2011 International Conference on Computer Communication and Management
Proc .of CSIT vol.5 (2011) © (2011) IACSIT Press, Singapore

TCP over Multi-Hop Wireless Mesh Network

Sumedha Chokhandre [1], Urmila Shrawankar [2]

[1] G. H. Raisoni College of Engineering, Nagpur, India.

[2] G. H. Raisoni College of Engineering, Nagpur, India.

Abstract. The Transmission Control Protocol (TCP) was designed to provide reliable end-to-end delivery of data over unreliable networks. Traditionally TCP was designed and optimized only for wired networks. Ignoring the properties of wireless Networks, TCP can be implemented with poor performance. In order to adapt TCP to wireless environment, improvements have been proposed in the literature to help TCP to differentiate between the different types of losses. Indeed, in wireless networks losses are not always due to network congestion, as it is mostly the case in wired networks. Here we present an overview of this issue and a detailed discussion of the major factors involved. In addition, we survey the main proposals which aim at increasing the TCP performance in the wireless environments.

Keywords: Wireless Mesh Network, TCP Performance, Packet Loss, Congestion.

1. Introduction

Wireless mesh networks (WMN) are rapidly emerging as a promising complement to existing broadband access infrastructures, because they extend wireless local area network (WLAN) access beyond traditional hotspot areas to enhance coverage and provide seamless mobility. In WMNs, all communications between mesh network nodes are over radio links. Applications of WMNs include backhaul for broadband access networks, metropolitan area mobile networks, and citywide surveillance systems.

TCP (Transmission Control Protocol) was designed to provide reliable end-to-end delivery of data over unreliable networks. In theory, TCP should be independent of the technology of the underlying infrastructure. In particular, TCP should not care whether the Internet Protocol (IP) is running over wired or wireless connections. In practice, it does matter because most TCP deployments have been carefully designed based on assumptions that are specific to wired networks. Ignoring the properties of wireless transmission can lead to TCP implementations with poor performance. In wireless networks, the principal problem of TCP lies in performing congestion control in case of losses that are not induced by network congestion. Since bit error rates are very low in wired networks, nearly all TCP versions now a day's assume that packets losses are due to congestion. Consequently, when a packet is detected to be lost, either by timeout or by multiple duplicated ACKs, TCP slows down the sending rate by adjusting its congestion window. Unfortunately, wireless networks suffer from several types of losses that are not related to congestion, making TCP not adapted to this environment. Numerous enhancements and optimizations have been proposed over the last few years to improve TCP performance over one-hop wireless (not necessarily Wireless) networks.

The rest of the paper is organized as follows: In Section 2 we describe the Traditional TCPs. Section 3 describe the related work for TCP Performance in wireless environment and Section 4 point out the Proposed Work and Section 5 & 6 includes conclusion and the references.

2. Traditional TCPs

2.1. TCP TAHOE

Tahoe refers to the TCP congestion control algorithm which was suggested by Van Jacobson. TCP is based on a principle of 'conservation of packets', i.e. if the connection is running at the available bandwidth capacity then a packet is not injected into the network unless a packet is taken out as well. TCP implements

this principle by using the acknowledgements to clock outgoing packets because an acknowledgement means that a packet was taken off the wire by the receiver. It also maintains a congestion window CWD to reflect the network capacity. However there are certain issues, which need to be resolved to ensure this equilibrium.

- Determination of the available bandwidth.
- Ensuring that equilibrium is maintained.
- How to react to congestion.

Slow Start:

TCP packet transmissions are clocked by the incoming acknowledgements. However there is a problem when a connection first starts up cause to have acknowledgements you need to have data in the network and to put data in the network you need acknowledgements. To get around this circularity Tahoe suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called 'slow-start'. The reason for this procedure is that an initial burst might overwhelm the network and the connection might never get started. Slow starts suggest that the sender set the congestion window to 1 and then for each ACK received it increase the CWD by 1. So in the first round trip time (RTT) we send 1 packet, in the second we send 2 and in the third we send 4. Thus we increase exponentially until we lose a packet which is a sign of congestion. When we encounter congestion we decreases our sending rate and we reduce congestion window to one and start over again.

Congestion Avoidance:

For congestion avoidance Tahoe uses 'Additive Increase Multiplicative Decrease'. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as a threshold value. It then set CWD to one and starts slow start until it reaches the threshold value. After that it increments linearly until it encounters a packet loss. Thus it increase it window slowly as it approaches the bandwidth capacity.

The problem with Tahoe is that it takes a complete timeout interval to detect a packet loss and in fact, in most implementations it takes even longer because of the coarse grain timeout. Also since it doesn't send immediate ACK's, it sends cumulative acknowledgements, therefore it follows a 'go back n 'approach. Thus every time a packet is lost it waits for a timeout and the pipeline is emptied. This offers a major cost in high band-width delay product links.

2.2. TCP RENO

This Reno retains the basic principle of Tahoe, such as slow starts and the coarse grain re-transmit timer. However it adds some intelligence over it so that lost packets are detected earlier and the pipeline is not emptied every time a packet is lost.

Reno requires that we receive immediate acknowledgement whenever a segment is received. The logic behind this is that whenever we receive a duplicate acknowledgment, then his duplicate acknowledgment could have been received if the next segment in sequence expected, has been delayed in the network and the segments reached there out of order or else that the packet is lost. If we receive a number of duplicate acknowledgements then that means that sufficient time have passed and even if the segment had taken a longer path, it should have gotten to the receiver by now. There is a very high probability that it was lost. So Reno suggests an algorithm called **'Fast Re- Transmit'.** Whenever we receive 3 duplicate ACK's we take it as a sign that the segment was lost, so we re-transmit the segment without waiting for timeout. Thus we manage to re-transmit the segment with the pipe almost full. Another modification that RENO makes is in that after a packet loss, it does not reduce the congestion window to 1. Since this empties the pipe. It enters into an algorithm which we call **'Fast-Re-Transmit'**.

Reno performs very well over TCP when the packet losses are small. But when we have multiple packet losses in one window then RENO doesn't perform too well and its performance is almost the same as Tahoe under conditions of high packet loss. The reason is that it can only detect a single packet loss. If there is multiple packet drops then the first info about the packet loss comes when we receive the duplicate ACK's. But the information about the second packet which was lost will come only after the ACK for the retransmitted first segment reaches the sender after one RTT.

2.3. NEW-RENO

New RENO is a slight modification over TCP-RENO. It is able to detect multiple packet losses and thus is much more efficient that RENO in the event of multiple packet losses.

Like Reno, New-Reno also enters into fast-retransmit 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 CWD multiples times.

The fast-transmit phase is the same as in Reno. The difference in the fast recovery phase which allows for multiple re-transmissions in new-Reno. Whenever new-Reno enters fast recovery it notes the maximums segment which is outstanding.

New-Reno suffers from the fact that it's takes one RTT to detect each packet loss. When the ACK for the first retransmitted segment is received only then can we deduce which other segment was lost.

2.4. SACK

TCP with 'Selective Acknowledgments' is an extension of TCP Reno and it works around the problems face by TCP RENO and TCP New-Reno, namely detection of multiple lost packets, and re-transmission of more than one lost packet per RTT.

SACKS retain the slow-start and fast retransmit parts of RENO. SACKS retain the slow-start and fast retransmit parts of RENO. It also has the coarse grained timeout of Tahoe to fall back on, in case a packet loss is not detected by the modified algorithm. SACK TCP requires that segments not be acknowledged cumulatively but should be acknowledged selectively. Thus each ACK has a block which describes which segments are being acknowledged. Thus the sender has a picture of which segments have been acknowledged and which are still outstanding. Whenever the sender enters fast recovery, it initializes a variable pipe which is an estimate of how much data is outstanding in the network, and it also set CWND to half the current size.

Every time it receives an ACK it reduces the pipe by 1 and every time it retransmits a segment it increments it by 1. Whenever the pipe goes smaller than the CWD window it checks which segments are not received and send them. If there are no such segments outstanding then it sends a new packet . Thus more than one lost segment can be sent in one RTT.

The biggest problem with SACK is that currently selective acknowledgements are not provided by the receiver. To implement SACK we'll need to implement selective acknowledgment which is not a very easy task.

2.5. VEGAS

Vegas are a TCP implementation which is a modification of Reno. It builds on the fact that proactive measure to encounter congestion is much more efficient than reactive ones. It tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggests a modified slow start algorithm which prevents it from congesting the network. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packet losses occur. However it still retains the other mechanism of Reno and Tahoe, and a packet loss can still be detected by the coarse grain timeout of the other mechanisms fail. The three major changes induced by Vegas are:

- New Re-Transmission Mechanism
- Congestion Avoidance
- Modified Slow-start

New Re-Transmission Mechanism:

Vegas extend on the re-transmission mechanism of Reno. It keeps track of when each segment was sent and it also calculates an estimate of the RTT by keeping track of how long it takes for the acknowledgment to get back. Whenever a duplicate acknowledgement is received it checks to see if the (current time segment transmission time)> RTT estimate; if it is then it immediately retransmits the segment without waiting for 3

duplicate acknowledgements or a coarse timeout. Thus it gets around the problem faced by Reno of not being able to detect lost packets when it had a small window and it didn't receive enough duplicate Ack's.

To catch any other segments that may have been lost prior to the retransmission, when a non duplicate acknowledgment is received, if it is the first or second one after a fresh acknowledgement then it again checks the timeout values and if the segment time since it was sent exceeds the timeout value then it re-transmits the segment without waiting for a duplicate acknowledgment. Thus in this way Vegas can detect multiple packet losses.

Also it only reduces its window if the re-transmitted segment was sent after the last decrease. Thus it also overcome Reno's shortcoming of reducing the congestion window multiple time when multiple packets are lost.

Congestion Avoidance:

TCP Vegas is different from all the other implementation in its behavior during congestion avoidance. It does not use the loss of segment to signal that there is congestion. It determines congestion by a decrease in sending rate as compared to the expected rate, as result of large queues building up in the routers. Thus whenever the calculated rate is too far away from the expected rate it increases transmissions to make use of the available bandwidth, whenever the calculated rate comes too close to the expected value it decreases its transmission to prevent over saturating the bandwidth. Thus Vegas combats congestion quite effectively and doesn't waste bandwidth by transmitting at too high a data rate and creating congestion and then cutting back, which the other algorithms do.

Modified Slow-start:

TCP Vegas differs from the other algorithms during its slow-start phase. The reason for this modification is that when a connection first starts it has no idea of the available bandwidth and it is possible that during exponential increase it over shoots the bandwidth by a big amount and thus induces congestion. To this end Vegas increases exponentially only every other RTT, between that it calculates the actual sending through put to the expected and when the difference goes above a certain threshold it exits slow start and enters the congestion avoidance phase.

3. Related Work

Several efforts for improving the performance of TCP in multi-hop wireless networks have recently been reported. The problem of TCP performance and degradation over multi-hop wireless network due to the conflict between data packets and acknowledgments are identified in [1].Also because of the impact of the MAC protocol on performance of TCP in multi-hop networks. The TCP throughput decreases exponentially as the number of hop increases due to hidden terminal problem, which increases the packet collision. Similar problems were evaluated in [2] where the authors show that using smaller values for both packet size and maximum window size in TCP setup can mitigate such problems to some extent. Several implementations of TCP are analyzed in [3] and they discovered that the throughput of TCP depends on the number of hops in the path as well as the performance of underlying routing protocol.

An end to-end combination scheme is evaluated in [4] to improve TCP throughput over multi-hop wireless network. In literature, different approaches focused on improving the throughput of TCP over multi-hop wireless networks have been described. Although much research has been done on improving TCP performance, only few approaches have been proposed for improving TCP performance in wireless mesh networks. A multi-channel assignment algorithm is proposed in [5] for constructing a wireless mesh network which eliminates the hidden terminal problem and thereby improving the performance of TCP in wireless multi-hop networks. A pacing scheme is proposed in [6] at the IP layer to improve the fairness in hybrid wireless/ wired networks. Gambiroza, Sadeghi and Edward [7][8][9] studied TCP performance in wireless mesh networks and they proposed a distributed link layer algorithm for achieving fairness among TCP flows. These approaches do not have a mechanism to avoid retransmission timeout caused by retransmission loss. Our proposed algorithm constitutes a modification at the transport layer in sender side rather than a

modification at the link layer for improving the performance of TCP by avoiding the frequent retransmission timeout in multi-hop wireless mesh networks.

4. Proposed Work

To overcome the throughput degradation of TCP in multihop wireless mesh networks, we modified the fast retransmit and recovery algorithms of TCP NewReno. The key idea of our algorithm is to calculate the outstanding packets and set the value of slow start threshold (ssthresh) based on half of the difference between maximum data packets sent and the last acknowledgment received at the TCP sender.

We used this difference for minimizing the frequent retransmission timeouts caused by retransmission loss. The value of ssthresh is used to determine whether the TCP should do Slow Start or Congestion Avoidance. TCP SAC adopts Slow Start and Congestion Avoidance of TCP NewReno and modifies Fast Retransmit and Recovery algorithms.

5. Conclusion

TCP assumes any packet loss as an indication of congestion, and provides two methods to detect packet loss: fast retransmits for light congestion, and retransmission timeouts (RTO) for heavy congestion. If TCP receives three successive duplicate acknowledgements, it halves its congestion window size assuming a packet is lost due to light congestion. If a TCP sender does not receive a new ACK before the retransmission timeout expires, it initializes its congestion window size to one segment assuming a packet is lost due to heavy congestion.

In this paper we first survey about the main challenges faced by the TCP in Wireless Mesh Network. All this challenges are sometime the reasons for the packet losses in the network. There are many traditional algorithms which are used for congestion avoidance in the network. To avoid the Performance degradation of TCP in the wireless network the new algorithm is proposed.

6. References

[1] Abhinav Gupta, Ian Wormsbecker and Carey Williamson, “Eperimental Evaluation of TCP Performance in Multi-hop Wireless Ad Hoc Networks”, Proc. IEEE MASCOTS, 2004.

[2] Mario Gerla, Ken tang and Rajive Bagrodia, “TCP Performance in Wireless Multi-hop Networks”, Proc.IEEE WMCSA, 1999.

[3] E.Altman and T.Jimenezl,”Novel Delayed Ack Techniques for improving TCP performance in Multi-Hop Wireless Networks”, Proc. of Personal Wireless Communication, 2003.

[4] S. Xu and T. Saadawi. ,”Performance evaluation of TCP algorithms in multi-hop wireless packet networks” Wireless Communications and Mobile Computing, 2001.

[5] Michelle Berger,”A performance Comparison of TCP protocols over Mobile Ad Hoc Networks”,Proc. IEEE CERMA,2006.

[6] Wei Ren and Hai Jini. ,”A Combination Scheme to Improve TCP Throughput over Multihops Wireless Mobile Ad Hoc Networks”,LNCS, PP.805-809,2006.

[7] Li-Ping Tung and Wei-Kuan Shih,”TCP Throughput Enhancement over Wireless Mesh Network”, IEEE Communication magazine, 2007.

[8] L. Yang, K.G.Seah and Q.Yin., “Improving fairness among TCP flows crossing Wireless Ad Hoc and Wired Networks”, Proc.ACM MobiHoc, 2003.

[9] V. Gambiroza, B. Sadeghi and Edward, “End-to-End Performance and fairness in multihop wireless backhaul networks”, Proc. ACM MOBICOM, 2004.