TCP source. We have called this variable as recovered2. If the acknowledgement of retransmitted packet does not cover recovered2, we retransmit the indicated packet at

once. The other algorithms of TCP-NewReno are unchanged. The following example in figure 4 explains working of modified TCP NewReno algorithm for MANET.

We also aim to adjust data transmission rate during retransmission based on whether the packet loss is due to link failure or due to congestion. In wireless network packet loss can also be because of link disconnection due to the movement in mobile node. TCP (mainly built for fixed network) misinterpret this loss as congestion and invokes congestion control. This leads to unnecessary retransmission and loss of throughput. To overcome this problem we are proposing a scheme in which information is sent back to the source that mobile node has moved and link is broken between the source and destination. Once the source is informed it will stop the process of retransmission and congestion control. Now instead of invoking congestion control blindly, source will focus on re-establishing the connection. Here, as soon as an intermediate node detects the disruption of the route due to node mobility it sends the route failure notification to the source.

In order to show the performance improvement of Modified TCP NewReno we compared it to TCP NewReno under different routing protocols. As far as packet loss ratio packet loss ratio is concerned the modified version outperforms TCP-NewReno (refer figure 5). Hence this modification will be advantageous for networks where packet losses are frequent.

The performance comparison of modified NewReno with TCP NewReno is shown in figure 5. Figure clearly shows the performance improvement in the percentage packet loss ratio in the modified NewReno over TCP-NewReno over various ad hoc routing protocols.

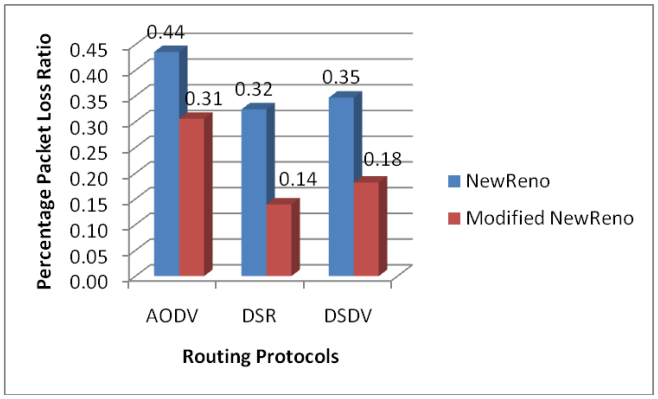


Fig. 5 Performance of Modified NewReno over NewReno

VI. CONCLUSION

We have analyzed the performance of TCP-NewReno over different routing protocol, mobility speed and number of nodes in the network (number of connections). In this study we have concluded that reactive routing protocols results in higher throughput and low packet loss ratio as compared to proactive protocols. On the other hand proactive protocol performs better in terms of delay and end-to-end delay. Finally, we have modified TCP-NewReno and shown an improvement in percentage packet loss ratio.

In our future work we aim to further enhance the performance of TCP NewReno in MANET.

REFERENCES

[1]. Ameer Ahmed, M.H.Zaidi, Kashif Sharif, "Enhancing TCP Performance in Mobile Ad hoc Networks", National Conference on Emerging Technologies 2004.

[2]. Dong Lin, H.T. Kung , "TCP Fast Recovery Strategies: Analysis and Improvements", INFOCOM'98.

[3]. Lianghui Ding, Wenjun Zhang, Wei Xie , "Modeling TCP Throughput in IEEE 802.11Based Wireless Ad Hoc Networks", Communication Networks and Services Research Conference, 2008, IEEE.

[4]. S. A. Ade, P.A.Tijare , "Performance Comparison of AODV, DSDV, OLSR and DSR Routing Protocols in Mobile Ad Hoc Networks", International Journal of Information Technology and Knowledge Management July-December 2010, Volume 2, No. 2, pp. 545-548.

[5]. Yuvaraju B. N, Niranjan N Chiplunkar , "Scenario Based Performance Analysis of Variants of TCP using NS2-Simulator", International Journal of Advancements in Technology Volume 4– No.9, August 2010

[6]. Hu Jing, Li Zhengbin, Niu Zhisheng, "A Modified TCP-NewReno Retransmission Scheme for Lossy Network", Fourth Optoelectronics and Communications Conference, 204 - 208 vol.1 , 1999, IEEE.

[7]. Seddik-Ghaleb,A.;Ghamri-Doudane,Y.; Senouci, S.-M. "TCP WELCOME TCP variant for Wireless Environment, Link losses, and Congestion packet loss Models". IEEE Conferences on Communication systems and Networks. Page(s): 1 – 8, Bangalore 2009.

[8]. S. Biaz and N. H. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver", In IEEE Symposium on Application-specific Systems and Software Engineering & Technology (ASSET'99). Richardson, TX, March, pp. 10–17 1999.

[9]. T. G. Basavaraju and Subir Kumar Sarkar, "Adhoc Mobile Wireless Networks: Principles, Protocols and Applications", Auerbach Publications, 2008.

Aparna Shrivastva received B. Tech. degree from RGPV Bhopal in Computer Science and Engineering and currently she is an M-Tech Scholar , Computer Sc. Department , RadhaRaman Institute of Science and Technology, Bhopal, India.

Ashish Khare received B. Tech. degree from RGPV Bhopal in Computer Science and Engineering and currently working as Professor in the Department of, Computer Sc. and Engg , RadhaRaman Institute of Science and Technology, Bhopal, India.

Sunita Tiwari received B. Tech. degree from OIST Bhopal in Computer Science and Engineering and M.Tech from IIT Delhi in Computer Science. She is currently working as Associate Professor at Department of Computer Science, KEC, Ghaziabad, India. Her research interest includes Web Services, Location Based Services and Soft Computing and E-commerce.