IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), ISSN: 2250-3501
Vol.3, No2, April 2013

The Solutions for Packet Reordering in TCP over Wireless Networks

K.LakshmiNadh
Research Scholar
JNTU Kakinada
Kakinada, India
l_nadh@yahoo.co.in

K.Nageswara Rao
Principal
PSCMR College of Engineering & Technology
Vijayawada, India
drknrao@ieee.org

Y. K. Sundara Krishna
Principal
Krishna University
Machilipatnam, India
yksk2010@gmail.com

Abstract—**Packet Reordering is a phenomenon on the Internet and must be taken into account when considering performance analysis in wireless networks. Due to various characteristics specific to wireless networks, such as signal fading and mobility, packets may be lost due to congestive and noncongestive losses. The noncongestive losses violate the design principles of some traffic control mechanisms in TCP and causes performance problems. In this paper we study to evaluate the performance of four solutions for TCP packet reordering, namely, TCP-PR, TCP-DCR, TCP-DOOR, and RR-TCP, under wireless networks.**

Keywords-Transmission Control Protocol (TCP; Fast Retransmission; Wireless Networks; packet reordering;congestion control;

I. INTRODUCTION

Traditional Internet applications such as telnet and ftp use Transmission Control Protocol (TCP) [1] as their transport layer protocol, which has been designed to provide reliable data delivery between end hosts. The time elapsed between when a data packet is sent and when an ACK for the packet is received is known as the round-trip time (RTT) of the communication between the source and the destination shown in Fig.1. TCP uses multiple timers to do its work. The most important of these is the retransmission timer. When the packet is sent a retransmission timer is started. If the packet is acknowledged before the timer expires, the timer is stopped. If on the other hand, the timer goes off before the acknowledgement (ACK) comes in, TCP assumes that the transmitted packet is lost along its traveling in the internet and the packet is retransmitted.

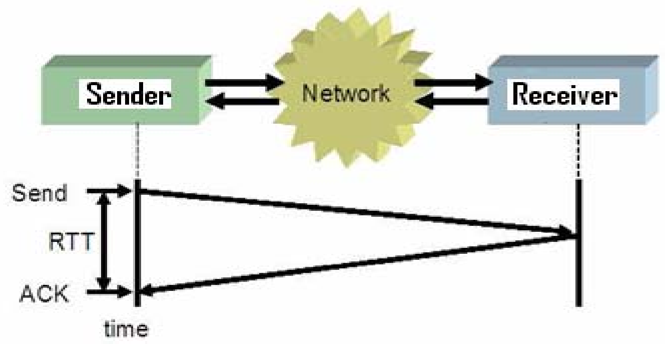

Fig 1: The Round Trip Time (RTT) represents a way to measure the network load

Protocols such as TCP requires packets to be accepted (i.e., delivered to the receiving application) in the order they are transmitted at the sender. Packets are sometimes mis-ordered in the network. For instance, since every packet contains the destination address, the network can deliberately route packets via different paths to the destination, possibly for load balancing purpose. Certain packets may be dropped by the network and retransmitted by the sender, causing the packets to arrive out-of-order at the receiver, which is an indication of the network congestion. TCP has two basic methods of finding out that a packet has been lost.

Retransmission timer

If an acknowledgement for a data packet does not arrive at the sender at a certain amount of time, then the retransmission timer (RTO) expires and the data packet is retransmitted [2].

Fast Retransmit

When a TCP sender receives three *dupacks* (duplicate acknowledgements) for a data packet *X*, it assumes that the data packet *Y* which was immediately following *X* has been lost, so it resends packet *Y* without waiting for the retransmission timer to expire [3]. Fast Retransmit uses a parameter called *dupthresh* which is fixed at three *dupacks* to conclude whether the network has dropped a packet.

TCP receivers generate cumulative acknowledgements which indicate the arrival of the last in-order data segment [4]. For example, assume that four packets *A*, *B*, *C* and *D* are transmitted through the network from a sender to a receiver. When packets *A* and *B* reach the receiver, it transmits back to the sender an *ack* (acknowledgement) for *B* which summarizes that both packets *A* and *B* have been received. Suppose packets *C* and *D* have been reordered in the network. At the time packet *D* arrives at the receiver, it sends the *ack* for the last in-order packet received which in our case is *B*. Only when packet *C* arrives, the *ack* for the last in-order packet (packet *D*) is transmitted.

The rest of the paper is organized as follows. In Section II, we introduce the TCP over Wireless Links and problems for TCP. Section III gives the existing Solutions to Packet reordering. A performance study of the algorithms under investigation is presented in Section IV. Finally, Section V concludes and discusses some possible extensions.

II.TCP OVER WIRELESS LINKS

As wireless Internet access becomes more and more popular and feasible, it is straightforward approach to use TCP over wireless links so that mobile terminals can remain compatible with existing systems. But TCP was designed for highly reliable links and stationery hosts. There are a lot of problems when it is used in lossy wireless links and mobile hosts [1]. In this section, some of the problems are listed.

Problems for TCP:

The congestion control mechanisms of TCP have been designed with the assumption that all packet losses are congestive losses. Due to the specific characteristics of wireless networks described earlier, TCP suffers poor performance because of noncongestive segment loss [5] including random loss, burst loss and packet reordering.

Packet Loss: The most outstanding characteristic of wireless links is its bit-error rate due to fading, interference, etc. It brings high segment loss rate to TCP over wireless links [6]. This fact violates the basic assumption of TCP protocol that regards segment loss as the signal of congestion. In this situation, conventional TCP will reduce the size of congestion window (cwnd) frequently. Hence cwnd will always be very small and network bandwidth cannot be fully utilized. Ideally, cwnd should not be decreased when the segment loss is caused by transmission error. High packet loss rate also brings high probability of multi-segments loss in one window. This may causes slow start even TCP New Reno is used. With high bit-error rate, the length of segment needs to be considered carefully. Large segment brings high segment loss rate and small segment brings high header overhead. It's very valuable to find the optimal value for wireless links.

Random Loss: The traditional congestion control measures for TCP have been designed for the wired network environment. In a wireless network, the loss of a data segment does not necessarily correspond to network congestion because it may be dropped due to signal fading. It is typical to have a one percent to two percent random loss rate, say, for IS-95 code division multiple access (CDMA) based data service [7]. With the misinterpretation of the nature of segment loss, the congestion control mechanisms react inappropriately by keeping the sending rate of a TCP connection small and some data segments are retransmitted spuriously. This leads to inferior performance.

Burst Loss: A burst loss event may be initiated by signal fading. In an infrastructure network, all incoming and outgoing communications for a mobile host are routed via the base station only. When it moves away from the coverage area of the base station, it needs to register at another base station in whose coverage area it moves. All subsequent communications are then routed via the new base station and the handoff process is completed. It can be shown [8] that the handover time for IEEE 802.11b wireless LANs typically takes one to two seconds to complete. However, a chain of packets delivered to a mobile host may be lost as they are routed to the old base station when a handoff is processed. Therefore, a handoff event can initiate a burst loss event. The frequency of the occurrence of a handoff event depends on the size of the coverage region and the mobility of the host involved. Thus, some packets belonging to the same traffic flow may be lost during the process. Thus, a burst loss event occurs whenever its transmission path is disrupted.

Packet Reordering: Packet reordering refers to the network behavior where the receiving order of a flow of packets differs from its sending order. Recent studies [9, 10] show that packet reordering is not a rare event. The presence of persistent and substantial packet reordering violates the inorder or near in-order channel assumption made in the design of some traffic control mechanisms in TCP. In an infrastructured network, the occurrence of packet reordering is associated with a handoff event. When a mobile host is handed over from one base station to another, packets transmitted to or sent from the host are routed via the new base station instead of the old one after the handoff is finished. Since packets traveling on different paths may take different times to arrive at a destination, packet reordering can then happen. In an ad hoc network, there is no fixed infrastructure and every mobile host can be a source, a destination, or a router. Topological changes in the network cause packets belonging to the same flow to be forwarded on different paths and arrive at a destination out of order. In addition to the issues of mobility, link-layer retransmission is another cause for packet reordering. Several link-layer retransmission approaches [2, 3] have been proposed to recover transmission errors locally by retransmitting the lost frames at the link layer.

III. SOLUTIONS TO PACKET REORDERING

Packet reordering is the main cause for Non-Congestion events. Some of the solutions for packet reordering include TCP_DOOR, TCP_DCR, TCP_PR and RR_TCP.

1) TCP-DOOR: TCP with detection of out-of-order and response (TCP-DOOR) [11] is a state reconciliation method, which recovers past congestion responses and/or disables future congestion responses for a time period, to eliminate the retransmission ambiguity and solve the performance problems caused by spurious retransmissions. The out-of-order events, which happen frequently in mobile ad-hoc networks, are deemed to imply route changes in the networks. The TCP packet sequence number and ACK duplication sequence number, or current timestamps, are inserted into each data and ACK segments, respectively, to detect reordered data and ACK packets. When out-of-order events are detected, a source can either temporarily disable congestion control or perform recovery during congestion avoidance. By temporarily disabling congestion control, the source will keep its state variable unchanged for a time period after detecting an out-of-order event. By instant recovery during congestion avoidance, the source recovers immediately to the state before the congestion response.

2) TCP-DCR: The delayed congestion response TCP (TCP-DCR) [6] is a sender-side response postponement approach, which defers a congestion response for a time period, to prevent unnecessary reduction of the congestion window size due to non-congestion events. TCP-DCR advances the time-delayed fast retransmit algorithm [3] by delaying a congestion response for a time interval after the first duplicate ACK is received. It has been suggested [5] that the delay time interval is set to one RTT so as to have ample time to deal with packet reordering due to link-layer retransmissions for loss recovery. To maintain ACK-clocking, TCP-DCR sends one new data segment upon the receipt of each duplicate ACK.

3) TCP-PR: TCP for persistent packet reordering (TCPPR) [13] is a sender-side retransmission by timeout algorithm, in which a TCP client generates an appropriate congestion response only when a retransmission timer expires, to improve the RTO timer to enhance TCP performance under persistent packet reordering. Instead of keeping track of the exponentially weighted moving average (EWMA) of the mean RTT, TCP-PR utilized a non-smoothed, exponentially weighted maximum possible RTT. By doing so, spikes in RTT can be promptly reflected in the estimated RTT. When a segment drop is detected, the size of the congestion window *cwnd* is set to half of *cwnd* at the time the segment has been sent. Congestion avoidance is then carried out. Subsequent occasional segment drops detected in the same congestion window will not cause any further reduction of *cwnd*. When more than half of a congestion window's worth of segments is inferred to be lost, *cwnd* is set to one and the slow start process is performed.

4) RR-TCP: The reordering-robust TCP (RR-TCP) [14] is a sender-side threshold adjustment solution, which adjusts the duplicate acknowledgement threshold *dupthresh* dynamically to proactively avoid, whenever possible, triggering a spurious fast retransmission and fast recovery and to avoid triggering a retransmission timeout. RR-TCP makes use of the duplicate selective acknowledgement (DSACK) option [12], which is used to report duplicate segments received, to detect segment reordering and retract the associated spurious congestion response. RR-TCP utilizes a combined cost function for retransmission timeouts, spurious fast retransmissions, and limited transmit to adapt the false fast retransmit avoidance ratio (FA radio). The FA ratio, which represents the portion of reordering events to be avoided in order to minimize the cost, can then be used to find the corresponding *dupthresh*. Thus, by changing the FA ratio based on the current network conditions, a mechanism is provided to raise or reduce *dupthresh* dynamically. RRTCP
also extends the limited transmit algorithm to permit a source to send up to one more ACK-clocked congestion window's worth of data. Besides, RR-TCP corrects the sampling bias against long RTT samples for the RTT and RTO estimations. Instead of skipping the samples for retransmitted segments in the Karn's algorithm [15], an RTT sample is taken for each retransmitted segment by taking it as the average of the RTTs for both the first and the second transmissions of that segment.

IV. PERFORMANCE EVALUATION

In this section, we present our simulation results and compare the surveyed algorithms. Section IV-A considers and compares the performance of the algorithms under study over a single-path network topology. Section IV-B investigates the performance of these algorithms and comparison of the simulation results.

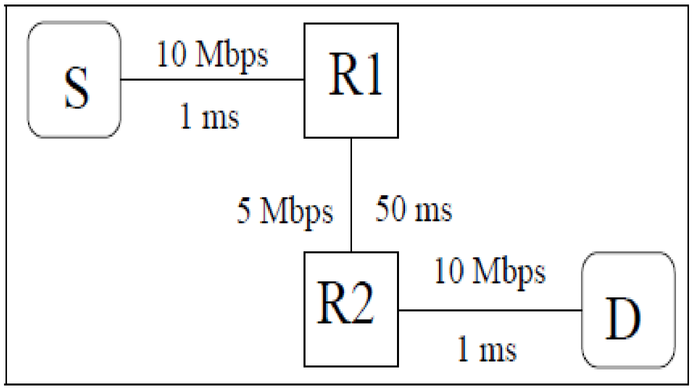

Fig.2. Single-path Network Topology

A. Simulation Setup

Our simulations are carried out in *ns-2*. The first topology, a single-path network topology, is shown in Fig. 2. It involves two end-systems (*S* and *D*) and two routers (*R*1 and *R*2). A single TCP flow from *S* to *D* lasting for 1000 seconds is simulated. The sender *S* uses the *sack*1 TCP and the receiver *D* is capable of generating the DSACK information. The path between *R*1 and *R*2 models the underlying network path connecting *R*1 and *R*2. The path usually consists of multiple hops. The central limit theorem suggests that the end-to-end delay over a multi-hop path, which is the sum of a large number of independent hop-delays, is approximately normally distributed. To simulate packet reordering, we periodically change the *R*1-*R*2 path delay according to a normal distribution. The time interval between two successive changes in delay, denoted as the delay update interval, imitates various extents of the reordering events. In our simulation, we update the delay every 50 ms or 100 ms. The former will introduce reordering events more frequently. The mean and standard deviation of the delay distribution simulate the reordering length distribution itself. The mean and standard deviation of the delay are 200*f* ms and 200*f* 3 ms, respectively, where *f* is the "packet delay factor." The factor *f* ranges from 1.0 to 3.8 in our simulations. A larger packet delay factor results in reordering events with longer reordering lengths. We are going to demonstrate the impact of the delay distribution to packet reordering when we show the simulation results with different delay update intervals and delay distributions.

B. Simulation Results and Discussion

Fig.3. depicts the goodput gain of TCP_DOOR, TCP_DCR, TCP_PR and RR_TCP under varying bandwidths ranges from

9 to 36 Mbps. The loss rate and reorder rate set to 5 and 3%, respectively. As shown in Fig.3, when bandwidth increases, the throughput of all other TCP's fluctuates lightly. When bandwidth reaches 36 Mbps, except RR_TCP the remaining TCP_DOOR, TCP_DCR and TCP_PR achieves only less than 30 Mbps goodput.

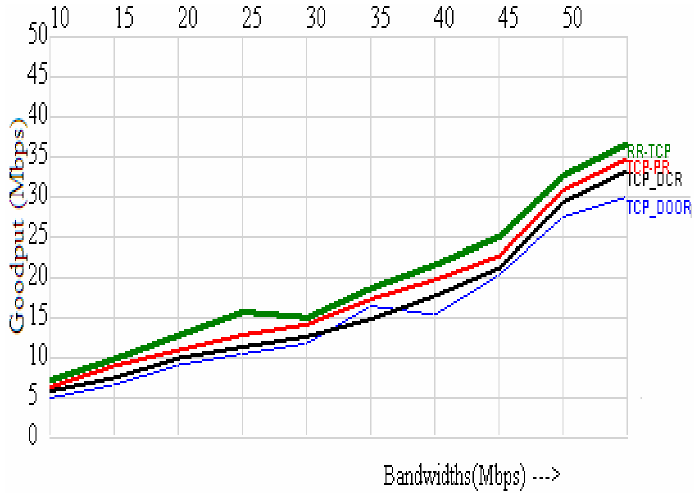

Fig.3. TCP Goodput according to different Bandwidths

Simulation results in Fig.4. Shows the goodput gain of TCP_DOOR, TCP_DCR, TCP_PR and RR_TCP in presence varying reorder rate ranges from 1 to 5%. When reorder rate increases, the performance of RR TCP becomes better than TCP_DOOR, TCP_DCR, TCP_PR.

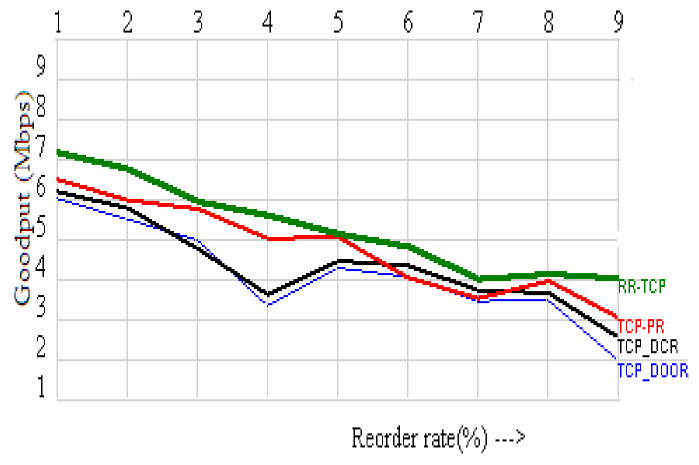

Fig.4. TCP Goodput according to different Reorder rates

V. CONCLUSION

This paper presents a simulation survey on packet re-ordering to investigate the performance of TCP over wireless networks. We have performed a simulation results to evaluate the performance of four solutions for TCP packet reordering, namely, TCP-PR, TCP-DCR, TCP-DOOR, and RR-TCP, under the scenarios of an infrastructure-based wireless network and a multihop wireless network. Various representative algorithms which preserve the end-to-end semantics are examined. Our simulation study reveals that RR-TCP outperforms all of the other algorithms, achieves a greater connection goodput. However, still TCP needs better solution for packet reordering.

REFERENCES

[1] J. Postel, "Transmission Control Protocol," Request for Comments, RFC 793, Protocol Specification, DARPA Internet Program, Sept. 1981.

[2] H. M. Chaskar, T. V. Lakshman, and U. Madhow, "TCP over Wireless with Link Level Error Control: Analysis and Design Methodology," *IEEE/ACM Trans. Net.*, vol. 7, no. 5, Oct. 1999,pp. 605–15.

[3] D. A. Eckhardt and P. Steenkiste, "A Trace-Based Evaluation of Adaptive Error Correction for a Wireless Local Area Network," *Mobile Networks and Applications*, vol. 4, no. 4, Dec.1999, pp. 273–87.

[4] Postel, J.: Transmission Control Protocol. RFC 793 (1981)

[5] Ka-Cheong Leung, Victor O.K. Li, Fellow, , and Daiqin Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP):Problems, Solutions, and Challenges" IEEE Transactions On Parallel and Distributed Systems, VOL. 18, April 2007

[6] S. Bhandarkar and A. L. N. Reddy. TCP-DCR: Making TCP Robust to Non-Congestion Events. *Lecture Notes in Computer Science*, Vol. 3042, pp. 712-724, May 2004.

[7] D. Mitzel, "Overview of 2000 IAB Wireless Internetworking Wksp.," Request for Comments, RFC 3002, Network Working Group, Internet Engineering Task Force, Dec. 2000.

[8] H. Velayos and G. Karlsson, "Techniques to Reduce IEEE 802.11b MAC Layer Handover Time," *Proc. IEEE ICC 2004*, vol.7, Paris, France, 20–24 June 2004, pp. 3844–48.

[9] J. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Trans. Net.*, vol. 7, no. 6, Dec. 1999, pp. 789–98.

[10] V. Paxson, "End-to-End Internet Packet Dynamics," *IEEE/ACM Trans. Net.*, vol. 7, no. 3, June 1999, pp. 277–92.

[11] F. Wang and Y. Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-Of-Order Detection and Response. *Proceedings of ACM MOBIHOC 2002*, pp. 217-225, Lausanne, Switzerland, 9-11 June 2002.

[12] S. Bohacek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka. A New TCP for Persistent Packet Reordering. *IEEE/ACM Transactions on Networking*, Vol. 14, No. 2, pp. 369-382, April 2006.

[13] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. *Request for Comments*, RFC 2883, Network Working Group, Internet Engineering Task Force, July 2000.

[14] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A Reordering-Robust TCP with DSACK. *Proceedings of IEEE ICNP 2003*, pp. 95-106, Atlanta, GA, USA, 4-7 November 2003.

[15] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *ACM Transactions on Computer Systems*, Vol. 9, No. 4, pp. 364-373, November 1991.

[16] V. Paxson. End-to-End Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, Vol. 7, No. 3, pp. 277-292, June 1999.