ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 5, May 2014

Simulation Based Performance Analysis of Snoop and obTCP for Vehicle to Vehicle Communication

A. Vishnuvardhan Reddy[1], **G. Vijaya Kumar**[2]

Assistant Professor, Department of Computer Science and Engineering, G. Pulla Reddy Engineering College (Autonomous), Kurnool, India[1,2]

Abstract: Internet Engineering Task Force (IETF) has recently standardized Network Mobility Basic Support Protocol to address the issue of mobility management for networks in motion (mobile networks). This development allows providing seamless Internet access to all the devices in the mobile network. A lot of research has been done to enhance the TCP performance in mobile networks and networks with wireless links, where packet losses are due to reasons other than congestion such as high Bit Error Rate. We considered Snoop and obTCP as they give promising results in mobile networks.
In this paper we consider Vehicle-to-Vehicle communication scenario where a host in an On-Board IP Network communicates directly with a host in another On-Board IP Network. We analyze the performance of TCP Snoop and obTCP in Vehicle-to-Vehicle communication under varying wireless link characteristics. We found that Snoop and obTCP performs almost same in this scenario due to complex timer cooperation among sender timer and agent's timer.
document gives formatting instructions for authors preparing papers for publication in the Proceedings of an International Journal. The authors must follow the instructions given in the document for the papers to be published. You can use this document as both an instruction set and as a template into which you can type your own text.

Keywords: NEMO, obTCP, SNOOP, Vehicle to Vehicle communication.

I. INTRODUCTION

In response to the increasing popularity of palm-top and other mobile computers, Mobile IP [1] was developed to enable these mobile devices to maintain Internet connectivity while moving from one Internet attachment point to another. In recent years we have witnessed an explosive growth of Networks in Motion. Network Mobility (NeMo) arises when a Mobile Router (MR) connecting a network to the Internet dynamically changes its point of attachment, maintaining the sessions of every device of the network. These networks are often called on-board IP Network (obIPN). A key characteristic of these systems is that the entire IP subnet is mobile and may rapidly changes its communication point to the outside world while moving. Existing IP protocols do not cope with this new requirement of entire network mobility. New extensions are required to support network mobility. IETF has recently standardized NEMO Basic Support Protocol [2] to address the issue of mobility management for networks in motion.

Transmission Control Protocol (TCP) [3, 4], works well in traditional networks made up of wired links and stationary hosts. In such networks the main cause for packet loss is congestion. TCP performs very well on such networks by adapting to end-to-end delays and packet losses caused by congestion. TCP provides reliability by maintaining a running average of estimated round-trip delay and mean deviation and calculating retransmission time out value, and by retransmitting any packet whose acknowledgment is not received within the time out duration. Due to the relatively low bit-error rates over wired networks, all packet losses are correctly assumed to be because of congestion. In the presence of the high error rates and intermittent connectivity characteristic of wireless links, in wireless last-hop networks or in obIPN, TCP reacts to packet losses as it would in the wired environment. It drops its transmission window size before retransmitting packets, initiates congestion control or avoidance mechanisms such as slow start [5], and resets its retransmission timer according to Karn's Algorithm [6]. These measures result in an unnecessary reduction in the link's bandwidth utilization thereby causing a significant degradation in performance in the form of poor throughput and very high interactive delays.

Several schemes have been proposed to mitigate the effects of non-congestion-related losses on TCP performance over networks that have wireless or similar high-loss links [7, 8, and 9]. Many proposed schemes can be classified into three basic groups, based on their fundamental working principle: end-to-end proposals, split-connection proposals and link-layer proposals. The end-to-end protocols attempt to make the TCP sender handle losses through the use of two techniques. First, they use some form of selective acknowledgments (SACKs) to allow the sender to recover from multiple packet losses in a window without resorting to a coarse timeout. Second, they attempt to have the sender distinguish between congestion and other forms of losses using an Explicit Loss Notification (ELN) mechanism. Split-connection approaches completely hide the wireless link from the sender by terminating the TCP connection at the base station. Such schemes use a separate reliable connection between the base station and the destination host. The second connection can use techniques such as negative or selective acknowledgments, rather than just regular TCP, to perform well over the wireless link. The third class of protocols, link-layer solutions, lies between the other two classes. These protocols attempt to hide link-related losses from the TCP sender by using local retransmissions and perhaps forward error correction over the wireless link. Since the end-to-end TCP connection passes through the lossy link, the TCP sender may not be fully shielded from wireless losses. This can happen either because of timer interactions between the two layers [10], or more likely because of TCP's duplicate acknowledgments causing sender fast retransmissions even for segments that are locally retransmitted. As a result, some proposals to improve TCP performance use mechanisms based on the knowledge of TCP messaging to shield the TCP sender more effectively and avoid competing and redundant retransmissions [8]

These TCP enhancement schemes are not suitable for On-Board IP Networks (ObIPN) as they consider a scenario where the last hop/link of an end-to-end TCP path is wireless and these schemes work with Mobile IP rather than NEMO Basic Support Protocol. This corresponds to the popular case of web-browsing via mobile terminals. Relatively little attention has been paid to the case of Mobile Network where multiple wireless links are present in the end-to-end TCP path. An efficient TCP performance enhancement scheme, called obTCP [11], for ObIPNs, has been proposed and it is proved in [11] that obTCP performs considerably better than TCP Snoop when multiple error prone wireless links are present. It confirms the possible use of obTCP for ObIPNs instead of other existing wireless versions of TCP.

In this paper we consider Vehicle-to-Vehicle communication where host in an ObIPN communicates directly with a host in another ObIPN. Police patrolling and other such type of applications may get benefited from this type of communication scenario where the devices and users in obIPNs can communicate with each other for better surveillance or performance.

In this paper we compare performance of Snoop and obTCP over Vehicle-to-Vehicle communication topology, which will help in understanding and improving the Vehicle-to-Vehicle communication.

In next section we describe briefly the functioning of Snoop and obTCP. Section III describes the experimental setup and Section IV presents the results and discussion. Section V concludes the paper.

II. Related Work

There are several TCP enhancement schemes proposed for wireless networks but obTCP is appropriate for ObIPNs, so we discuss working details of obTCP in this section. obTCP extends the idea of Tcp Snoop. We also present the functional details of the Snoop protocol.

A. Snoop

The Snoop protocol [8] introduces an agent called Snoop agent at the base station that monitors every packet that passes through the connection in either direction. The Snoop module maintains a cache of TCP packets sent from the FH that haven't yet been acknowledged by the MH. When a new packet arrives from the FH, Snoop adds it to its cache and passes the packet onto the routing code which performs the normal routing functions. The Snoop module also keeps track of all the acknowledgments sent from the mobile host. When a packet loss is detected (either by the arrival of a duplicate acknowledgment or by a local timeout), it retransmits the lost packet to the MH if it has the packet cached. Thus, the base station (Snoop) hides the packet loss from the FH by not propagating duplicate acknowledgments, thereby preventing unnecessary congestion control mechanism invocations.

To deal with data transfer from MH to FH Snoop protocol proposes a slight modification to the TCP code at the mobile host. At the base station, Snoop agent keeps track of the packets that were lost in any transmitted window, and generate negative acknowledgments (NACKs) for those packets back to the mobile. This is especially useful if several packets are lost in a single transmission window. These NACKs are sent when either a threshold number of packets (from a single window) have reached the base station or when a certain amount of time has expired without any new packets from the mobile.

Though the Snoop considerably improves the TCP performance in wireless last-hop networks, this protocol requires that the RTT at the wireless link be small enough to allow multiple retransmissions on wireless links before the sender retransmission timer times out. If the wire-less links are slow such that the RTT is large enough to cause the sender to time out, this leads to retransmission at the sender when the retransmission is being performed on the wireless link by Snoop agent.

B. obTCP

obTCP works by employing obTCP agents at the BS and MR. The obTCP agents maintain a cache of TCP packets sent from FH that have not yet been acknowledged by the receiver. The packets that reach the BS are handled in the same way as in Snoop protocol, but the ACKs are handled in a different way. The main function of obTCP agent at BS is to cache TCP packets and perform local retransmissions over the wireless link as reported by the obTCP agent at MR. The obTCP agent at BS will receive two types of ACK packet from obTCP agent at MR: Standard TCP ACK and SNACK packet. If an ACK is received it is forwarded to the FH and the buffer spaces are freed. However, if a SNACK is received, obTCP agent at BS checks its cache. If the packet is found, it is retransmitted immediately over wireless link. Otherwise, the obTCP agent at BS assumes that the packet has been lost due to congestion in wired network or flushed prematurely from the cache. In this case, the obTCP agent at BS sends an indication (congestion packet) to the obTCP agent at MR saying not to suppress the duplicate ACKs for these lost packets. It is important because if the duplicate ACKs are suppressed by obTCP agent at MR, it is unnecessarily delaying the Fast Retransmission from the FH because the packets are not available at the BS. In order to activate Fast Retransmission as early as possible, obTCP agent at MR should not suppress the duplicate ACKs for these lost packets.

The obTCP agent at MR has four main functions: i) caching TCP packets received from obTCP agent at BS, ii) dropping duplicate ACKs, iii) detecting and reporting packet corruption to the obTCP agent at BS, iv) retransmitting packets those are lost in the path between MR to receiver. If the obTCP agent at MR finds a gap in sequence number of the received packets, it generates a SNACK specifying all the packets those might have been lost in the wireless link and forwards to the obTCP agent at BS. If the packets reach the obTCP agent at MR in sequence it stores them in the cache and forwards to the receiver. When duplicate ACKs reach the obTCP agent at MR, it checks its cache. If the packet is found it is retransmitted. Otherwise it checks whether it has received any indication (congestion packet) from obTCP agent at BS about this packet. If the packet has been lost in wired network, it will get an indication from the obTCP agent at BS. In this case the obTCP agent at MR will not suppress these duplicate ACKs in order to initiate Fast Retransmission at the FH.

III. SIMULATION

A. *Vehicle-to-Vehicle Communication*

We compare the performance of obTCP and Snoop in a special topology where devices in one vehicle equipped with ObIPN needs to communicate with devices in another such a vehicle, directly via base station. This scenario is shown in Figure 1.

B. *Simulation Setup*

The network topology shown in figure 1 is simulated using ns-2.29 [12], [13] in Red Hat Linux 9.0. The system consists of an 11 Mbps, 5 ms propagation delay wire-less links between MH-MR and MH1-MR1, a 2 Mbps, 10 ms wireless link between BS-MR and BS1-MR1 and a 100Mb 20ms wired link between BS and BS1. The queues used in all links are drop tail. The buffer size at all links is 2000 packets. The MH is a TCP-Reno agent with maximum congestion window is of size 256 packets, slow start threshold is set to 256 packets, initial/reset value of congestion window is set to 1 packet and maximum bound on window size is 256 packets. Packet size is 1000 Bytes. To simulate the wireless link characteristics, error is incorporated in wireless links. The wireless links between MH-MR and MH1-MR1 have 1% error rate. The error wireless links between MR-BS and MR1-BS1 has been varied from 1% to 20% error rate. We assume wired link between base stations are error free, so that we can concentrate on how obTCP and Snoop recover from non-congestion related losses. The simulation duration is 2000 sec.

The performance metrics for our simulations is TCP Throughput. To analyze the performance of TCP, we calculate the throughput at TCP receiver. If N is the total number of received packets and T is the simulation duration, then the throughput Tpt in packet/Sec is calculated as

$$Tpt = \frac{N}{T} \qquad (1)$$

IV. RESULTS AND DISCUSSION

We compare the performance of Snoop and obTCP under the impact of link errors in Vehicle-to-Vehicle Communication scenario. We carry out investigations for different link error scenarios and different TCP control parameters.

The throughput of the TCP depends on how the congestion window varies. In Vehicle to Vehicle Communication scenario, there exist more number of wireless links and more agents (at MR, BS, MR1 and BS1 respectively). Whenever a packet loss is detected in a wireless link by any of these agents, they fallow local retransmission mechanism. And each agent (Snoop or obTCP) maintains its own timer for each retransmitted packet. Based on these timers the agent transmits the loss packet from its cache. At high error rates there are too many packet losses in this topology and there is a chance that locally retransmitted packet is lost resulting in more number of retransmissions. These agents require that the RTT at the wireless link be small enough to allow multiple retransmissions on wireless links before the sender retransmission timer times out. If RTT at the wireless link is high and error rate of wireless link is also high Snoop/obTCP agent can not perform sufficient number of retransmissions to recover from the loss and eventually sender retransmission timer times out. So there should be well coordination between agent timer and TCP sender timer. And TCP consider these timeouts as indication of congestion and decreases its sending rate, leading to lesser throughput. Snoop employs two agents in this topology, one at BS and one at BS1 resulted in fewer local retransmissions in the network. And propagation of duplicate acknowledgements by Tcp receiver is relatively fast in Snoop. So timeouts are lesser than in the case of obTCP. The obTCP though performs well in ObIPNs, in this topology it yields *throughput almost equal to that of Snoop*. This observation is evident from the figures 2 to 6 where TCP sender, when employed obTCP agent at BS and MR, times out more frequently.

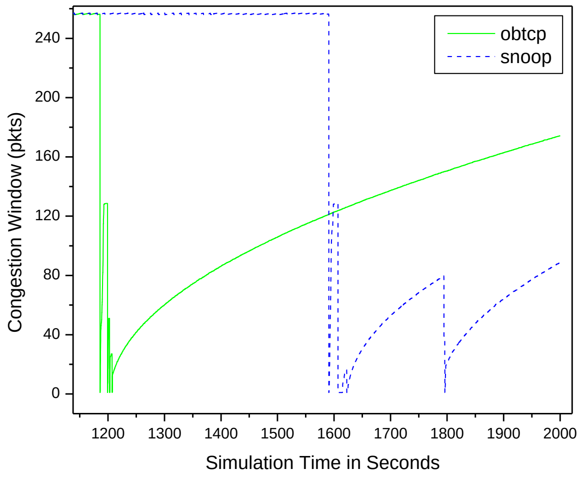

Fig. 2. Congestion Window variations vs. Simulation Time at error rate 1% at link A and D, 1% at link B and C respectively

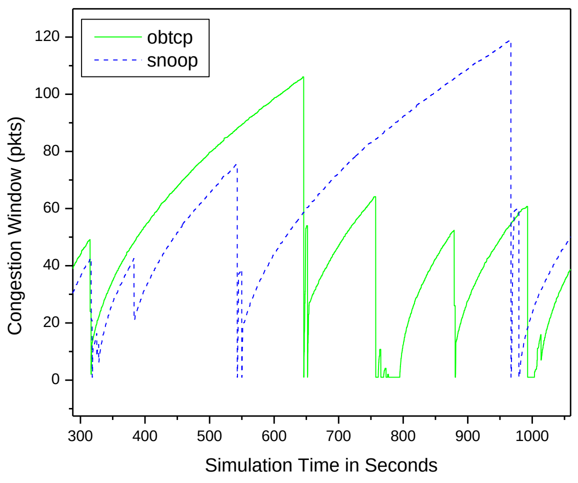

Fig. 3. Congestion Window variations vs. Simulation Time at error rate 1% at link A and D, 5% at link B and C respectively

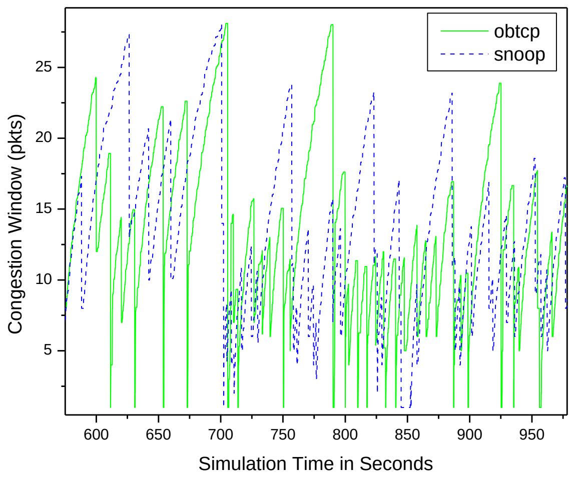

Fig. 4. Congestion Window variations vs. Simulation Time at error rate 1% at link A and D, 15% at link B and C respectively

From the figures 2, 3 and 5 we can observe more number of timeouts in obTCP than in Snoop.

TABLE 1
VALUES OF PACKETS DELIVERED, CONGESTION WINDOW CUTS AND TIMEOUTS OBTAINED IN SIMULATION

Error Rate	Packets delivered (obTCP)	Time outs (obTCP)	Cwnd cuts (obTCP)	Packets delivered (Snoop)	Time outs (Snoop)	Cwnd cuts (Snoop)
1	36131	4	4	35831	10	11
5	30483	47	54	31060	18	28
10	24335	81	115	25128	38	102
15	18183	128	223	19915	38	234
20	13112	199	339	13721	169	512

Above table shows the values of total number of packets delivered during the simulation time with obTCP and Snoop as well.

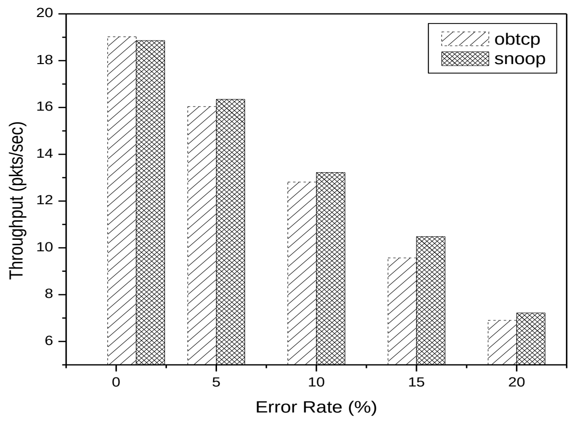

Fig. 5. Throughput at varying error rates at links B and C

Congestion window is dropped for every fast retransmission (upon receiving duplicate *acks*) or timeout event. In the event of timeout congestion window is set to its initial value. We can observe in Table 1, Snoop causes more congestion window drops than obTCP at different instances of error at wireless links. But timeouts out of total congestion window cuts are more in obTCP than in Snoop. It occurs when Tcp sender times out before DUPACKs from Tcp receiver reaches the Tcp sender. The throughput values in Table 1 and the throughput bars in figure 7 shows that Snoop and obTCP perform equally well in this topology.

V. CONCLUSION

We compare the performance of Snoop and obTCP under the impact of link errors in Vehicle-to-Vehicle Communication scenario. We carry out investigations for different link error scenarios and different TCP control parameters. We found that Snoop and obTCP performs almost same in this scenario due to complex timer cooperation among sender timer and agent's timer.

REFERENCES

[1] C. Perkins, "IP Mobility Support," IETF RFC 2002, Oct. 1996.
[2] Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu and Pascal Thubert. "Network Mobility (NEMO) Basic Support Protocol". IETF RFC 3963, Jan 2005.
[3] J. B. Postel. Transmission Control Protocol. RFC, Information Sciences Institute, Marina del Rey, CA, September 1981. RFC 793.
[4] Douglas E. Comer, Internetworking with TCP/IP, vol-1. Prentice Hall, India, 2006.
[5] V. Jacobson. Congestion Avoidance and Control. In Proc. ACM SIGCOMM 88, August 1988.
[6] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. ACM Transactions on Computer Systems, 9(4):364–373, November 1991.
[7] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In Proc. 15th International Conf. on Distributed Computing Systems (ICDCS), May 1995.
[8] H. Balakrishnan, S. Seshan, and R.H. Katz. Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. ACM Wireless Networks, 1(4), December 1995.
[9] R. Yavatkar and N. Bhagwat. Improving End-to-End Performance of TCP over Mobile Internetworks. In Mobile 94 Workshop on Mobile Computing Systems and Applications, December 1994.
[10] A. DeSimone, M. C. Chuah, and O. C. Yue. Throughput Performance of Transport-Layer Protocols over Wireless LANs. In Proc. Globecom '93, December 1993.
[11] B. Sardar, P.Chand, D. Saha, "A Novel Version of Wireless TCP for Vehicular On-Board IP Networks", IEEE VTC, Vol. 2, pp. 876-880, 2006.
[12] Network Simulator, ns-2 web site. [Online]. Available: http://www.isi.edu/nsnam/ns/
[13] Kevin Fall, ed., Kannan Varadhan, ed., "The ns Manual," VINT Project, November 2008.

BIOGRAPHIES

A Vishnuvardhan Reddy, is working as an Assistant Professor in Department of Computer Science and Engineering of G. Pulla Reddy Engineering College (Autonomous), Kurnool, India. He received his M.E degree in Software Engineering from Jadavpur University, Kolkata, India. He received his B.Tech degree in Computer Science and Engineering from Sri Venkateswara University, tirupati, India. His area of research is in the design and implementation of transport layer protocol for vehicular ad-hoc networks.

G Vijaya Kumar, received his B.Tech and M.Tech Degree in Computer Science & Engineering from Jawaharlal Nehru Technological University, Hyderbad, AP, India, in 2002, and 2006 respectively. At present, he is pursuing Ph.D in Computer Science & Engineering from Jawaharlal Nehru Technological University, Anantapur, India. His research interests include mobile ad hoc networks, cross-layer design, network security and wireless mesh networks. Currently he is working as Assistant Professor in Depatment of Computer Science and Engineering of G.Pulla Reddy Engineering College (Autonomous), Kurnool, Andhra Pradesh. He has 9 years of teaching experience.