ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (17-23), Month: July - September 2014, Available at: www.researchpublish.com

Review report of Congestion control for wireless network

[1]Sonia, [2]Rupali Malhotra

[1]M.Tech Student Satpriya college of Engineering, MDU
Haryana (India)
[2]Asst.Prof, Satpriya Group of Engineering & Technology
Rohtak, Haryana (India)

Abstract: **Transmission Control Protocol (TCP) is considered one of the most important protocol in the internet. An important mechanism in tcp is the congestion control mechanism which controls tcp sending rate & makes tcp react to cngesfion signals. Now a days,tcp may work in n/w with some links that have lossy nature(ex. Wireless network). Tcp treats all packet losses if they due to congestion. Tcp reduces sending rate aggressively when there are transmission errors in an uncongested network. In this paper I present different solutions to overcome the performance degradation problem tcp faces when working over lossy links. Many solutions have been proposed but I will concentrate on end to end solutions that require no help from the intermediate network**.

Keywords: **TCP, packet control, wireless networks, packet delay, packet delay variation.**

1. INTRODUCTION

The performance of Transmission Control Protocol (TCP) has greatly improved since 1988, when the congestion avoidance and control algorithms were first introduced. TCP is currently the most widely used Internet transport protocol. In 2002, TCP traffic accounted for 95% of the IP network traffic. This was due to a variety of popular Internet applications and protocols. Web (HTTP), file transfer (FTP), and e-mail (SMTP) rely on TCP as the underlying transport protocol. Internet applications that rely on TCP today are likely to do so in the future. With a growing deployment of wireless networks, it is important to support these applications in both wireline and wireless environments. Hence, wireless networks will also require good TCP performance.

Wireless networks have different characteristics compared to wireline networks. TCP, which was carefully designed and tuned to perform well in wireline networks, suffers performance degradation when deployed in wireless networks.

2. TRANSMISSION CONTROL PROTOCOL

TCP is a connection-oriented transport layer protocol. It provides reliable byte stream services for data applications. Its key features include reliability, flow control, connection management, and congestion control. Major TCP versions are Tahoe, Reno, and New Reno. They differ mainly in their congestion control algorithms. Tahoe, the original version of TCP, employs three congestion control algorithms: slow start, congestion avoidance, and fast retransmit. TCP Reno extends Tahoe with a fast recovery mechanism. NewReno, the latest major version of TCP, modifies TCP Reno's fast recovery algorithm and addresses the issue of partial acknowledgements (ACKs).

Differences between the characteristics of wireline and wireless networks have significant impact on TCP performance. TCP was designed and optimized to perform well in wireline networks. Wireless links, with considerable packet losses due to link errors, delay variations, and long sudden delays, violate TCP's essential design assumptions. Improving TCP performance in wireless networks has been an ongoing research activity since the mid 90's. Most improvements dealt with TCP's reaction to high bit error rate (BER) and TCP performance degradation due to delay and delay variation in wireless links. Performance of TCP's congestion control algorithms particularly deteriorates when TCP is deployed in mixed wireline/wireless networks. We describe here TCP's timer and window management, congestion control algorithms, and round-trip time (RTT) estimation.

A. TCP Windows

TCP maintains two windows to perform congestion control and avoidance: the receiver's advertised window (*rwnd*) and the congestion window (*cwnd*). They define the maximum number of bytes the receiver may receive and the sender may send, respectively. The number of bytes that may be sent to the network is the minimum of the two. With *rwnd* sufficiently large, the larger the *cwnd*, the more data TCP can send, resulting in larger TCP throughput.

The growth of the *cwnd* is ACK paced: with every segment that TCP sends, the receiver issues an ACK to acknowledge the receipt of the data. The receipt of the ACKs increases the *cwnd* and enables the sender to send more data.

B. TCP Congestion Control Algorithms

TCP packets may be lost due to link errors or network congestion. Since losses due to link errors in wireline networks are rare, TCP deals only with packet loss due to network congestion. Hence, packet loss always implies network congestion. TCP congestion avoidance and control were first introduced when Internet experienced its first series of "congestion collapses."

TCP detects network congestion via duplicate ACKs and timeouts. Each byte of the transmitted data is assigned a unique sequence number (*seqno*). When a data packet loss occurs, TCP receiver issues a duplicate ACK for any out-of-sequence data packet received. Upon receiving a predefined number of consecutive duplicate ACKs, TCP assumes that a packet is lost.

In most TCP implementations, the threshold is set to three (known as three duplicate ACKs). Note, however, that when *cwnd* < 4 or the network is temporarily disconnected, the number of duplicate ACKs is less than three, and thus insufficient to trigger three duplicate ACKs. TCP handles this situation by keeping a timer called Retransmission Timeout (RTO). When the timer expires, it assumes packet loss [1], which triggers congestion control. TCP congestion control mechanism includes :

- increasing *cwnd* by one segment size per RTT and halving *cwnd* for every window experiencing a packet loss (Additive Increase Multiplicative Decrease, AIMD)
- Retransmission Timeout (RTO), including exponential back-off when timeout occurs
- slow start mechanism for initial probing of the available bandwidth
- ACK clocking (self-clocking) the arrival of ACKs at the sender, used to trigger transmission of new data.
  TCP Reno congestion control algorithms are shown in Fig

1.

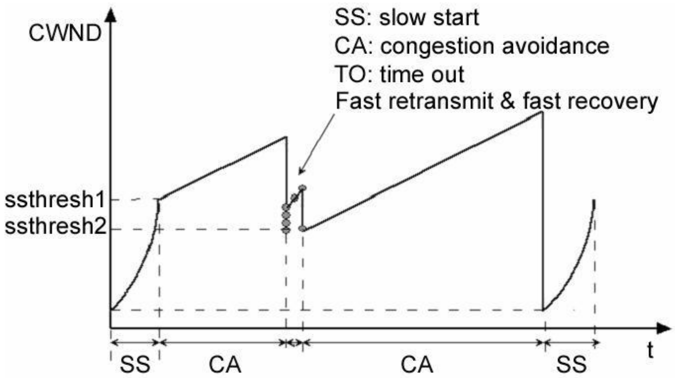

Fig. 1. TCP congestion control algorithms.

Slow Start: At the onset of a TCP connection, TCP employs the slow start mechanism to probe the network capacity. Slow start is also employed after a packet loss is detected by the RTO mechanism. When the transmission starts, the sender's *cwnd* is set to the initial window (IW) size. Congestion window *cwnd* is increased by at most SMSS (sender maximum segment size) bytes for each ACK received that acknowledges new data. The slow start threshold (*ssthresh*) may be arbitrarily high and could be reduced when congestion occurs. When congestion is detected by the RTO mechanism, *cwnd* is set to IW and *ssthresh* is set to 0.5×*cwnd*. In both situations, slow start is used as long as *cwnd* < *ssthresh* . Slow start ends when *cwnd* > *ssthresh* or when congestion is detected. When *cwnd* = *ssthresh*, the sender may use either slow start or congestion avoidance.

Congestion Avoidance: If *cwnd* > *ssthresh*, congestion avoidance is employed to probe the network capacity more slowly than during the slow start. Congestion window *cwnd* is incremented by one full-size segment per RTT. In most cases, TCP operates in the congestion avoidance phase. Congestion avoidance ends only when congestion is detected.

TCP moves from congestion avoidance to fast retransmit. The incoming segments are considered out-of-order by the receiver when a packet loss occurs. For any out-of-order packet received, the receiver immediately sends a duplicate ACK acknowledging the next expected *seqno*. After receiving three duplicate ACKs, the sender retransmits what appears to be the lost packet without waiting for the retransmission timer to expire. It uses the sequence number contained in the duplicate ACKs. Along with the retransmission, TCP also sets *ssthresh* to

$$ssthresh = \max\left(\frac{FlightSize}{2}, 2 \cdot SMSS\right)$$

where FlightSize is the size of the outstanding data in the network.

Fast Recovery*:* Fast recovery takes place immediately after the sender performs fast retransmit. Here, a new ACK is defined as the ACK acknowledging the sequence number beyond the lost segment. TCP first inflates *cwnd* to *ssthresh* +3.*SMSS* . This reflects the three segments that have left the network (three duplicate ACKs would require three packets to leave the network). For every additional duplicate ACK received, the sender increments *cwnd* by SMSS to reflect that an additional segment has left the network. This new *cwnd* may also allow the sender to transmit a new segment. When a new ACK is received, the sender sets *cwnd* to *ssthresh* to deflate the *cwnd*, and the congestion avoidance phase continues.

C. Karn's algorithm: RTT estimation and RTO

After a segment is transmitted, an ACK is expected by the sender. If the RTO timer expires before the ACK is received, the segment is retransmitted. This resynchronizes the transmission in case the segment is lost. Therefore, if the calculated RTO is too large, unnecessary time will be spent waiting for the timer to expire. Thus, it will cause TCP performance degradation [1]. If the calculated RTO is too small, the timer may expire prematurely and cause unnecessary retransmissions.

RTT is estimated using Karn's algorithm. RTO is calculated based on the estimated RTT and the RTT deviation. TCP measures the round-trip time of the ACKs for data segments and this interval is called sample RTT. The moving average of RTT, called a smoothed RTT (*srtt*), and the mean deviation (*rttvar*) are calculated as:

$$srtt = (1 - g) \times srtt + g \times sampleRTT \quad rttvar = (1 - h) \times rttvar - h \times |sampleRTT - srtt|,$$

with recommended parameter values:

$g = 0.125$ and $h = 0.25$. RTO is calculated as:

$$RTO = srtt + 4 \times rttvar.$$

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (17-23), Month: July - September 2014, Available at: www.researchpublish.com

3. CHARACTERISTICS OF WIRELESS NETWORKS

Mobile connectivity provided by wireless networks allows users to access information anytime and anywhere. The growth of cellular telephone systems is accompanied with a growing number of wireless-enabled laptops and personal digital assistants (PDAs). Cellular networks evolved from 1G analog systems to 2G systems (GSM and PDC), 2.5G systems (GPRS and PDC-P), and 3G systems (Wideband CDMA and cdma2000). During the past decade, the quality of wireless links has been improved in terms of BER and link bandwidth.

Wireless networks still exhibit the following characteristics:

A. High bit error rate (BER)

Wireless networks experience random losses. BER in wireless networks is significantly higher than in wireline networks. Packet error rates range from 1% in microcell wireless networks up to 10% in macrocell networks. Even with optimized link layer retransmission algorithms in 3G networks, packet error rate remains ~1%.

B. Long and varying delay

Wireless links have a large latency. Typical RTTs in 2.5G and 3G networks vary from a few hundred milliseconds to one second. Furthermore, they are likely to experience sudden delay changes (delay spikes) greatly exceeding the typical RTT. (Delay spike is defined as a sudden increase in the latency of a communication path) Wireless WANs have a typical latency of up to 1 s. These delay changes may cause spurious TCP timeouts. Wireless links experience delay changes due to link recovery, temporary disconnections, traffic priority, and link/MAC layer protocols.

C. Bandwidth

Bandwidth of cellular networks increased as they evolved from 1G analog systems to 2G systems (10–20 kbps for uplink and downlink), to 2.5G (10–20 kbps uplink and 10–40 kbps downlink), and 3G systems (up to 64 kbps uplink and 384 kbps downlink) . Data rates vary due to mobility and the interference from other users . Mobile users share the bandwidth within a cell. As users move among cells, they affect the bandwidth available to other users. Furthermore, a user may move to another cell with higher or lower bandwidth. These factors cause variable wireless link data rates. TCP was designed to handle the changes in bandwidth with its self-clocking scheme. However, a sudden increase in RTT could still cause spurious timeouts.

D. Path asymmetry

Cellular 2.5G and 3G systems employ asymmetric uplink and downlink data rates.

4. IMPROVING TCP PERFORMANCE

A number of solutions have been proposed to solve the problem of non-congestion related packet losses misinterpreted by TCP [10] – [12] and to reduce the impact of delays and delay variations on TCP performance in wireless networks.

A. Wireless Link Errors

The main characteristic of a wireless network is the high BER on its links. It violates the fundamental assumption of TCP that packet loss caused by link error is negligible (<< 1%) and that packet loss is caused only by network congestion. High BER in wireless networks causes packet loss regardless of network congestion. The main cause for TCP's performance degradation in a mixed wireless/wireline environment is its inability to detect the origin of the packet loss.

When a packet loss is detected, TCP employs congestion control algorithms to reduce the transmission rate. A single packet loss on the link will cause duplicate ACKs and *cwnd* to be reduced by half according to the fast retransmit and fast recovery algorithms. TCP resolves the congestion in the network by lowering its transmission rate. However, lowering the transmission rate will degrade TCP performance if the packet loss is not caused by congestion.

One approach to improving TCP performance is to reduce the adverse effect of wireless link errors. Proposed solutions either hide the wireless link error from the TCP sender or make the sender aware of the causes of segment losses. The first approach resolves the error within the wireless domain without the TCP sender being aware of the error. These solutions often modify the base station and/or the mobile host. If the link error is well shielded from the sender, modifying the sender is not necessary. The examples are I-TCP, M-TCP, and Snoop , . The second approach explicitly makes the sender aware of the wireless link error by handling differently segment losses caused by wireless link errors and losses due to network congestion. This approach requires the base station to send explicit congestion messages to the sender or a mechanism to detect the causes of loss at the sender. An example is TCP Westwood.

Based on the design principles, the solutions may also be categorized as: split connection (I-TCP), link layer retransmission (Snoop), and end-to-end (TCP Westwood , WTCP).

B. Wireless Link Delays

Wireless networks have larger latency and delay variations than wireline networks. Long sudden delays during data transfers are common in GPRS wireless WANs. Furthermore, experimental and analytical data indicate that mobility increases packet delay and delay variation and degrades the throughput of TCP connections in wireless environments. Three major adverse effects are: spurious fast retransmit, spurious timeouts, and ACK compression.

Spurious timeout: It may occur on links with long sudden delays. With its RTO timer, TCP is designed to handle even large gradual changes in delays. Nevertheless, TCP cannot handle well long sudden delays because it is unable to adjust its RTO fast enough. When the RTO timer expires, TCP assumes that the outstanding packets are lost and triggers the congestion control.

Spurious timeout is illustrated in Fig. 3. The three arrows show three critical events. The sudden long delay on the link occurs at 5 s. The first arrow indicates the moment when the TCP sender's RTO timer expires. TCP sender assumes that the previously sent packets are lost. The cwnd is reduced to the initial window (two segments). TCP then retransmits the first two unacknowledged segments. At 11 s, the link delay terminates (marked by the second arrow). The sender receives the first new ACK and starts recovering from timeouts by entering the slow start phase. All the unacknowledged segments are to be retransmitted. Since some ACKs on the wireless link have also been delayed, they accumulate and arrive together at the sender when the link recovers. This causes a burst of data segments to be sent. This is known as ACK compression. The retransmission unnecessarily utilizes the scarce wireless bandwidth and may potentially increase the recovery time.

The unnecessary retransmission of segments may introduce an additional spurious fast retransmit. At 11.97 s, the retransmitted segments arrive at the receiver. Since previously transmitted segments have been received after the link recovered, TCP receiver generates a duplicate ACK for every out-of-order segment. These duplicate ACKs (ACK 136) are shown between 11.97 s and 12.54 s. When the number of duplicate ACKs exceeds the duplicate ACK threshold, another spurious fast retransmit is triggered. This further worsens the situation. A gap appears after 12.54 s (graph labeled seqno) immediately after ACK 137 is received. During the fast retransmit, for every duplicate ACK received, the sender artificially inflates the cwnd by one segment and, if the cwnd permits, transmits the next segment. (The change in cwnd is not shown. It can be seen from the seqno showing new segments that are sent with ACKs received.) When the new ACK 137 is

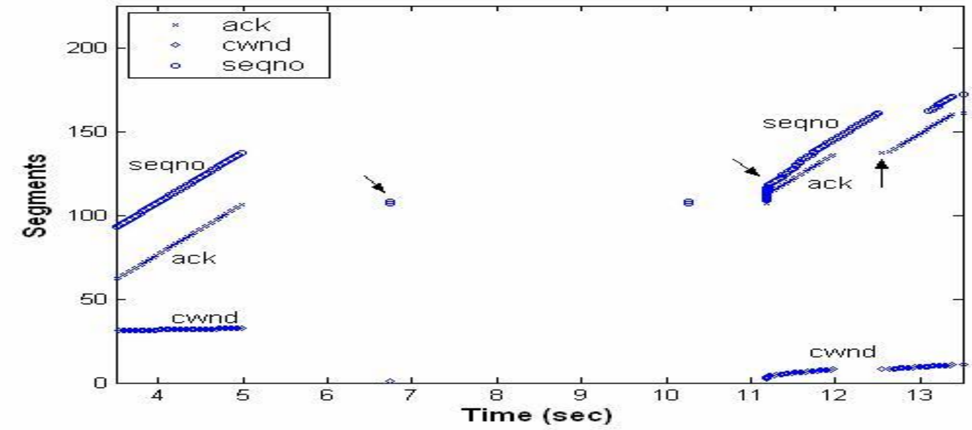

Fig. 2. Spurious timeout.

Eifel algorithm was proposed to enhance TCP's adaptation to link delays in wireless networks. Both spurious

timeout and spurious fast retransmit are caused by TCP's retransmission ambiguity, which occurs when an ACK arrives for a segment that has been retransmitted. Hence, there is no indication which transmission is being acknowledged. Eifel algorithm is an end-to-end solution, which requires modifying only the TCP sender. It first eliminates the retransmission ambiguity by using additional information in the ACKs. Then, it restores the payload and resumes transmission with the next unsent segment. Timestamp option is used to provide the additional information to identify the segment that triggered the duplicate ACK. Timestamp clock is stored in the header of every outgoing segment and echoed back with its corresponding ACK. The sender also keeps track of the timestamp of the first retransmission. The received ACK can be identified by comparing the timestamp stored in the sender with the timestamp in the received ACK. If the ACK was triggered by the original segment, spurious retransmission has occurred. The sender then restores the *cwnd* and possibly RTO. Instead of retransmitting the unacknowledged segments, the next unsent segment is transmitted.

Although Eifel algorithm effectively reduces the impact of spurious timeouts and spurious fast retransmits by eliminating the retransmission ambiguity, it has two major drawbacks: it requires modification of all TCP clients in the wireline domain and requires that both the sender and the receiver have the 12-byte TCP timestamp option enabled in every segment and the corresponding ACKs. Furthermore, its performance in the cases of high link errors is questionable.

5. CONCLUSION

In this paper, we proposed packet control filters to improve TCP performance in wireless networks with delay variations and long sudden delays. TCP connections were simulated in a mixed wireline and wireless network using the ns-2 simulator. The simulation results show that the proposed algorithms reduce spurious fast retransmit and spurious timeouts in TCP. They improve TCP's throughput, goodput, and bandwidth consumption.

In cases of long sudden delays, TCP performance is also improved, depending on the path characteristics. Packet control filters can be conveniently deployed at the intermediate routers to control the transmission of TCP segments and ACKs. Future improvements may include more accurate delay generators and multi-connection simulation scenarios while using genuine wireless traffic traces for performance evaluations.

REFERENCES

[1] W. Stevens, TCP/IP Illustrated, Volume 1: The Protocols. Reading, MA: Addison-Wesley, Professional Computing Series, 1984.

[2] Information Sciences Institute, "Transmission control protocol," IETF RFC 793, Sept. 1981.

[3] V. Jacobson, "Congestion avoidance and control," in Proc. ACM SIGCOMM, Stanford, CA, Aug. 1988, pp. 314–329.

[4] I. Stojmenovic, Handbook of Wireless Networks and Mobile Computing. New York, NY: John Wiley & Sons, 2002.

[5] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF RFC 2581, Apr. 1999.

[6] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," IETF RFC 2582, Apr. 1999.

[7] S. Floyd, "A report on recent developments in TCP congestion control," IEEE Communications Magazine, vol. 39, no. 4, pp. 84–90, Apr. 2001.

[8] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over second (2.5G) and third (3G) generation wireless networks," IETF RFC 3481, Feb. 2003.

[9] A. Gurtov, "Effect of delays on TCP performance," in Proc. IFIP Personal Wireless Communications (PWC'01), Aug. 2001.

[10] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in Proc. 15th International Conference on Distributed Computing Systems (ICDCS), Vancouver, BC, May 1995, pp. 136–143.

[11] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," IEEE Journal on Selected Areas in Communications, vol. 22, no. 4, pp. 747–756, May 2004.

[12] H. Balakrishan, S. Seshan, E. Amir, and R. H. Katz, “Improving TCP/IP performance over wireless networks,” in Proc. ACM MobiCom, Berkeley, CA, Nov. 1995, pp. 2–11.

[13] R. Ludwig and R. H. Katz, “The Eifel algorithm: making TCP robust against spurious retransmission,” ACM Computer Communications Review, vol. 30, no. 1, pp. 30–36, Jan. 2000.

[14] S. Fu, M. Atiquzzaman, and W. Ivancic, “Effect of delay spike on SCTP, TCP Reno, and Eifel in a wireless mobile environment,” in Proc. International Conference on Computer Communications and Networks, Miami, FL, Oct. 2002, pp. 575–578.

[15] K. Luo and A. O. Fapojuwo, “Impact of terminal mobility on TCP congestion control performance,” in Proc. 3rd IASTED International Conference on Wireless and Optical Communications, Banff, Canada, July 2003, pp. 360–365.

[16] K. Brown and S. Singh, “M-TCP: TCP for mobile cellular networks,” ACM SIGCOMM Computer Communication Review, vol. 27, no. 5, pp. 19–42, Oct. 1997.

[17] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby “Performance enhancing proxies,” Internet Draft [Online]. Available: http://community.roxen.com/developers/idocs/drafts/draft-ietf-pilc-pep-04.html.

[18] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang, “TCP Westwood: end-to-end congestion control for wired/wireless networks,” Wireless Networks (WINET), vol. 8, no. 5, pp. 467–479, Sept. 2002.

[19] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bhargavan, “WTCP: a reliable transport protocol for wireless wide-area networks,” in Proc. ACM MobiCom, Seattle, WA, Aug. 1999, pp. 231–241.

[20] P. Karn and C. Partridge, “Improving round-trip time estimates in reliable transport protocol,” ACM Transactions on Computer Systems, vol. 9, no. 4, pp. 364–373, Nov. 1991.

[21] ns–2 [Online]. Available: http://www.isi.edu/nsnam/ns.

[22] M. C. Chan and R. Ramjee, “TCP/IP performance over 3G wireless links with rate and delay variation,” in Proc. ACM MobiCom, Atlanta, GA, Sept. 2002, pp. 71–82.