To Study TCP Variants in Mobile Ad-hoc Network

Dr. Madhu Goel[1] Ms. Divya Garg[2]
[1]H.O.D. in CSE Deptt.[2]P.G Scholar
[1, 2]Kurukshetra Institute of technology & Management
[1, 2]Kurukshetra University, Kurukshetra

Abstract—*The main goal of this paper is to study the different congestion control and avoidance mechanisms that have been proposed for TCP/IP protocols, namely: TCP Tahoe, TCP Reno, TCP New-Reno, TCP Vegas, SACK and TCP Westwood. All these implementations suggest the mechanisms for deciding when a segment should be re-transmitted and how the sender should behave when it encounters the congestion. MANET is the wireless network with no central authority and is highly dynamic in nature. So, it suffers from significant throughput degradation. To overcome this problem various TCP Variants have been proposed. So, in this paper we will study various TCP Variants.*

Keywords: -TCP Tahoe, TCP Reno,TCP New Reno, TCP Vegas, SACK, TCP Westwood.

I. Introduction

Mobile ad-hoc network is an infrastructure-less, dynamic network. Mobile ad-hoc network is a collection of wireless mobile nodes that can communicate with each other without help of any centralized authority. To provide end-to-end communication throughout the network, nodes cooperate with each other to handle network functions, such as packet routing. These networks are fully distributed and can work at any place without the help of any fixed infrastructure as access points or base stations. Figure 1 shows a simple ad-hoc network with 3 nodes. Node 1 and 3 are not within range of each other so node 2 can be used to forward packets between node 1 and 3. Node 2 will act as a router. All three nodes together form an ad-hoc network.

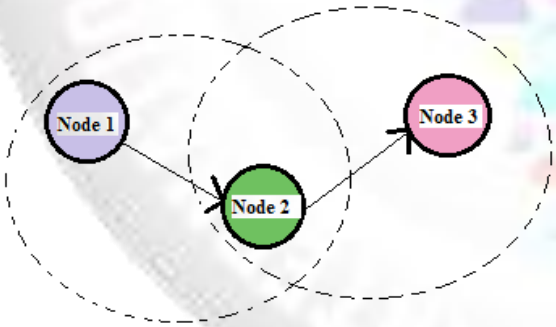

Fig. 1: Example of MANET

A. TCP in MANET

A wireless local-area network (LAN) uses radio waves to connect devices such as laptops to the internet and to your business network and its applications. An Ad hoc network is a collection of mobile nodes and wireless communication network is used to connect these mobile nodes. TCP (Transmission Control Protocol) was designed to provide reliable end to end delivery of data over unreliable network. Traditionally TCP assumes that all the packet losses are due to congestion. Most TCP deployment has been carefully designed in the context of wired networks. Ignoring the properties of wireless Ad hoc networks can lead to the implementations with poor performance. In order to adapt TCP to wireless network, improvement has been proposed in the literature to detect different types of losses. Indeed, in mobile or static Ad hoc networks losses are not always due to network congestion, as it is in wired networks. So, our main aim is to study the various TCP Variants that were designed to overcome these problems.

II. TCP Congestion Control Algorithm

The four phases are: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery which are described as follows:

A. Slow Start

Slow Start is a mechanism that is used by the sender to control the transmission rate, and is also known as sender based flow control. The rate of acknowledgements returned by the receiver helps to determine the rate at which the sender can transmit data. When a TCP connection established, the Slow Start algorithm sets the size of congestion window to one segment, which is the maximum segment size (MSS) initialized by the receiver during the connection establishment phase. As the acknowledgements are returned by the receiver, the size of congestion window increases by one segment for each acknowledgement returned. At some point the congestion window may become too large that reaches to the ssthresh, a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. At this point, Congestion Avoidance is used to slow the transmission rate and also 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.

B. Congestion Avoidance

In the Congestion Avoidance phase, timer expiring or the reception of three duplicate ACKs can implicitly signal the sender that the network is congested. The sender immediately sets its window to one half of the current window size. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start phase

C. Fast Retransmit and Fast Recovery

TCP sets a timer each time whenever a data segment is transmitted, and thus it ensures the reliability. TCP retransmits the packet when it does not obtain any acknowledgement within the fixed time-out interval. The sender implements the fast retransmit algorithm for identifying and also repairing the loss. This fast retransmit phase is used based on the incoming duplicate ACKs if there are at least three duplicate ACK's it can be assumed that a data 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 data 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. 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.

III. TCP VARIANTS

A. TCP Tahoe

The first version of TCP is known as TCP Tahoe. When a connection is initialized, TCP implements a mechanism called slow start to increase the congestion window. Initially the rate is low. And it increases rapidly and gradually. For every acknowledged packet, the congestion window increases by one MSS. When the congestion window exceeds threshold "ssthresh", the algorithm enters a new state, called congestion avoidance. This threshold is updated at the end of each slow start.

B. TCP Reno

TCP Reno is an improvement over Tahoe. The Reno version of TCP introduces another phase called fast recovery phase. If three duplicate ACKs are received, Reno will perform a fast retransmit, and enter a state called fast recovery where it will halve the congestion window and retransmits the lost packet that was signaled by three duplicate acknowledgements, and waits for an acknowledgment of the entire transmit window. If there is no acknowledgment, if an ACK times out, TCP Reno experiences a timeout and enters the slow start phase, as Tahoe.

C. TCP Vegas

TCP Vegas provides a TCP congestion avoidance algorithm that uses packet delay rather than packet loss. The Vegas congestion detection algorithm differs from earlier TCP Tahoe and Reno where congestion is detected by packet drops only after it has actually happened. TCP Vegas can identify the queuing delay and based on this, it will adjust the congestion window size. The difference between expected traffic and actual traffic is used to adjust the size of the congestion window. Both the increase and decrease of the rate is additive. But in TCP Reno, Congestion window keep increasing until a packet is lost, and therefore they will always face packet loss at some point or other.

Vegas give 40 to 70% better throughput than TCP Reno with less than half the packet loss. In addition to the modified congestion avoidance mechanism, the TCP Vegas also uses the retransmission mechanism to avoid timeout. If the sender is unable to receive 3 duplicate ACKs (due to lose segments or window size is too small), in such a case, the sender can do retransmission after one dup ACK is received, if RTT estimate > timeout. The slow start phase is modified so that the sender tries to find the correct window size without causing a loss. The Vegas algorithm heavily depends on accurate calculation of the base RTT value.

D. TCP New Reno

TCP New Reno is a slight modification over TCP-Reno. It is able to detect multiple packet losses and thus it is much better than Reno in the event of multiple packet losses. Like Reno, New-Reno also enters into fast-retransmit phase when it receives multiple duplicate packets, however it differs from Reno in that it doesn't exit fast-recovery until all the data which was out standing at the time it entered fast recovery is acknowledged. Thus it overcomes the problem faced by Reno of reducing the congestion window multiples times. The fast recovery phase proceeds as in Reno, however when a fresh ACK is received then there are two cases:

- If it ACK's all the segments which were outstanding when we entered fast recovery then it exits fast recovery and sets CWD to threshold value and continues congestion avoidance.
- If the ACK is a partial ACK then it deduces that the next segment in line was lost and it re-transmits that segment and sets the number of duplicate ACKS received to zero. It exits Fast recovery when all the data in the window is acknowledged.

E. SACK

The client sends request to the server, and the server gives a response that is broken into four TCP segments. The server transmits all four packets in response to the request but the second response packet is dropped somewhere in the network and never reaches the host. Let's discuss what happens:

Enter Selective Acknowledgments

SACK works as follows: It allows the client to say "I have not received data 2, but I have received data segment 3 and 4".

- Step 1

Response segment 2 is lost.

- Step 2

The client realizes that a segment is missing between segments 1 and 3. It sends a duplicate acknowledgment for segment 1, and also uses a SACK option indicating that it has received segment 3.

- Step 3

The client receives data segment 4 and then sends another duplicate acknowledgment for segment 1, but this time SACK option used to show that it has received segments 3 through 4.

- Step 4

The server receives the client's duplicate ACK for segment 1 and SACK for segment 3. By this, the server can understand that the client has not received data segment 2, so segment 2 is retransmitted again. The next Selective Acknowledgement (SACK) received by the server indicates that the client has received segment 4 successfully.

- Step 5

The client receives segment 2 and then sends an acknowledgment to indicate that client has received all data up to an including segment 4.

F. TCP Westwood

Westwood TCP is a new congestion control algorithm that is based on end-to-end bandwidth estimate. In particular, TCP

Westwood estimates the available bandwidth by counting and filtering the flow of returning ACKs and adaptively sets the *cwnd*and the *ssthresh*after congestion by taking into account the estimated bandwidth.

In particular, when three DUPACKs are received, both the congestion window (*cwnd*) and the slow start threshold (*ssthresh*) are set equal to the estimated bandwidth (*BWE*) times the minimum measured round trip time (*RTTmin*); when a coarse timeout expires the *ssthresh*is set as before while the *cwnd*is set equal to one.

The pseudo code of the Westwood algorithm is reported below:

1) On ACK reception:

cwnd is increased accordingly to the Reno algorithm;
the end-to-end bandwidth estimate BWE is computed;

2) When 3 DUPACKs are received:

ssthresh =max(2, (BWE* RTTmin) / seg_size);
cwnd = ssthresh;

3) When coarse timeout expires:

ssthresh = max(2,(BWE* RTTmin) / seg_size);
cwnd = 1;

IV. Conclusion

The purpose of this paper was to study the various TCP Variants such as TCP Tahoe, TCP Reno, TCP New Reno, TCP Vegas, SACK and TCP Westwood which are different congestion control and avoidance mechanisms. All the variants perform differently under different conditions. For the future prospective these variants can be enhanced or some new variants can be introduced.

References

[1] Aarti and Dr. S. S. Tyagi, "Study of MANET: Characteristics, Challenges, Application and Security Attacks", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013.

[2] C. Siva Ram Murthy and Manoj, B.S. Second Edition, Low price Edition, Pearson Education, 2007.Adhoc Wireless Networks, Architectures and Protocols.

[3] Dong kyun Kim, Juan-Carlos Cano and P. Manzoni, C-K. Toh," A comparison of the performance of TCP-Reno and TCP-Vegas over MANETs", 1-4244-0398-7/06/$20.00 ©2006 IEEE

[4] Foezahmed, Sateesh Kumar Pradhan, Nayeema Islam, and Sumon Kumar Debnath**,"** Evaluation of TCP over Mobile Ad-hoc Networks" , *(*IJCSIS) International Journal of Computer Science and Information Security,Vol. 7, No. 1, 2010.

[5] Xiang Chen, HongqiangZhai, Jianfeng Wang, and Yuguang Fang ,"TCP performance over mobile ad hoc networks",CAN. J. ELECT. COMPUT. ENG., VOL. 29, NO. 1/2, JANUARY/APRIL 2004

[6] GAVIN HOLLAND NITIN VAIDYA,"Analysis of TCP Performance over Mobile Ad Hoc Networks", Wireless Networks 8, 275–288, 2002 ⁄2002 Kluwer Academic Publishers. Manufactured in the Netherlands

[7] Harpreet Singh Chawla, M. I. H. Ansari, Ashish Kumar, Prashant Singh Yadav," A Survey of TCP over Mobile ADHOC Networks", International Journal of Scientific & Technology Research Volume 1, Issue 4, May 2012 ISSN 2277-8616

[8] BogdanMoraru Flavius Copaciu Gabriel Lazar Virgil Dobrota,"Practical Analysis of Implementations: Tahoe, Reno, NewReno"

[9] Ashish Ahuja, Sulabh Agarwal, Jatinder Pal Singh, Rajeev Shorey," Performance of TCP over Different Routing Protocols in Mobile Ad-Hoc Networks,"0-7803-571 8-3/00/$10.00 *02000* IEEE.

[10] LaxmiSubedi, MohamadrezaNajiminaini, and LjiljanaTrajković," Performance Evaluation of TCP Tahoe, Reno, Reno with SACK, and NewReno Using OPNET Modeler".