Survey of Performance based Transmission Control Protocol in MANET

Sapna Bagde[1]
Department of Information Technology
Barkatullah University Institute of Technology
Bhopal, M.P., India.
Prof. Poonam Sinha
Department of Information Technology
Barkatullah University Institute of Technology
Bhopal, (M.P.), India
Asst. Prof. Ashish Jain
Department of Information Technology
Barkatullah University Institute of Technology
Bhopal, (M.P.), India

Abstract: Transmission Control Protocol (TCP) is a connection-oriented transport service that ensures the reliability of message delivery. It verifies that messages and data were received. TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. TCP provides a communication service at an intermediate level between an application programs. TCP is the protocol used by major Internet applications such as the World Wide Web, email, remote administration and file transfer. TCP is a reliable transport protocol that is well tuned to perform well in traditional networks. However, several experiments and analysis have shown that this protocol is not suitable for bulk data transfer in high bandwidth, large round trip time networks because of its slow start and conservative congestion control mechanism. In this paper we discussed a survey of Performance Based Transmission Control Protocol in Mobile Ad-hoc Network environment. The performance based techniques are categorized based upon different approaches like throughput, end-to-end delay, congestion control etc. We also analysis the major improvement in recent methods for performance based TCP in MANET.

Keywords: MANET, TCP Techniques, Performances Analysis.

Introduction

Ad hoc networks are complex distributed systems that consist of wireless mobile or static nodes that can freely and dynamically self-organize. In this way they form arbitrary and temporary ad-hoc network topologies, allowing devices to seamlessly interconnect in areas with no pre-existing infrastructure. Depending on whether the wireless nodes are mobile or static, an ad hoc network is called a Mobile Ad hoc Network (MANET). TCP was designed to work in wired networks and because of that its performance is quite poor when it is required to work in the typically loss wireless environment. Unlike cellular networks, where only the last hop is based on a wireless medium, ad hoc networks are composed exclusively of wireless links, where multihop connections may be in place. Besides, in an ad hoc scenario all nodes can move freely and unpredictably, which makes the TCP congestion control quite hard since it is a clock based mechanism. Consequently, the error-detection and error-recovery strategies inherent in standard TCP need to be adapted in order to fit this environment. In particular, since the errors in this environment occurs not only due to congestion but also due to medium constraints and mobility, TCP needs to distinguish the nature of the error so that it can take the most appropriate action for each case. Additionally, the emerging link and network layer algorithms for this kind of network can play a key role on TCP performance. Likewise, factors such as path asymmetry (that may also be caused by lower layers strategies, among other elements) and congestion window size might also impair the performance of this protocol. Although there are a number of differences between cellular and ad hoc networks, some of the ideas developed in some proposed solutions for the former can be used in the latter as well. In ad hoc environment TCP can be led to persist mode whenever a long disconnection occurs and under wireless medium induced losses it can simply retransmit the lost packet instead of invoking its congestion control mechanism. Both ideas have already been evaluated for cellular networks where only the last hop is of concern. Ad hoc networks pose some tough challenges to TCP due to the fact that it was not designed to operate in such a highly dynamic scenario in terms of topology. In reality, even though TCP has evolved significantly over the years toward a robust and reliable service protocol, the focus has been primarily on wired networks. In this scenario, the additive-increase/multiplicative decrease strategies coupled with the fast recovery and fast retransmit mechanisms inherent in most of current TCP versions, provide an effective congestion control at the end nodes [1-6]. In recent years, several schemes have been proposed for satellite as well as for cellular wireless networks, but much has still to be done on mobile ad hoc framework. In this sense, it is quite important the knowledge of both the main factors affection TCP performance in such an environment and the actual potentiality of the recent existing proposed solutions, which are addressed in this paper.

Background on TCP

TCP techniques adjust the network congestion avoidance parameters of TCP connections over high-bandwidth, high-latency networks. Well-tuned networks can perform up to 10 times faster in some cases. However, blindly following instructions with understanding their real consequences can hurt the performance as well.

TCP Bandwidth-delay product (BDP):

Bandwidth-delay product (BDP) is a term primarily used in conjunction with the TCP to refer to the number of bytes necessary to fill a TCP "path", i.e. it is equal to the maximum number of simultaneous bits in transit between the transmitter and the receiver.

High performance networks have very large BDPs. To give a practical example, two nodes communicating over a geostationary satellite link with a round trip delay of 0.5 seconds and a bandwidth of 10 Gbit/s can have up to 0.5×10^{10} bits, i.e., 5 Gbit = 625 MB of unacknowledged data in flight. Despite having much lower latencies than satellite links, even terrestrial fiber links can have very high BDPs because their link capacity is so large. Operating systems and protocols designed as recently as a few years ago when networks were slower were tuned for BDPs of orders of magnitude smaller, with implications for limited achievable performance.

TCP Buffers:

The original TCP configurations supported buffers of up to 64 Kbytes (64 KB), which was adequate for slow links or links with small round trip times (RTTs). Larger buffers are required by the high performance options described below.

Buffering is used throughout high performance network systems to handle delays in the system. In general, buffer size will need to be scaled proportional to the amount of data "in flight" at any time. For very high performance applications that are not sensitive to network delays, it is possible to interpose large end to end buffering delays by putting in intermediate data storage points in an end to end system, and then to use automated and scheduled non-real-time data transfers to get the data to their final endpoints [3] and [7].

Transmission Control Protocol Working Phases:

TCP protocol operations may be divided into three phases. Connections must be properly established in a multi-step handshake process (connection establishment) before entering the data transfer phase. After data transmission is completed, the connection termination closes established virtual circuits and releases all allocated resources.

A TCP connection is managed by an operating system through a programming interface that represents the local end-point for communications, the Internet socket. During the lifetime of a TCP connection it undergoes a series of state changes:

→LISTENING: In case of a server, waiting for a connection request from any remote client.

→SYN-SENT: waiting for the remote peer to send back a TCP segment with the SYN and ACK flags set. ('SYN-SENT' state is usually set by TCP clients)

→SYN-RECEIVED: waiting for the remote peer to send back an acknowledgment after having sent back a connection acknowledgment to the remote peer. ('SYN-RECEIVED' state is usually set by TCP servers)

→ESTABLISHED: The port is ready to receive/send data from/to the remote peer.

→FIN-WAIT-1: Indicated that the server is waiting for the application process on its end to signal that it is ready to close.

→FIN-WAIT-2: Indicates that the client is waiting for the server's fin segment (which indicates the server's application process is ready to close and the server is ready to initiate its side of the connection termination)

→CLOSE-WAIT: The server receives notice from the local application that it is done. The server sends its fin to the client.

→LAST-ACK: Indicates that the server is in the process of sending its own fin segment (which indicates the server's application process is ready to close and the server is ready to initiate its side of the connection termination).

→TIME-WAIT: Represents waiting for enough time to pass to be sure the remote peer received the acknowledgment of its connection termination request. According to RFC 793 a connection can stay in TIME-WAIT for a maximum of four minutes known as a MSL (maximum segment lifetime).

→CLOSED: Connection is closed.

TCP Connection establishment:

To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the three-way (or 3-step) handshake occurs:

1. **SYN**: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
2. **SYN-ACK**: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number (A + 1), and the sequence number that the server chooses for the packet is another random number, B.
3. **ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A + 1, and the acknowledgement number is set to one more than the received sequence number i.e. B + 1.

At this point, both the client and server have received an acknowledgment of the connection [6-8].

TCP Connection termination:

The connection termination phase uses, at most, a four-way handshake, with each side of the connection terminating independently. When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. Therefore, a typical tear-down requires a pair of FIN and ACK segments from each TCP endpoint. After both FIN/ACK exchanges are concluded, the terminating side waits for a timeout before finally closing the connection, during which time the local port is unavailable for new connections; this prevents confusion due to delayed packets being delivered during subsequent connections.

A connection can be "half-open", in which case one side has terminated its end, but the other has not. The side that has terminated can no longer send any data into the connection, but the other side can. The terminating side should continue reading the data until the other side terminates as well.

It is also possible to terminate the connection by a 3-way handshake, when host A sends a FIN and host B replies with a FIN & ACK (merely combines 2 steps into one) and host A replies with an ACK. This is perhaps the most common method.

It is possible for both hosts to send FINs simultaneously then both just have to ACK. This could possibly be considered a 2-way handshake since the FIN/ACK sequence is done in parallel for both directions.

Some host TCP stacks may implement a half-duplex close sequence, as Linux or HP-UX do. If such a host actively closes a connection but still has not read all the incoming data the stack already received from the link, this host sends a RST instead of a FIN. This allows a TCP application to be sure the remote application has read all the data the former sent—waiting the FIN from the remote side, when it actively closes the connection. However, the remote TCP stack cannot distinguish between a Connection Aborting RST and this Data Loss RST. Both cause the remote stack to throw away all the data it received, but that the application still didn't read.

Some application protocols may violate the OSI model layers, using the TCP open/close handshaking for the application protocol open/close handshaking; these may find the RST problem on active close.

For a usual program flow like above, a TCP/IP stack like that described above does not guarantee that all the data arrives to the other application.

TCP Resource usage:

Most implementations allocate an entry in a table that maps a session to a running operating system process. Because TCP packets do not include a session identifier, both endpoints identify the session using the client's address and port. Whenever a packet is received, the TCP implementation must perform a lookup on this table to find the destination process.

The number of sessions in the server side is limited only by memory and can grow as new connections arrive, but the client must allocate a random port before sending the first SYN to the server. This port remains allocated during the whole conversation, and effectively limits the number of outgoing connections from each of the client's IP addresses. If an application fails to properly close unrequited connections, a client can run out of resources and become unable to establish new TCP connections, even from other applications.

Both endpoints must also allocate space for unacknowledged packets and received (but unread) data.

TCP Data transfer

There are a few key features that set TCP apart from User Datagram Protocol:

→Ordered data transfer — the destination host rearranges according to sequence number.

→Retransmission of lost packets — any cumulative stream not acknowledged is retransmitted.

→Error-free data transfer.

→Flow control — limits the rate a sender transfers data to guarantee reliable delivery. The receiver continually hints the sender on how much data can be received (controlled by the sliding window). When the receiving host's buffer fills, the next acknowledgment contains a 0 in the window size, to stop transfer and allow the data in the buffer to be processed.

→Congestion Control

TCP Reliable transmission

TCP uses a sequence number to identify each byte of data. The sequence number identifies the order of the bytes sent from each computer so that the data can be reconstructed in order, regardless of any fragmentation, disordering, or packet loss that may occur during transmission. For every payload byte transmitted, the sequence number must be incremented. In the first two steps of the 3-way handshake, both computers exchange an initial sequence number (ISN). This number can be arbitrary, and should in fact be unpredictable to defend against TCP Sequence Prediction Attacks. TCP primarily uses a cumulative acknowledgment scheme, where the receiver sends an acknowledgment signifying that the receiver has received all data preceding the acknowledged sequence number. The sender sets the sequence number field to the sequence number of the first payload byte in the segment's data field, and the receiver sends an acknowledgment specifying the sequence number of the next byte they expect to receive [1] and [3] and [5].

Performances Analysis of TCP Techniques

M. Jehan, Dr. G.Radhamani & T.Kalakumari et. al. analyzed six TCP Congestion Control Algorithms and their performance on Mobile Ad-hoc Networks (MANET). More specifically, we describe the performance behavior of BIC, Cubic, TCP Compound, Vegas, Reno and Westwood congestion control algorithms. The evaluation is simulated through Network Simulator (NS2) and the performance of these congestion control algorithms is analyzed with suitable metrics. The proposed simulations has been successfully implemented and evaluated using NS-2 simulator on a computer with Intel Core 2 Duo CPU (T6400 processor @ 2.00 GHz) 2 GB of RAM. A random wireless mobile ad hoc network topology was used for these experiments. They appraised the performance of these congestion control algorithms in very ideal condition without any cross traffic and any additional flows. In this small MANET scenario, the algorithm BIC (Binary Increase Congestion Control) provided good throughput after 75 seconds but algorithm Vegas provided stable and excellent throughput almost all over on the whole run time. So it move towards to the wrapping up that the algorithm Vegas will be the suitable algorithm for small and dynamic mobile ad hoc network scenario. Except Vegas, all other assessed algorithms provided very poor throughput during initial stage of the communication (less than 50 Seconds). They conclude TCP Vegas will be the best algorithm from the list.

In this work, they preferred six algorithms for evaluation, because they are default algorithms in several standard operating systems.

This work could extend in select few algorithms from each of these four categories and will evaluate their performance in MANET scenario. There are other varying network parameters and metrics that the authors are working on the same. Based on the results, it needs to be extending the further enhancement towards specific application on MANETs [1].

M.Jehan et. al. analyzed the TCP BIC and TCP Vegas congestion control algorithms and the performance of these algorithms through NS2 simulator with proper parameters. The tremendous growth of wireless networks demands the need to meet different congestion control algorithms for wireless network. Usual transmission control protocol reduces its performance by misinterpreting mobility losses due to node motion as congestion losses in wireless mobility network. The term network load is used to describe how much data will be transmitted over a connection with in the time duration. It is a gross measurement, taking the total amount of data transferred in a given period of time as a rate, without taking into consideration the quality of the signal itself. The ratio of the number of packets transmitted and the quantity of packets delivered to destination node within the time period.

They successfully evaluated the TCP BIC and TCP Vegas congestion control algorithms using NS2 simulation tool. The performance of these two algorithms analyzed in ideal condition without any cross traffic and any other additional flows. In this small MANET scenario, the algorithm BIC provided good throughput after 75 seconds but algorithm Vegas provided stable and excellent result almost all over on the whole run time.

This work is not sufficient reliable on performance on TCP congestion control. It could be improving the work evaluation with based on the as slow start and Congestion Avoidance [2].

Marco Di Felice et. al. proposed a new paradigm to provide additional spectrum utilization opportunities in wireless ad hoc networks. Recent research in this field has mainly focused on devising spectrum sensing and sharing algorithms, to allow an opportunistic usage of licensed portions of the spectrum by Cognitive Radio Users. However, it is also important to consider the impact of such schemes on the higher layers of the protocol stack, in order to provide efficient end-to-end data delivery. Since TCP is the de facto transport protocol standard on Internet, it is crucial to estimate its ability in providing stable end-to-end communication over Cognitive Radio Ad Hoc Networks. The contributions of this paper are twofold. First, they proposed an extension of the NS-2 simulator to support realistic simulation of CRAHNs. The extension allows modeling the activities of Primary Users (PUs), and the opportunistic spectrum management by CRUs in the licensed band. Second, it provided an accurate simulation analysis of the TCP performance over CRAHNs, by considering the impact of three factors: (i) spectrum sensing cycle, (ii) interference from PUs and (iii) channel heterogeneity.

They investigated the performance of different TCP variants over CRAHNs, i.e. TCP Reno, TCP New Reno, TCP with Selective Acknowledgment (TCP SACK) and TCP Vegas. The CRAHN environment is constructed. They assume that 5 PU (Primary Users) spectrum bands are present (i.e. B=5). Moreover, we assume that the number of CRU channels is equal to the number of primary bands (i.e. n=1), i.e. there is complete overlapping between CRU channels and PU spectrum bands. This assumption makes easier to investigate the impact of PUs interference on CRUs using the same spectrum band. Individual spectrum bands are occupied randomly and independently of each other by PUs, according to the ON/OFF model.

In the first case, the network is composed of 2 CRUs. A TCP/FTP connection is established between the static CRUs. No mobility effect is considered at this stage. They studied the performance of TCP under different CRAHNs characteristics, e.g. the sensing time interval of CRUs, interference caused by PUs activity and bandwidth variation in an heterogeneous channel environment. The choice of the single-hop scenario can be motivated firstly, the single-hop scenario is simple enough to understand the impact of CRAHNs characteristics on the dynamics inside the TCP, while this might be difficult to investigate in multi-hop topologies. Second, the single-hop topology constitutes a “base case”, from the point of view of protocol performance. If we discover that a single parameter, e.g. the sensing time interval, has a strong impact on TCP performance, then this effect would be emphasized in a multi-hop environment by the presence of multiple intermediate nodes between the source and the destination pair. Moreover, although very simple, the single-hop topology constitutes a realistic model for the evaluation of infrastructure-based CR networks, where the mobile CRUs are attached to a fixed Cognitive Base Station (CBU). In the second scenario, they considered multihop CRAHNs topologies, composed of 25 mobile CRUs. It varied the number of active TCP/FTP connections. The multi-hop scenario is used to evaluate end-to-end TCP performance when all the CRAHNs characteristics are considered.

In this paper, they addressed performance evaluation of TCP over Cognitive Radio Ad Hoc Networks (CRAHNs). To this aim, we presented an extension of the NS-2 tool for the modeling and simulation of CRAHNs. Their extension provides accurate modeling of PUs activities and of CRUs spectrum management functionalities, including spectrum sensing, decision and mobility schemes. Moreover, it provides facilities to support cross-layer information sharing among network protocols at different layers of the protocol stack.

Simulation results show that sensing time interval and type of PU activity play a critical role in deciding the TCP performance.

The simulation results highlight that transport protocols proposed for traditional wireless ad hoc networks might not work well communication over Cognitive Radio Ad Hoc Networks. Since transport layer is still an explored research area for Cognitive Radio Ad Hoc Networks (CRAHNs), studying and understanding problems of classical TCPs in the cognitive environment is fundamental to design novel transport protocol solutions for Cognitive Radio Ad-hoc Networks [3].

Jian Liu et. al. presented an approach where they implemented a thin layer between Internet protocol and standard TCP that corrects these problems and maintains high end-to-end TCP throughput. The protocol in Free BSD, and in this paper, we present results from extensive experimentation done in an ad hoc network. The solution improves TCPs throughput by a factor of 2–3. The goal in designing ATCP (Ad-hoc TCP) was to provide a complete solution to the problem of running TCP over multihop wireless networks. Specifically, they wanted to design a protocol that has the following characteristics.

1) Improve TCP Performance for Connections set up in ad-hoc Wireless Networks. TCP performance is affected by the problems of high BER and disconnections due to route re-computation or partition. In each of these cases, the TCP sender mistakenly invokes congestion control. The appropriate behavior in these cases ought to be the following.

→ High BER: Simply retransmit lost packets without shrinking the congestion window.

→ Delays due to Route Re-computation: Sender should stop transmitting and resume when a new route has been found.

→Transient Partition: As above, the sender should stop transmitting (because they did not want to flood the network with packets that cannot be delivered anyway) until it is reconnected to the receiver.

→Multipath Routing: In this case, when TCP at the sender receives duplicate ACKs, it should not invoke congestion control because multipath routing shuffles the order in which packets are received.

2) Maintain TCP’s Congestion Control Behavior. This is an important goal because if losses are caused due to network congestion, they did not want the TCP sender to assume that these losses were due to high BER and continue transmitting. In this case, they want TCP to shrink its congestion window in response to losses and invoke slow start.

3) Appropriate CWND Behavior. When there is a change in the route (e.g., a reconnection after a brief partition), the congestion window should be recomputed.

4) Maintain End-to-End TCP Semantics. It believe that it is critical to maintain end-to-end TCP semantics in order to ensure that applications do not crash.
5) Be Compatible with Standard TCP. This is necessary because we cannot assume that all machines deployed in an ad hoc network have ATCP installed. Thus, machines with or without ATCP should be able to set up normal TCP connections with machines that may or may not have ATCP.
Furthermore, applications running at machines with ATCP Should not are aware of ATCP's presence. Sometimes, it is likely that an ad hoc network may be connected to wire line networks through access points. In such situations, the sender or receiver of a TCP connection may lie in the wire line network with the other end-point in the ad hoc network. It is important to ensure that TCP connections work normally in these cases as well. The approach to the problem of improving TCP's performance while maintaining compatibility is to introduce a thin layer between TCP and IP called ATCP. In this paper, they presented a solution to the problem of running TCP in ad hoc wireless networks. The solution is to implement a thin layer between IP and ad-hoc TCP (called ATCP) that ensures correct TCP behavior while maintaining high throughput.
This is done by putting TCP into persist mode when the network is disconnected or when there are losses due to high bit error.
The Ad-hoc TCP's performance is need to improve as measured by the time to transfer large files. And also need to maintain end to end delay in ATCP congestion control behavior when there is network congestion [4].
Gavin Holland et. al. investigated the effects that link breakage due to mobility has on TCP performance. Through simulation, they showed that TCP throughput drops significantly when nodes move, due to TCP's inability to recognize the difference between link failure and congestion. They also analyzed specific examples, such as a situation where throughput is zero for a particular connection. It introduce a new metric, expected throughput, for the comparison of throughput in multi-hop networks, and then use this metric to show how the use of explicit link failure notification (ELFN) techniques can significantly improve TCP performance.
In this performance analysis, they set up a single TCP-Reno connection between a chosen pair of sender and receiver nodes and measured the throughput over the lifetime of the connection.
They use throughput as the performance metric in this paper. The TCP throughput is usually less than "optimal" due to the TCP sender's inability to accurately determine the cause of a packet loss. The TCP sender assumes that all packet losses are caused by congestion. Thus, when a link on a TCP route breaks, the TCP sender reacts as if congestion was the cause, reducing its congestion window and, in the instance of a timeout, backing-off its retransmission timeout (RTO). Therefore, route changes due to host mobility can have a detrimental impact on TCP performance. To gauge the impact of route changes on TCP performance, they derived an upper bound on TCP throughput, called the expected throughput. The TCP throughput measure obtained by simulation is then compared with the expected throughput.
They obtained the expected throughput as follows. We first simulated a static (fixed) network of n nodes that formed a linear chain containing n − 1 wireless hops. The nodes used the 802.11 MAC protocol for medium access. Then, a one-way TCP data transfer was performed between the two nodes at the ends of the linear chain, and the TCP throughput was measured between these nodes. This set of TCP throughput measurements is analogous to that performed using similar (but not identical) MAC protocols.
This paper, investigated the effects of mobility on TCP performance in mobile ad hoc networks. Through simulation, noted that TCP throughput drops significantly when node movement causes link failures, due to TCP's inability to recognize the difference between link failure and congestion. Then they made this point clearer by presenting several specific examples, one of which resulted in zero throughput, the other, in an unexpected rise in throughput with an increase
in speed. We also introduced a new metric, expected throughput, which provides a more accurate means of performance comparison by accounting for the differences in throughput when the number of hops varies. We then used this metric to show how the use of explicit link failure notification (ELFN) can significantly improve TCP performance, and gave a performance comparison of a variety of potential ELFN protocols. In the process, they discovered some surprising effects that route caching can have on TCP performance.
This work need intend to investigate ELFN protocols in more detail, as well as the effects that other mobile ad hoc routing protocols have on TCP performance. Also more research is needed to better understand the complex interactions between TCP and lower layer protocols when used over mobile ad hoc networks, and to find solutions to the problems caused by these interactions [5].

Martin Kohlwes et. al. reported the results from a long series of various measurements on the behavior of TCP over a UMTS wireless channel performed in two different UTMS networks. They conclude that at least in good conditions TCP throughput is close to theoretical maximum, and that RTT is fairly stable. Practically no packet losses were detected, and spurious retransmissions were extremely rare. No performance benefit was observed when TCP retransmission timer modifications, such as the Eifel and FRTO algorithms were used.
Two types of TCP connections were used in the measurements. First, connections between two UEs

were used. While this is an interesting case to study, most of the TCP traffic will probably be related to accessing services in the fixed network. Thus, the bulk of the measurements were performed in a scenario, where a dedicated Linux-based server, running multiple TCP-based services was contacted from a client running on a laptop connected to the UE. To study the behavior of the TCP state machine in more detail, small modifications to the Linux kernel were done, enabling the logging of various internal variables, such as the congestion window size, the threshold value, and so on. In addition to this data, TCP dump was used to obtain packet traces for further analysis during all measurements.
The control mechanism measurements were carried out UMTS offered a stable data channel for TCP to operate in. High packet loss rates and highly varying RTT commonly attributed to wireless channels were not observed. Thus, UMTS certainly has the potential to offer high-speed internet connectivity using standard protocols and settings. Whether this potential continues to be realized in commercial networks in the future is an interesting question. In the measurements, no other users were to our knowledge present in the network.
Due to the design of the UMTS system, these conclusions should also hold for networks used by not a single, but a small group of users. The commercial realities, on the other hand, make it unlikely that these expensive networks run with only a few users per cell. It is unclear what takes place when networks become truly crowded [6].

Conclusion and Future Works

The Transmission Control Protocol (TCP) was designed to provide reliable end-to-end delivery of data over unreliable networks. In practice, most TCP deployments have been carefully designed in the context of wired networks. Ignoring the properties of wireless ad-hoc Networks can lead to TCP implementations with poor performance. In order to adapt TCP to the ad hoc environment, improvements have been proposed in the literature to help TCP to differentiate between the different types of losses. Indeed, in mobile or static ad hoc networks losses are not always due to network congestion, as it is the case in wired networks or wireless networks. After survey the techniques on TCP we conclude that the causes of TCP performance degradation in MANETs are due to many major problems. These problems are TCP is unable to distinguish between losses due to route failures and network congestion; TCP suffers from frequent route failures, End to End delay, and throughput. During the survey, we also find some points that can be further explored in the future using advanced technique in feature extraction method and will improve the performance of TCP technique to achieve more efficient accuracy in network congestion, throughput and reduce the end to end delay time.

After surveying different techniques we define the Advantages and Disadvantages of techniques in the table:

Techniques	**Advantages/ Merits**	**Disadvantages /Future Improvement Direction**
TCP Congestion Control Algorithms, BIC, Cubic, Compound, Vegas, Reno, Westwood.	The performance of these congestion control algorithms in very ideal condition without any cross traffic and any additional flows.	This work could extend in select few algorithms from each of these four categories and will evaluate their performance in MANET scenario. There are other varying network parameters and metrics that the authors are working on the same. Based on the results, it needs to be extending the further enhancement towards specific application on MANETs [1].
TCP Congestion Control Algorithms, MANET, BIC, Vegas	The performance of these algorithms analyzed in ideal condition without any cross traffic and any other additional flows. In this small MANET scenario, the algorithm BIC provided good throughput after 75 seconds but algorithm Vegas provided stable and excellent result almost all over on the whole run time.	This work is not sufficient reliable on performance on TCP congestion control. It could be improving the work evaluation with based on the as slow start and Congestion Avoidance [2].
Spectrum Management, Transmission Control Protocol	It provides facilities to support cross-layer information sharing among network protocols at different layers of the protocol stack. Simulation results show that sensing time interval and type of PU activity play a critical role in deciding the TCP	The simulation results highlight that transport protocols proposed for traditional wireless ad hoc networks might not work well communication over Cognitive Radio Ad Hoc Networks. Since transport layer is still an explored research area for Cognitive Radio Ad Hoc Networks (CRAHNs), studying and

	performance.	understanding problems of classical TCPs in the cognitive environment is fundamental to design novel transport protocol solutions for Cognitive Radio Ad-hoc Networks [3].
Ad-hoc TCP algorithm	It presented a solution to the problem of running TCP in ad hoc wireless networks. The solution is to implement a thin layer between IP and ad-hoc TCP (called ATCP) that ensures correct TCP behavior while maintaining high throughput.	The Ad-hoc TCP's performance is need to improve as measured by the time to transfer large files. And also need to maintain end to end delay in ATCP congestion control behavior when there is network congestion [4].
Explicit feedback algorithm, TCP Reno	The use of explicit link failure notification (ELFN) can significantly improve TCP performance, and gave a performance comparison of a variety of potential ELFN protocols. In the process, they discovered some surprising effects that route caching can have on TCP performance.	This work need intend to investigate ELFN protocols in more detail, as well as the effects that other mobile ad hoc routing protocols have on TCP performance. Also more research is needed to better understand the complex interactions between TCP and lower layer protocols when used over mobile ad hoc networks, and to find solutions to the problems caused by these interactions [5].
TCP, FRTO algorithm	To study the behavior of the TCP state machine in more detail, small modifications to the Linux kernel were done, enabling the logging of various internal variables, such as the congestion window size, the threshold value, and so on. TCP throughput is close to theoretical maximum, and that RTT is fairly stable. Practically no packet losses were detected, and spurious retransmissions were extremely rare.	It is unclear what takes place when networks become truly crowded [6].

References

[1] M.Jehan, Dr.G.Radhamani, and T.Kalakumari, "VEGAS: Better Performance than Other TCP Congestion Control Algorithms on MANETs", International Journal of Computer Networks (IJCN), Volume (3): Issue (2): 2011, pp. 151-158.

[2] M.Jehan, G.Radhamani, and T.Kalakumari, "Experimental Evaluation of TCP BIC and Vegas in MANETs", International Journal of Computer Applications (0975 – 8887) Volume 16– No.1, February 2011, pp. 34-38.

[3] Marco Di Felice, Kaushik Roy Chowdhury, and Luciano Bononi, "Modeling and Performance Evaluation of Transmission Control Protocol over Cognitive Radio Ad Hoc Networks", MSWiM'09, October 26–29, 2009, Tenerife, Canary Islands, Spain.Copyright 2009 ACM 978-1-60558-616-8/09/10, pp. 4-9.

[4] Jian Liu, and Suresh Singh," ATCP: TCP for Mobile Ad Hoc Networks", IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 19, NO. 7, JULY 2001, 1300-1315.

[5] GAVIN HOLLAND, and NITIN VAIDYA, "Analysis of TCP Performance over Mobile Ad Hoc Networks", Wireless Networks 8, 275–288, 2002, Kluwer Academic Publishers. Manufactured in Netherlands, pp. 275-278.

[6] Martin Kohlwes and Janne Riihiarvi and Petri Mahonen, "Measurements of TCP Performance over UMTS Networks in Near-Ideal Conditions", IEEE 2005.

[7] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback based scheme for improving TCP performance in ad hoc wireless networks," in Proc. 18th Int. Conf. Distributed Computing Systems, Amsterdam, The Netherlands, May 26–29, 1998, pp. 474–479.

[8] S. Floyd, "TCP and explicit congestion notification," ACM Compute Communication. Rev., vol. 24, no. 5, Oct. 1994, pp. 10-23