ISSN: 2321-7782 (Online)
Volume 1, Issue 7, December 2013
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

A Survey: High Speed TCP Variants in Wireless Networks

Hrituparna Paul[1]
Assistant Professor
National Institute of Technology
Agartala - India

Priyanka Sarkar[2]
Assistant Professor
National Institute of Technology
Agartala - India

Abstract: ***Mobile Ad-Hoc Network is a self-directed group of mobile users that communicate using wireless links with no support from any pre-existing infrastructure network and used as a highly reliable end-to-end protocol for transporting applications. But wireless networks suffer from significant throughput degradation and delays. It uses Congestion Control and Avoidance algorithms which degrades end-to-end performance in wireless system. Through simulations we study the effects of Destination Sequenced Distance Vector (DSDV), Ad hoc On demand Distance Vector (AODV) and Dynamic Source Routing (DSR) routing protocols on TCP with TCP Vegas, Selective Acknowledgment (SACK) option, and Cubic TCP in this paper.***

Keywords: Multi-hop wireless networks, TCP, SACK, Vegas, cubic, DSDV, AODVand DSR.

I. INTRODUCTION

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

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

Transport Control Protocol /Internet Protocol (TCP/IP) is a connection oriented protocol of the transport layer. It provides features like flow control, reliability and congestion control. It has been very effective in data transmission delivery and have also developed variants to possess the possibility to increase performance and multiple packet loss recovery. Today, the TCP is extensively tuned to provide high-quality performance in the conventional wired network. In fact, the TCP is responsible for providing reliable data transport in the Internet. However, it cannot offer reliable service while using e-mail, internet search and file transmission in a MANET. This protocol is a standard networking protocol on the internet and is the most widely used transport protocol for data services like file transfer, e-mail and WWW browser. It is primarily designed for wire-line networks, faces performance degradation when applied to the ad hoc scenario. In addition, various routing protocols behave differently

over the variants of TCP. It is essential to understand the performance of different MANET routing protocols under TCP variants. In this paper, we have done a performance analysis of MANET Routing Protocols over different TCP Variants.

The paper is organized as follows. Section I provides Introduction. Section II describes the Standard TCP congestion control algorithms. Section III describes the various TCP variants. Section IV summarize the TCP variants Section V presents the simulation Set Up of our work, Section VI presents the results and finally Section VII gives the conclusion and future scope of our work which concludes the paper.

II. TCP CONGESTION CONTROL ALGORITHM

The four algorithms, Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery [4][5] are described below:

A. Slow Start [6]

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

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

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

B. Congestion Avoidance [6]

During data transfer phase of a TCP connection the Slow Start algorithm is used. There may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. When this happens, Congestion Avoidance is used to slow the transmission rate and Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow down and stay slow.

In the Congestion Avoidance [6] algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked.

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

C. Fast Retransmit and Fast Recovery [6]

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

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

III. TCP VARIANTS

A series of congestion collapses in internet is observed for the first time in October 1986 [5]. In 1988, Van Jacobson proposed three algorithms for congestion avoidance and control: Slow Start, Congestion Avoidance and Fast Retransmit. Later, a data recovery algorithm called Fast Recovery [7] was also proposed by Jacobson. These four algorithms are fundamental congestion control algorithms and are included in modern TCP implementations. However, above mentioned congestion control algorithms have also undergone several modifications to improve the performance of TCP on wired as well as wireless networks.

TCP Vegas:

TCP Vegas implementation tries to detect the incipient stages of congestion before packet losses occur. It uses proactive mechanisms to increase and decrease the size of cwnd. Other TCP variants assume packet loss as a sign of congestion in the network whereas TCP Vegas uses the difference in the expected RTT [8] and the actual RTT [8] to adjust the cwnd size [8]. Thus, the performance of TCP Vegas largely depends on the accuracy of RTT estimation.

A modified Slow Start algorithm is implemented in TCP Vegas to start the “Self-Clocking” mechanisms of TCP. It also has a new retransmission policy which retransmits the lost packet after receiving one (rather than three) duplicate acknowledgement if the estimated RTT is greater than the retransmission timeout value. Brakmo et al [8] provides a detailed description about the mechanisms of TCP Vegas.

The major drawback of TCP Vegas is that it lacks mechanisms that handle rerouting of connection [13]. Rerouting a path may change the RTT of the connection and may affect the accuracy of RTT estimation. If the new route has shorter RTT, cwnd size will be increased but it does not degrade the performance of TCP Vegas [13]. However, if the new route has longer RTT, TCP Vegas incorrectly assumes that the increase in RTT is due to congestion and thus reduces its cwnd size, resulting in substantial decrease in throughput [13].

TCP with Selective Acknowledgment (SACK) option:

Congestion control algorithms implemented in SACK TCP are extension of TCP Reno's congestion control algorithms. SACK TCP uses TCP Reno's congestion control algorithms to increase and decrease the size of cwnd. However, it makes minimal changes to other congestion control algorithms like Fast Recovery and Fast Retransmit. Thus it improves overall throughput of the network by avoiding unnecessary delays in retransmitting the lost packets.

When congestion is detected by the loss of a data packet, SACK TCP enters Fast Recovery phase as in TCP Reno. It retransmits the lost packet and reduces the cwnd by half. To estimate the number of outstanding packets in the network, SACK TCP uses a variable called pipe. A new packet is transmitted by SACK TCP only if the value of pipe is less than the value of cwnd

CUBIC TCP:

CUBIC TCP is an enhanced version of Binary Increase Congestion Control (BIC) TCP. BIC TCP, proposed in [9], focuses on solving the RTT unfairness problem. It combines two algorithms called additive increase and binary search increase. Additive increase ensures linear RTT fairness when cwnd is large and binary search increase ensures TCP-friendliness when cwnd is small. Binary search increase algorithm is described in detail in [9].

CUBIC further improves the performance of BIC TCP with respect to RTT unfairness problem by incrementing cwnd independent of RTT [10]. During steady state, CUBIC increases cwnd size aggressively if it is far from equilibrium and slowly when it is close to equilibrium [10]. However, TCP-unfairness problem is not addressed by CUBIC TCP.

IV. SUMMARY OF TCP VARIANTS[1]

Table 1 Comparison between different TCP Variants

TCP Congestion Control Variants	Supported Features	Problems
TCP Vegas:	• Modified slow-start • New retransmission • Congestion avoidance	• Vegas may not stabilize if buffers are small
SACK	• Slow start algorithm • Congestion avoidance • Fast retransmit • SACK options at the receiver	• Current receivers are unable to support the SACK options
CUBIC TCP:	• Slow start algorithm • Fast Retransmit and Fast Recovery • Builds on BIC by simplifying the window response function and improving its TCP friendliness and RTT fairness through RTT independent congestion window updates.	• Suffers from long convergence times between competing CUBIC flows.

V. SIMULATION SETUP AND METHODOLOGY

The results in this paper are based on the simulations done on ns-2.

Mobile topologies:

In mobile topologies we designed a network model consisting of 30 nodes in a 1500 x 300 meter flat, rectangular area. Our network model is analogous to the one in [3]. The mobility patterns are generated using the mobility pattern generator provided in ns-2 model. The mean speed with which nodes move is 10 m/s. We generate 25 such mobility patterns and our simulation results are based on the average throughput of 25 mobility patterns. Other parameters are same as mentioned above for static topologies. Simulation results are discussed in section V.

Performance Metric:

The performance metric used in our study is throughput. In mobile topologies the distance between the source and destination keeps varying. The number of hops on the path from source to destination may increase or decrease. Hence, we use another performance metric called *expected throughput* as defined in [3]. It is calculated as follows:

Let T_i denote the throughput obtained for the string topology, where I denotes the number of hops and $1 \leq i \leq \infty$. When $i = \infty$ it means that the network is partitioned and hence throughput $T_\infty = 0$. Let t_i be the duration for which the distance between source and destination in mobile topology is i hops ($1 \leq i \leq \infty$). The expected throughput is then calculated as:

$$\frac{\sum_{i=1}^{\infty}(t_i \times T_i)}{\sum_{i=1}^{\infty}(t_i)}$$

Actual throughput is the throughput measure obtained by simulations. The obtained actual throughput is then compared with the expected throughput.

VI. RESULTS AND ANALYSIS

Table I to Table III shows the expected throughput, actual throughput and their percentage of expected throughput which have been achieved for CUBIC, SACK and Vegas with DSDV, AODV and DSR respectively.

TABLE I. THROUGHPUT (IN KBPS) USING DSDV

TCP Variant	Expected Throughput	Actual Throughput	Percentage Achieved
Vegas	235.192	87.492	37.20
SACK	344.105	156.251	45.40
CUBIC	1391.354	646.1158	46.43

TABLE II. THROUGHPUT (IN KBPS) USING AODV

TCP Variant	Expected Throughput	Actual Throughput	Percentage Achieved
Vegas	295.105	254.136	86.11
SACK	334.781	273.985	81.84
CUBIC	1368.019	1171.209	85.61

TABLE III. THROUGHPUT (IN KBPS) USING DSR

TCP Variant	Expected Throughput	Actual Throughput	Percentage Achieved
Vegas	234.296	173.712	74.14
SACK	295.105	254.136	86.11
CUBIC	1343.527	1271.254	94.62

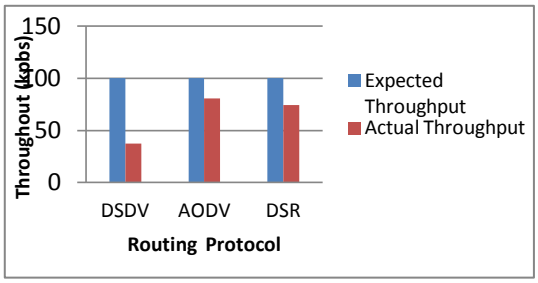

Figure 1. Throughput (in Kbps) using TCP Vegas

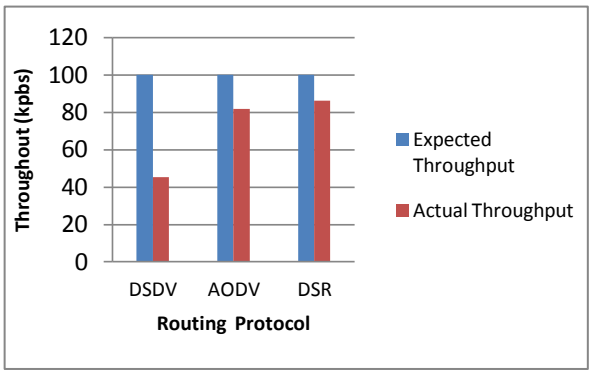

Figure 2. Throughput (in Kbps) using SACK TCP

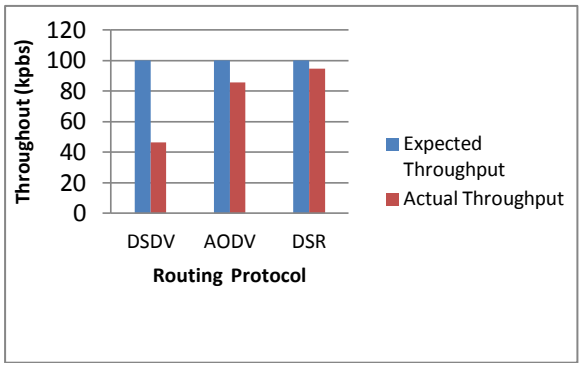

Figure 3. Throughput (in Kbps) using CUBIC TCP

We denote the expected throughput values as 100 and actual throughput values accordingly for the TCP variants as shown in Figure 1, 2 and 3.

It has been observed that the highest throughput obtained by almost all TCP variants mentioned here is with DSR. The actual throughput of DSR is almost similar to the expected throughput. The least throughput is obtained by the variants of TCP is with DSDV.

VII. Conclusion and Further Work

Through simulations we have observed the behavior of Destination Sequenced Distance Vector (DSDV), Ad hoc on demand Distance Vector (AODV) and Dynamic Source Routing (DSR) routing protocols on TCP Vegas, CUBIC and SACK in mobile multi-hop wireless networks.

Here we have not considered the effects of non-congestion losses on the performance of high-speed TCP variants. We have not taken into account the Convergence speed, RTT fairness and TCP fairness. As future work, we intend to study the also the effects of non-congestion losses and the performance of the TCP variants with the above mentioned parameters.

References

1. K. Satyanarayan Reddy, Lokanatha C. Reddy," A Survey On Congestion Control ProtocolsFor High Speed Networks", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.7, July 2008.

2. D. Johnson, D. Maltz and Y. Hu., "The dynamic source routing protocol for mobile ad hoc networks", IETF MANET Working Group, Internet Draft, 2003.
3. M.K.J. Kumar and R.S. Rajesh, "Performance analysis of MANET routing protocols in different mobility models", IJCSNS International Journal of Computer Science and Network Security, vol. 9 No.2, pp 22-29, Feb 2009
4. K.Kathiravan, Dr. S. Thamarai Selvi, A.Selvam "Tcp Performance Analysis For Mobile Ad Hoc Network Using Ondemand Routing Protocols.
5. JACOBSON, V. Congestion avoidance and control. In Proceedings of SIGCOMM '88 (Stanford, CA, Aug. 1988), ACM.
6. Suhas Waghmare et. al "Comparative Analysis of different TCP variants in a wireless environment", 978-1-4244-8679-3/11 ©2011 IEEE
7. S. Floyd, T. Henderson, "The Newreno Modification to TCP's Fast Recovery Algorithm," Request For Comments 2582, Experimental, (April 1999).
8. L. Brakmo and L. Peterson, "TCP Vegas: End-to-end congestion avoidance on a global internet," IEEE Journal on Selected Areas in Communication, vol. 13, 1465 – 1480, (Oct. 1995).
9. I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," Proceedings of the third PFLDNet Workshop, France, 2005.
10. Komal Zaman1 Muddesar Iqbal1 Muhammad Shafiq1, Azeem Irshad2, Saqib Rasool2," A Survey: variants of TCP in Ad-hoc networks", Advances in Computer Science: an International Journal, Vol. 2, Issue 5, No.6 , November 2013 ISSN : 2322-5157

