Performance Evaluation of Routing Protocols by TCP Variants in Mobile Ad-hoc Networks

Senthilvelsi. M\(^1\), Muruganandam. A\(^2\)

\(^1\)Research Scholar, Don Bosco College, Dharmapuri, TamilNadu, India
\(^2\)Assistant Professor, PG and Research Department of Computer Science, Don Bosco College, Dharmapuri, TamilNadu, India

vendhurs@gmail.com, murugan_andam@yahoo.co.in

Abstract- Mobile Ad-hoc Network (MANET) is a collection of mobile devices dynamically forming a communication network without any centralized control and pre-existing network infrastructure. Due to the presence of mobility in the MANET, the interconnections between stations are likely to change on a continual basis, resulting in frequent changes of network topology. Consequently, routing becomes a vital factor and a major challenge in such a network. In this paper to study the impact of four Internet Engineering Task Force (IETF) standardized routing protocols on MANETs and thereby comprehensively analyzes their performance under varying network sizes and node mobility rates. The four routing protocols that are considered in the analysis are Optimized Link State Routing (OLSR), Ad-hoc On-demand Distance Vector (AODV), Dynamic Source Routing (DSR) and Temporally Ordered Routing Algorithm (TORA). In addition, from a transport layer’s perspective, it is necessary to consider Transmission Control Protocol (TCP) as well for MANETs because of its wide application, which enjoys the advantage of reliable data transmission in the Internet. However, the factors such as scalability and mobility cause TCP to suffer from a number of severe performance problems in an ad-hoc environment. Hence, it is an importance to identify the most suitable and efficient TCP variants that can robustly perform under these specific conditions. In this paper evaluate the performance of three TCP variants under a variety of network conditions. The simulations results reveal that out of the three, the Selective Acknowledgement (SACK) variant can adapt relatively well to the changing network sizes while the Reno performs most robustly in different mobility scenarios. On the other hand, the research asserts the fact of superiority of proactive protocol, over reactive and hybrid ones when routing the same traffic in the network. The reactive protocols AODV performance has been found to be remarkable.

Keywords— OLSR, MANET, QOS, OPNET

I. INTRODUCTION

The use of wireless technology has become a ubiquitous method to access the Internet or making connection to the local network due to its easier and inexpensive deployment with a possibility of adding new devices to the network at no or lower cost. Devices equipped with wireless adapters together with a wireless access point constitute Wireless Local Area Networks (WLANs). Wireless access points, representing a fixed infrastructure, allow devices equipped with wireless adapters to be linked together in a Local Area Network (LAN) and to get access to the Internet. The reliance upon an existing
infrastructure and its potential limitations on mobility can be a major drawback. Wireless-capable devices may operate as autonomous entities, communicating via multiple wireless hops without a pre-established fixed infrastructure. In the discussion that follows, such wireless-equipped devices are referred to as nodes and function as both clients and servers in the network to forward the data packets. Such a network is called a Mobile Ad-hoc Network (MANET), where the nodes employed in the network can change their location from time to time. Nodes can also join or leave the network freely and arbitrarily without any restriction. A routing protocol is mainly used to discover the shortest, most efficient and correct path(s) while providing the data transmissions between different wireless devices in ad-hoc network. In recent times, MANETs are found to be able to insert the routing functionality into the mobile nodes, which save energy for other nodes by bringing down the routing overhead in the network. This routing algorithm establishes the communications and formalizes agreement among nodes, which is essential to the overall performance of a MANET. Consequently, a variety of MANET routing protocols has evolved over recent time. Examples of such routing protocols are, among others, Optimized Link State Routing (OLSR) protocol, Wireless Routing Protocol (WRP), Ad-hoc On-Demand Distance Vector (AODV) routing protocol, Dynamic Source Routing (DSR) protocol and Temporally Ordered Routing Algorithm (TORA).

The above mentioned a transport layer protocol like Transmission Control Protocol (TCP) is also needed to establish a reliable end-to-end connection in the network. TCP dominates the connection-oriented communications and ensures a reliable data transmission over the unreliable Internet Protocol (IP). Most of the internet traffic is carried out as well as the majority of widely used applications are provided by TCP. Applications like File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP) make use of TCP/IP suite for their operation. Hence, TCP is preferred to be implemented at the transport layer of an ad-hoc network which, eventually, facilitate in connecting to the Internet, thereby providing a large extent of applications. Hence, it is highly likely for the TCP to have a trust-worthy and stable performance in MANET environment. Along with the reliability feature, the TCP facilitates in managing the flow and congestion control in the data communication mechanism. Due to the congestion problem, the network performance can go down by several orders of magnitude. As a consequence, the TCP executes four intertwined algorithms, which prevent senders from overwhelming the TCP receiver. The algorithms are defined as slow-start, congestion-avoidance, fast-retransmit and fast-recovery. By implementing these mechanisms, the TCP can realize the throughput maximization so as to maintain a high performance of the network. The congestion-control algorithms introduced in the TCP Tahoe version are: Slow start, Congestion Avoidance and Fast Retransmission. Apart from these three mechanisms, TCP Reno, TCP New Reno and TCP Selective Acknowledgement (SACK) support fast recovery algorithm.

1.1 Problem Statement

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. Several studies reveal that TCP does not perform as well in a mobile environment as it does in other networks. There are several factors that affect the TCP performance in MANETs, such as dynamic topology, shared medium, signal fading and high bit errors. The dynamic topology, the packet losses occur due to the broken routes between the nodes whereas TCP assumes that the losses are due to the network congestion. The network experiences the counterproductive invocation of congestion control mechanisms employed by the TCP. Additionally, the nodes experience hidden and exposed node-problems due to the shared medium, thereby resulting in significant performance degradation in the network. Similarly, there are other types of constraints that have to be encountered when the TCP is analyzed in the MANET environment. The study of TCP performance along with the investigation of the main factors affecting the TCP performance in a MANET environment becomes an important area of research. Since MANETs are gaining immense popularity day by day, it is important to address the issue of constructing and developing an efficient MANET routing protocol. This is a formidable task as all routing protocols developed for MANET may not lead to adequate performance. Hence, it is now widely recognized that determining the specific routing protocols that can perform better in a given MANET scenario would be an important contribution to contemporary research.

The dynamic nature of MANETs, the routing mechanism experiences a host of problems by being more susceptible to errors compared to that in a wired network. In particular, member nodes can be affected by churn leading to routes disappearing and re-appearing, which in turn leads to sudden packet losses and higher message delays in the network. Hence, the routing in MANET is becoming more complex compared to in a typical wired LAN or ad-hoc network. Similarly, there are other factors like network size, network load, bandwidth and signal strength that affects the performance of the MANET routing protocols. An analysis is required in order to gain an insight of these factors that determine the performance of the routing protocol. The most important study, how the different network parameters and protocols interact,
and to what extent each of the individual factors affects the routing performance observed from the transport layer, i.e., the TCP.

1.2 Motivation and Main Contribution

A MANET is an evolving technology, which offers a cost-effective and scalable method to connect wireless devices. This technology has become increasingly popular due to its potential application in many domains. For instance, such a network can be helpful in rescue operations where there is not sufficient time or resource to configure a wired network. MANETs are also very useful in military operations where the units are moving around the battlefield in a random way and a central unit cannot be used for synchronization. MANET has been considered as a convincing candidate for better wireless services, research to enhancing its functionality is still in its infancy. Currently, research has been undertaken with regard to the task of identifying more suitable routing protocols and TCP variants for MANET environment. This paper focus on three TCP variants as well as four routing protocols in order to assess their performance in a few realistic MANET scenarios, which will eventually help to better understand their comparative merits and suitability for deployment under different network scenarios.

Among several TCP variants, three types are considered important for our investigation, namely TCP Reno, TCP New Reno and TCP Selective Acknowledgment (SACK). These three variants are reckoned as the most prominent transport layer mechanisms, which offer standard window-based congestion control algorithms. Apart from that these variants have readily available implementations in most of the network simulators. Again, for the routing protocols, we select two reactive routing protocols, such as AODV and DSR, one proactive routing protocol[1] such as OLSR, and one hybrid routing protocol[2] such as TORA. We choose these protocols since they cover a range of design choices, including source routing, hop-by-hop routing, periodic advertisement, and on-demand route discovery. The choice of these four protocols is also motivated by the fact that they have been proposed in the Internet Engineering Task Force (IETF) MANET Working Group. Even though many MANET routing protocols have been proposed in recent years, current literature reports only a limited amount of performance study between them. More specifically, very few researches had previously been attempted to contrast their performance in a realistic manner. This research provides a realistic and quantitative performance analysis of several key routing protocols in the same framework within the MANETs. In recent years, one of the purposes of existing research is to make improvement to the overall TCP performance for MANET scenario. However, prior to making such improvement, it would be worthwhile to investigate as to what extent the TCP performance is degraded in MANET environment when subjected to different network stresses and topology changes. To the best of our knowledge, this study would be first of its kind, in undertaking experiment through analyzing the performance of three TCP variants (Reno, New Reno and SACK) and four routing algorithms (DSR, TORA, OLSR, AODV) in a MANET environment.

1.3 Proposed System

In this paper, the main challenges affecting the performance of TCP in a MANET environment. Subsequently, the research investigates how well the mentioned TCP variants respond to various performance differentials, such as download response time, upload response time and retransmission attempt, aside from identifying the most suitable TCP version for a specific routing protocol in different network scenarios. Such analysis is important since it facilitates in determining the most suitable and robust TCP variant in a bid to optimizing the traffic goals in respective networks. The research also examines the routing performance with respect to TCP under a variety of network conditions. In order to evaluate such performance, end-to-end delay and throughput are considered as performance metrics. In our dissertation, a number of important system parameters such as network size and node mobility speed are taken into consideration. The changes of such parameters are made (i.e. small, medium and large size network and low, medium and high node speed) to realize different realistic MANET scenarios as well as to gauge the extent of their impact on the performance of network and transport layer protocols. In our study, all these scenarios are simulated and analyzed using Discrete Event Simulation (DES) software, namely Optimized Network Evaluation Tool (OPNET). The motivation behind using the OPNET as the selected simulator is presented in the research methodology section.

1.4 Aims and Objectives

The major objectives of the study are summarized as follows:

Apply both qualitative and quantitative research methods that will guide the study in proper direction.
Set up a platform for performing the simulation in OPNET and becoming familiar with different tools of OPNET software.

Employ the TCP congestion control algorithms during the implementation of the proposed existing TCP variants in a MANET simulation environment.

Discuss different constraints that affect the TCP performance in wireless network and critically examine various approaches that are suggested in the literature for improving the TCP performance.

Perform a simulation study of TCP’s behavior when many active flows compete for bandwidth over the same link. Simulate different routing protocols in different network scenarios against several performance metrics, and analyze the results.

Compare and analyze the protocols and the TCP versions based on their performance in the simulated environment. Draw conclusions by presenting and interpreting the outcomes.

II. TRANSPORT CONTROL PROTOCOL (TCP)

It is divided into two sections. Section 2.1 outlines the congestion control algorithms used by the TCP while a detailed description of different TCP variants is presented in section 2.2.

2.1 Congestion Control Algorithms

This section discusses about four intertwined congestion control mechanisms: slow start, congestion avoidance, fast retransmit and fast recovery. A TCP must not be more aggressive in sending data than these four algorithms allow.

2.1.1 Slow Start and Congestion Avoidance

The TCP sender employs the slow start and congestion avoidance algorithms to avoid more data to be sent in the network. The TCP sender imposes the congestion window while the receiver imposes the advertised window. The minimum of the congestion window and the advertised window regulates the data transmission. Besides, the slow start threshold (ssthresh), known as a state variable, is used to decide which one is to be used among the slow start or congestion avoidance algorithms for controlling the data transmission. During the beginning of the transmission, there are many unfamiliar conditions present in the network; therefore TCP needs to gradually discover the network by assessing the bandwidth and determining the available capacity. This will eventually prevent the network from being congested with large bursts of data. Figure 2.1 as shown below the slow start and congestion avoidance mechanisms executed by the TCP. Upon establishing a new connection, TCP starts the slow start mechanisms and sets the congestion window size to one segment. The congestion window size is incremented by one for each ACK received by the TCP sender. Thus, 1 packet is sent in the first Round Trip Time (RTT), 2 packets are for the second RTT, 4 packets are for the third RTT and continue incrementing exponentially. This is why slow start phase is also known as the exponential growth phase where slow start increases the window size by the number of segments acknowledged. This process will be continuing until either of the following situations occurs:

1) An acknowledgment is not received for some segments
2) A predetermined slow start threshold value is reached
3) The congestion window size becomes equal to the receiver’s advertised window size.

If either of these events takes place, TCP enters the congestion avoidance (linear growth) phase. Each time an ACK is received, congestion avoidance suggests that the congestion window size should be increased by (segment size × segment size) / congestion window. Here, segment size and congestion window is maintained in bytes.
2.1.2 Fast Retransmission and Fast Recovery

A packet segment is transmitted; TCP sets a timer each time and thus ensures the reliability. TCP retransmits the packet, if it does not obtain any acknowledgement within the fixed time-out interval. The reason for not getting any acknowledgement 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 repairing the loss. This fast retransmit phase is applied mainly based on the incoming duplicate ACKs. 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. 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 without waiting for a retransmission timer to expire.

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. 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, 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 fast recovery algorithm, congestion avoidance phase is again invoked instead of slow start phase as soon as the fast retransmission mechanism is completed.

2.2 TCP Variants

The original design of the Transmission Control Protocol (TCP) worked reliably, but was unable to provide acceptable performance in a large and congested network. The development of the TCP has therefore been made progressively since its original incarnation in 1988. This section presents several TCP versions which have been proposed with different mechanism in order to control and avoid the network congestion.

2.2.1 TCP Tahoe

The earlier versions of TCP offered a go-back model which used to implement the cumulative positive acknowledgment. For this purpose, retransmit timer expiration was needed in order to re-transmit the lost data. However, these former versions were unable to reduce the network congestion. Hence, for modification to earlier TCP implementations, the TCP Tahoe variant was implemented with slow-start, congestion avoidance, and fast retransmits algorithms. This version modified the Round Trip Time (RTT) estimator which is needed for adjusting the values of Retransmission Time Out (RTO). In Tahoe version, when the sender accepts three duplicate acknowledgments for a single TCP segment, it assumes that a data packet is lost and hence resends the data packet regardless of the expiration of the retransmission time. To identify a packet loss, the TCP Tahoe version needs a complete timeout interval or even longer sometimes due to the coarse grain timeout. In addition, upon detection of a packet loss, every time it waits until the pipeline is emptied which eventually establish a high cost in the band-width delay product links.
2.2.2 TCP Reno

The implementation of Tahoe, the TCP Reno version adds more mechanisms so as to detect the lost packets in shorter time and also prevent the pipeline from being empty every time a packet is lost. The packet segment is assumed to be lost as soon as the duplicate acknowledgements are reached to its threshold level. Then the TCP enters the Fast Retransmit phase through which the lost segment is retransmitted. When the Fast Retransmit phase is completed, TCP Reno employs the Fast Recovery algorithm which does not let the pipeline to be empty and also provides extra incoming duplicate ACKs to clock subsequent outgoing packets. Fast Recovery assumes whenever a duplicate ACK is attained, each time there is a single packet left in the pipe. As a result, the TCP Reno sender is capable of making sharp estimation over the amount of outstanding data in the network. Meanwhile, after entering the Fast Recovery phase, the TCP sender waits until half a window of dup ACKs are achieved, and then transmits a new data packet for each additional dup ACK. Finally, the sender leaves the Fast Recovery phase when it receives a new ACK for the new data.

The variant TCP Reno can smoothly detect the single packet drop; however this version experiences difficulty in case of multiple packets dropped from the window and the performance becomes almost as like as Tahoe version. When multiple packets are dropped, the loss information of the initial packet is arrived after the reception of the duplicate ACK. On the other hand, the information about the second packet is obtained after the acknowledgement of the retransmitted initial packet is reached to the sender. This ACK of the retransmitted initial packet is arrived after one RTT and hence it takes longer time to process the second packet loss.

2.2.3 TCP New Reno

In case of multiple packet loss, the TCP New-Reno does not wait for the retransmission timer to be expired and hence this variant provides a dominating performance over the Reno version. In New Reno, the performance concerns about the behavior of the partial ACKs, which do not take TCP out of Fast Recovery phase while it takes TCP out from the Fast Recovery phase in Reno version. Moreover, in New-Reno, receiving partial ACKs often indicates the loss of the packets which instantly follows the acknowledged packet in the sequence space. Thus for the multiple packet losses, the New-Reno becomes able to retransmit all the packets lost from a particular window and therefore the New-Reno does not leave the Fast Recovery phase unless the acknowledgement for all outstanding data in the network is completed. However, New-Reno may experience poor performance as it takes one RTT for identifying the packet loss and therefore it is possible to infer about the information of other lost packet only when the ACK for the first retransmitted segment is received.

2.2.4 TCP SACK

TCP uses a cumulative acknowledgment scheme through which only a single lost segment can be detected per Round Trip Time. This scheme does not allow the received packets that are not at the left edge of the receiver window to be acknowledged. Hence, in order to discover the lost packet, the sender has to either wait for a Round Trip Time or retransmit the received packet unnecessarily. Consequently, TCP loses its ACK-based clock and thus decreases the overall throughput. In order to overcome these limitations, A SACK mechanism, combined with a selective repeat retransmission policy is developed. TCP SACK is basically an upgraded version of TCP New Reno which takes steps to solve the major problems experienced by the New Reno version. Such problems include the detection of multiple lost packets and retransmission of more than one lost packet per RTT. The Selective Acknowledgments (SACK), the information about the arrived data segments can be reached successfully to the sender. As a result the sender only needs to retransmit the actual lost packet. The TCP SACK offers a significant feature so that the segments are acknowledged selectively instead of being acknowledged cumulatively.

In the Fast Recovery stage of SACK version, a variable is maintained by the sender in order to measure the number of outstanding data in the network. This variable is called a pipe and it is not maintained in any of the earlier TCP versions. As long as the estimated number of outstanding packets is found below than the congestion window value, a data is transmitted or retransmitted by the sender. When the sender sends a new data or retransmits an old packet, the variable pipe is incremented by one while it is decremented by the same value upon receiving a duplicate ACK with a selective acknowledgment option. Though TCP SACK provides many advantages, it is not an easy task to implement selective acknowledgment options in TCP SACK version. Hence, currently the TCP receivers are found to be reluctant for providing the selective acknowledgment option.
III. Routing Protocol

3.1 Routing Protocols

AODV and TORA are on-demand algorithms. Unlike proactive protocols such as OLSR, on-demand protocols do not maintain routes between all the nodes in an ad hoc network. Rather, routes are established when needed through a route discovery process in which a route request is broadcast. A route reply is returned either by the destination or by an intermediate node with an available route. Route error messages are used to invalidate routing table entries when link failures are detected. The Ad-hoc On-Demand Distance Vector (AODV) routing protocol is an improvement of the Destination-sequenced Distance Vector (DSDV) routing protocol. It is based on distance vector and also uses the destination sequence numbers to determine the freshness of the routes. AODV requires hosts to maintain only active routes. The advantage of AODV is that it tries to minimize the number of required broadcasts. It creates the routes on an on-demand basis, as opposed to maintain a complete list of routes for each destination. Therefore, the literature on AODV, classifies it as a pure on-demand route acquisition system. TORA is a reactive routing algorithm based on the concept of link reversal and used in MANETs to improve the scalability. Highly dynamic Mobile Ad-hoc Networks can be used by TORA [4][1]. It is an adaptive routing protocol used in multi-hop networks. It makes scaled routes between source and destination. There are three basic functions in TORA: Route Creation, Route Maintenance and Route Erasure. OLSR is a proactive link-state routing protocol, which uses Hello and Topology Control (TC) messages to discover and then disseminate link state information throughout the Mobile Ad-hoc Network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths.

3.2 Transport Protocols

After the introduction of first version of TCP, several variants are introduced, here we are discussing the most famous implementation of TCP called Tahoe, Reno, New Reno and SACK.

TAHOE

In the first version of TCP there was no congestion control mechanism. So after observing the congestion, Jacobson introduced several Congestion Control algorithms and this version is called TCP-Tahoe. The congestion control algorithms introduced in this version are: Slow Start, Congestion Avoidance and Fast Retransmit.

RENO

TCP Reno is the most widely adopted Internet TCP protocol. It employs four transmission phases: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

NEW RENO

TCP New Reno is a modification of TCP Reno. It improves retransmission process during the fast recovery phase of TCP Reno. TCP New Reno can detect multiple packet losses and does not exit the fast recovery phase until all unacknowledged segments at the time of fast recovery are acknowledged. Thus, as in TCP Reno, it overcomes reducing the congestion window size multiple times in case of multiple packet losses. The remaining three phases (slow start, congestion avoidance, and fast retransmit) are similar to TCP Reno.

SACK

SACK algorithm allows a TCP receiver to acknowledge out-of-order segments selectively rather than cumulatively by acknowledging the last correctly in order received segment. The receiver acknowledges packets received out of order and the sender then retransmits only the missing data segments instead of sending all unacknowledged segments.

3.3 OLSR

The Optimized Link State Routing (OLSR) is operated as a proactive (table-driven) routing protocol i.e. frequently exchanges topology information with other nodes of the network. This protocol is basically an optimization of traditional link state protocol developed for Mobile Ad-hoc Network. The responsibilities of OLSR protocol are to minimize the required number of control packets transmission and also to shorten the size of control packets. On top of that, OLSR trims down the control traffic overhead in the network with the help of Multipoint Relays (MPRs). The MPR concept is the key idea
behind OLSR protocol which is basically a node's one-hop neighbors in the network as shown in Figure 5.1. For route calculation, the MPR technique is employed in order to form the route between the source and the destination in the network. In addition, the MPRs yield an efficient mechanism for flooding the control traffic by significantly minimizing the number of packet transmissions. Yet, the MPRs are to be involved in another task when the link state information is declared in the network. The task includes declaring the link-state information for their MPR selectors and hence providing the shortest paths to all destinations. In MANET, the MPRs are assigned from the one-hop adjacent nodes with “symmetric” (bi-directional) linkages. Thus, by determining the path through the multipoint relays, it is possible to keep away the difficulties experienced during the packet transmission over a uni-directional link.

![Figure 3.1 Multipoint Relays of the OLSR network](image)

OLSR employs three different types of control message namely

1) HELLO,
2) Topology Control (TC), and
3) Multiple Interface Declaration (MID).

### 3.3.1 AODV

Ad-hoc On-Demand Distance Vector (AODV) is considered an efficient MANET routing protocol and supports both unicast and multicast routing mechanisms. The AODV routing protocol utilizes an on-demand technique in order to discover the routes. This means that the route between two endpoints (nodes) is formed as per requirement for the source node and maintained as long as the routes are needed. The protocol uses a destination sequence number to recognize the most recent path and to guarantee the freshness of the routes. Reactive protocols like AODV shrinks the control traffic overhead at the cost of higher latency in discovering new routes\(^\text{[4]}\). Although AODV is a reactive protocol, some characteristics of a proactive protocol are often followed by this. For instance, the protocol broadcasts the periodic HELLO messages to notify the neighbor nodes that the link is still active. AODV does not have any function until there is a valid route between the source and the destination in MANET. Upon requiring the formation of a new route, the source node transmits a Route Request (RREQ) packet. After flooding the RREQ packet, the source node waits until a Route Reply (RREP) packet is received as an acknowledgement. However, within a specific time, a RREP may not be received and in that case a new RREQ is to be sent again by the source node. And for this additional transmission of RREQ, the predefined waiting interval needs to provide a binary exponential back-off and therefore it is multiplied by two \( (2) \) each time. The binary exponential back-off must be utilized in order to reduce the network congestion. After receiving a RREQ, the neighbor node either generates a RREP message to the sender or rebroadcasts the RREQ depending on the availability of a valid route to the destination. The validity of the route is confirmed after making a comparison between the sequence numbers of the intermediate node and the destination sequence number of the Route Request packet. Once the RREP is received by the source node, it stores the information of this particular route and starts transmitting data toward that destination. However, in case of the reception of the multiple RREPs, the route with the shortest hop count will be selected. In case a link failure is experienced, a Route Error (RERR) message is created and returned to the originator of the data in a hop-by-hop fashion and the process replicates. The purpose of generating the RERR message is to inform other nodes about the current broken link. The source node disables the route as soon as it receives the Route Error message and invokes the route discovery mechanism again if it is necessary.
3.3.2 DSR

Dynamic Source Routing (DSR) is a widely used reactive (on-demand) routing protocol which is designed particularly for the Mobile Ad-hoc Networks. DSR permits the network to run without any existing network infrastructure and thus the network becomes as a self-organized and self-configured network. This protocol maintains an on-demand approach and hence extinguishes the periodic table-update messages needed in the table-driven approach[40]. Consequently, it is able to prevent the control packets from consuming much bandwidth. Like other on-demand routing protocols, DSR does not provide the transmission of any periodic hello packet (beacon), which is essential for informing its presence to other nodes. Instead, during the route construction phase, it establishes the route by flooding a Route Request packet in the network. Each Route Request packet holds a sequence number which is generated by all the nodes through which the packet is flooded. By using this sequence number, loop formation and multiple transmission of the same Route Request is possible to be evaded. When a Route Request packet is reached to its final destination, the destination node sends a Route Reply packet to the source node through the opposite way the Route Request is travelled. Since, it cannot be an efficient mechanism for the nodes to provide continuous flooding; DSR utilizes the route caches to store the routing information[9]. In MANETs, the DSR protocol generates two mechanisms namely route discovery and route maintenance for the purpose of discovering and maintaining the route between the endpoints. Both mechanisms are utilized to support the unidirectional (asymmetric routes) links in wireless ad-hoc network.

3.3.3 TORA

The Temporally Ordered Routing Algorithm (TORA) is a highly efficient distributed routing protocol and known as a hybrid protocol which can simultaneously support both table-driven and on-demand approach in multi-hop wireless networks. This protocol belongs to the family of the link reversal routing mechanism based on the Gafni-Bertsekas (GB) and Lightweight Mobile Routing (LMR) algorithms. The TORA protocol’s reaction to link failure is structured as a temporally ordered sequence of diffusing computations, where all computations comprising of a sequence of directed link reversals. TORA is implemented on the top of the Internet MANET Encapsulation Protocol (IMEP) and ensures link status sensing, reliability, loop-free routes, multiple routes and many others essential services. TORA implements four mechanisms in the network, which are known as creating routes, maintaining routes, erasing routes, and optimizing routes.

IV. EXPERIMENTAL RESULTS AND COMPARISON

4.1 Varying Network Size

The performance analysis of different routing protocols and TCP variants within MANET has been carried out through these scenarios where each scenario is presented against node sizes representing small, medium and large network. The routing performance is evaluated using TCP SACK variant since this is considered as a newer and widely deployed version now-a-days [8]. On the other hand, the performances of different TCP variants are assessed with DSR routing protocol as the DSR interacts with TCP more efficiently than the other protocols under different realistic MANET scenarios. To observe the impact of node variation on routing and TCP performance, the target applications are run with various network sizes (30, 60 and 100 nodes). Though this section deals with network size issue; it is much more realistic for a MANET environment to generate at least a low mobility rate instead of keeping it fully static. Accordingly, a moving speed of 10 m/s with an average pause time of 100 sec is set to allow the mobile nodes to move slowly in the network.

4.2 Impact on Throughput

Throughput refers to the amount of traffic successfully received by the destination node. The routing efficiency can be predicted by observing the overall throughput received by the network.

© 2015, IJCSMC All Rights Reserved
Figure 4.1: Average throughputs for different routing protocols; (a) Small network size (node=30), (b) Medium network size (node=60), (c) Large network size (node=100)

4.3 Impact on End-to-End Delay

End-to-end delay for a data packet is measured from the time it is created to the time it is received. High end-to-end delay indicates more broken links and frequent re-routing during the transmission of the data packet.
4.4 Summary of TCP Performance

The simulation results reveal some important MANET characteristics which have impacted on the TCP performance. The changes in network size and mobility rate to higher values result in a variety of reactions to different TCP versions. In this section we have summarized the performance results of Reno, New Reno and SACK variants within the MANET environment. The performance is studied in terms of download response time; upload response time and retransmission attempts. With a large number of nodes, the network is supposed to experience extra high loads and thereby the TCP performance is expected to be dramatically affected. In what follows, the summary results are displayed, followed by a brief discussion for each case. In Figure 4.3, different TCP versions under study are analyzed in accordance with the download response time.

Figure 4.3 Performance of TCP variants for different network sizes; (a) In terms of download response time (b) In terms of upload response time (c) In terms of retransmission attempts.
V. CONCLUSION AND FUTURE WORK

Mobile ad-hoc networking has received a lot of attention as the wireless networking and the mobile computing devices are now capable of supporting the requirements of such technology. TCP optimization in such a network has become a challenging matter because of some unique characteristics of MANETs. This research makes contribution in three areas. Firstly, the study undertakes an analysis towards a comprehensive performance evaluation of four IETF standardized routing protocols in a MANET environment. The considered routing protocols are DSR, AODV, OLSR and TORA, covering a range of design choices, including source routing, hop-by-hop routing, periodic advertisement and on-demand route discovery. Secondly, the study analyzes the performance of the three most widely used TCP variants (Reno, New Reno and SACK) in an ad-hoc environment. In this respect, an investigation is made into aspects as to how well these variants respond to different network conditions, particularly with respect to extension of network size and variation of mobility rate. Finally, using the simulation environment, an analysis is carried out on the results of throughput, end-to-end delay, upload response time, download response time and retransmission attempts. These results have facilitated in determining the most suitable routing protocols and TCP variants that can perform more efficiently and robustly in a mobile ad-hoc network. Each of the protocols and TCP variants studied in this research are found to have performed well in most cases. Obviously, our future work will address all the limitations mentioned in the previous section. Aside from that, many interesting issues have surfaced during the course of this study, which need to be mentioned to give others some future research directions. For instance, in our dissertation, we have considered two network factors (node size and mobility); the pursuit of future research may include aspects relating to evaluation of the MANET performance under other important factors like network load and transmission range. In this dissertation, a comparative analysis on four MANET routing protocols (viz. OLSR, AODV, DSR and TORA) has been carried out to evaluate their performance, the outcomes of which would be useful in many other situations. However, there are other protocols such as DSDV, ZRP and SSR that can be pursued in any future research. Aside from this, an investigation as to how ad-hoc network performance can be improved, using the cross-layer interactions can also be an important area of future research. Furthermore, since a MANET is formed without centralized controls, it is posing vulnerable to security attacks now-a-days. Hence, in any future study, such security issues in an ad-hoc network can be pursued.

References


**AUTHORS’ PROFILES**

**Mrs. M. Senthamilselvi** received the M.Sc., in Government Arts College, Dharmapuri in 2011 respectively, doing her M.phil., in the specialization of Computer Science in Don Bosco College, Dharmapuri, Tamilnadu. She is a Research Scholar in the field of Mobile Ad hoc Networks. Her ongoing research focused on Routing Protocol in Wireless Sensor Network.

**Mr. A. Muruganandam** received the M.Sc., in Thanthai Hans Roever College, Perambalur, and M.Phil., degree in the specialization of Computer Science from Manonmaniam Sundaranar University, in 1999 and 2004 respectively. He is an Assistant Professor cum Head in the Research Department of Computer Science, Don Bosco College, Sogathur, Dharmapuri, Tamilnadu, India. Presently pursuing Doctorate Degree [Ph.D.] in Computer Science at Bharathiar University, Coimbatore. He is a Research Scholar in the field of Wireless Sensor Networks.