On the use of Snoop with Geostationary Satellite Links

Joel Sing and Ben Soh
Department of Computer Science and Computer Engineering
La Trobe University
Bundoora VIC 3083, Australia
joel@ionix.com.au, ben@cs.latrobe.edu.au

Abstract

The performance of TCP decreases significantly when used over networks that exhibit a high bit error rate, as TCP has insufficient information to distinguish between loss that is due to corruption and that which is due to congestion. As a result, all loss is assumed to be congestion based, causing TCP to invoke its congestion control algorithms, even if the loss was not congestion related. A TCP-aware link level protocol known as Snoop was developed for use with wireless networks and it has been shown to provide significant improvements in TCP over wireless performance. Given that satellite links and wireless networks share many characteristics that result in high bit error rates, the use of Snoop may provide similar performance gains for satellite links. This paper details research undertaken to determine the applicability of Snoop to geostationary satellite links. A model is developed and simulated, with the results being detailed, discussed and analysed.

1. Introduction

The Transmission Control Protocol (TCP) [10] suffers a decrease in performance when used over networks that exhibit a high Bit Error Rate (BER); an issue that has been heavily researched and well documented, particularly for wireless [6, 5, 12] and satellite links [3, 2, 9, 8, 4, 7, 11]. Having been primarily designed for use over wired networks, the congestion control algorithms implemented in TCP interpret any loss within the network as congestion. As a result, when TCP is used over networks that exhibit a high BER, performance is severely degraded whenever data is lost due to corruption, as this causes TCP to invoke its congestion control algorithms reducing the amount of data being injected into the network.

A lot of research has been undertaken in order to improve the performance of TCP over wireless networks. One promising outcome has been the development of Snoop [5], a TCP-aware link level retransmission protocol designed specifically for use with wireless networks, being implemented at the wired-to-wireless gateway or base station. Numerous researchers [5, 12] have investigated the application and effectiveness of Snoop when used in wireless networks. However, it would appear that the use of Snoop within a geostationary satellite environment has not been contemplated nor investigated.

This paper details research that investigates the application and use of Snoop within networks that utilise geostationary satellite links. Section 2 provides the background and basis for the research, with the design of the simulation model being outlined in section 3. Section 4 presents the results from the simulations which are tabled, analysed and discussed in detail. Section 5 describes possibilities for further research. Finally, conclusions are drawn in section 6.

2. Background

Wireless and satellite networks share a number of characteristics. Both typically exhibit a high BER due to loss of signal strength, channel fading, noise and interference - all of which are undesired sideffects caused by radio based transmission. Additionally, both can suffer transmission delays and disruptions resulting from handoffs between base stations or satellites, as is the case for Low Earth Orbit (LEO) or Medium Earth Orbit (MEO) constellations. Arguably the biggest difference between the two technologies is defined by the propagation delay and corresponding Round Trip Time (RTT). For a local wireless network, the propagation delay can be in the order of a few milliseconds, with geostationary satellite links typically exhibiting a 250ms propagation delay, resulting in an RTT that exceeds 500ms.

The Snoop protocol consists of two modules; *snoop_data()* and *snoop_ack()*. The first module, *snoop_data()*, monitors all packets being routed across the wireless link and records the last sequence number

Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)
0-7695-2316-1/05 $20.00 © 2005 IEEE

forwarded for the connection. In addition, it maintains a local cache of each unacknowledged packet. The second module, *snoop_ack()*, monitors the acknowledgments being sent back over the wireless link to the sender. If the acknowledgment is for a sequence number that is higher than the last acknowledged sequence number, Snoop removes all acknowledged packets from the local cache, updates its RTT estimate for the wireless link and forwards the acknowledgment to the sender. If however the acknowledgment is a duplicate of the last acknowledgment, indicating that the next packet in sequence has not been received, one of three things will occur. If a copy of the lost packet is available in the local cache and it has not been previously retransmitted by the sender, Snoop will retransmit the packet over the wireless link. An estimate of the maximum number of duplicate acknowledgments is then calculated based on the number of packets that were transmitted between the lost packet and its retransmission. Further duplicate acknowledgments that correspond to these packets are simply discarded. If the packet is not cached locally, or has been previously retransmitted by the sender, the duplicate acknowledgment is forwarded to the sender. Snoop will also perform timeout based retransmissions, determined by using the local RTT estimate for the wireless link. Any local retransmissions performed will reduce the need for end-to-end retransmissions and minimise costly timeouts. The full specification for the Snoop protocol is available in [5].

Due to the similarities between wireless and satellite networks, it is envisaged that similar performance gains would be experienced when Snoop is implemented at the terrestrial gateway for a geostationary satellite link. As previously identified, the major difference between the two technologies is defined by the difference in propagation delay. The impact of the significantly increased RTT on Snoop is currently unknown.

3. Simulation Design

In order to evaluate the effectiveness of Snoop when implemented at the terrestial gateway to a geostationary satellite link, a model was developed for use with Network Simulator 2 (*ns2*) [1] - a discrete event simulator designed for networking research. The model emulated a 560ms Round Trip Time (RTT) over three nodes that were interconnected using a full duplex link and a geostationary satellite link. The geostationary satellite link in turn consists of two satellite terminals communicating via a geostationary satellite. The exact topology used for the simulation is shown in figure 1.

The error model used during simulations was a fixed rate, bit distributed model, with the error rate being varied from 1 in 10^4 bits through to 1 in 10^{13} bits. Each of the main

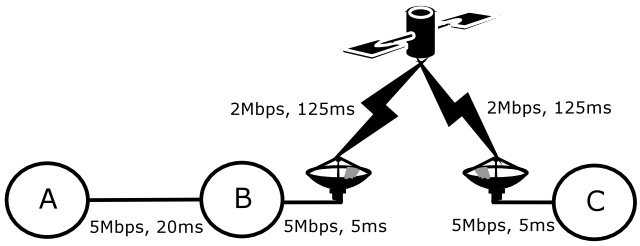


Figure 1. Topology of Simulated Network

TCP variants (Tahoe, Reno, New Reno, SACK, FACK and Vegas) were simulated at each error rate, with and without the use of Snoop at the terrestrial gateway. A single FTP session was established between the two end nodes and data transferred from node *A* to node *C* continuously for 300 seconds. The TCP receive window (*rwnd*) size and link queue sizes were set to large values in order to prevent packet loss and throughput decreases caused by third-order effects. All simulations used a maximum segment size of 500-bytes, an initial window size of two segments and delayed acknowledgments where implemented at the receiver. The *ns2* simulator was used to execute each of the 120 runs, generating multiple trace files from which throughput data was extracted.

4. Results

Performance was measured using TCP sequence numbers, providing an indication of achieved throughput at a given point in time. The TCP sequence numbers of packets successfully received at node *C* were recorded, along with the arrival time of the packet. Tabular versions of the maximum sequence numbers were generated and time versus sequence number graphs plotted (see figure 2). Table 1 details the maximum sequence number received for each TCP variant during simulation at error rates between 1 in 10^5 bits and 1 in 10^{10} bits, without the use of Snoop. Table 2 shows the maximum sequence number received for each variant at the various error rates, with Snoop being implemented at the terrestrial gateway. Table 3 show the difference in sequence numbers caused by the implementation of Snoop. Finally, table 4 details the performance gained via the use of Snoop as a percentage.

It is interesting to note that all TCP variants excluding TCP Vegas, perform equally as well with or without the use of Snoop, at an error rate of 1 in 10^9 bits or lower. TCP Vegas appears to open its congestion window more conservatively, increasing the duration of slow-start, which in turn decreases the overall throughput achieved during the connection lifetime.

TCP Vegas also exhibits some unusual properties. It is

Variant \ BER	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	1553	5730	18585	95135	142076	142076
Reno	1772	6749	18624	89687	142076	142076
New Reno	1924	6597	18624	89687	142076	142076
SACK	1800	6160	18368	78575	142076	142076
FACK	1619	7834	20515	79358	142076	142076
Vegas	2760	9018	27812	88771	112549	112549

Table 1. TCP Sequence Numbers without Snoop

Variant \ BER	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	3745	13911	52528	114528	142076	142076
Reno	6329	23694	42893	96195	142076	142076
New Reno	3906	27668	44414	96111	142076	142076
SACK	3335	30936	44702	84769	142076	142076
FACK	3201	17021	48249	83818	142076	142076
Vegas	3085	13094	42874	86113	112549	112549

Table 2. TCP Sequence Numbers with Snoop

Variant \ BER	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	2192	8181	33943	19393	0	0
Reno	4557	16945	24269	6508	0	0
New Reno	1982	21071	25790	6424	0	0
SACK	1535	24776	26334	6194	0	0
FACK	1582	9187	27734	4460	0	0
Vegas	325	4076	15062	-2658	0	0

Table 3. Difference in Sequence Numbers using Snoop

Variant \ BER	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	141%	143%	183%	20%	0	0
Reno	257%	251%	130%	7%	0	0
New Reno	103%	319%	138%	7%	0	0
SACK	85%	402%	143%	8%	0	0
FACK	98%	117%	135%	6%	0	0
Vegas	12%	45%	54%	-1%	0	0

Table 4. Performance Gained using Snoop

the best performing variant when the error rate is 1 in 10^7 bits or higher, being surpassed by TCP Tahoe, Reno and New Reno when the error rate is 1 in 10^8 bits. At an error rate of 1 in 10^5 bits and without Snoop, TCP Vegas performs 43% better than its nearest rival, TCP New Reno. With the introduction of Snoop, TCP Vegas becomes the worst performing variant at the same error rate, achieving less than 79% of TCP New Reno's throughput and less than 49% of TCP Reno's throughput. This is almost the inverse of the results detailed in [12] where TCP Vegas outperforms all other variants when Snoop was implemented. Additionally, at an error rate of 1 in 10^8 bits the implementation of Snoop causes TCP Vegas to perform worse than it does at the same error rate without Snoop. TCP Vegas may be sensitive to the large RTT emulated in this model, however further research would be required to clearly identify the cause of this behaviour.

TCP SACK performs poorly when used over a geostationary satellite link that exhibits an error rate of 1 in 10^7 bits or higher, being consistently outperformed by TCP Vegas and TCP New Reno. TCP FACK also outperforms TCP SACK for error rates between 1 in 10^6 bits and 1 in 10^8 bits. The performance of TCP SACK increased significantly with the use of Snoop. At an error rate of 1 in 10^6 bits the use of Snoop resulted in TCP SACK performing 402% better; a performance gain of 143% was also achieved at an error rate of 1 in 10^7 bits.

The biggest gain in sequence numbers was achieved when TCP Tahoe was used on a network that had an error rate of 1 in 10^7 bits. With this configuration the implementation of Snoop increased throughput by 33,943 sequence numbers.

The Reno family appears to be the most consistently performing variants, providing reasonable throughput both with and without the use of Snoop. The introduction of Snoop did however provide a significant increase in performance for both TCP Reno and TCP New Reno, with throughput increasing more than three-fold when the network exhibited an error rate of 1 in 10^6 bits.

If the error rate is 1 in 10^9 bits or less, most TCP variants appear to only suffer from third-order effects such as those caused by the maximum receive window size and router buffer sizes. At an error rate of 1 in 10^9 bits or lower TCP Tahoe, Reno, New Reno, FACK and SACK provide identical performance. TCP Vegas performs slightly worse, presumably due to the increased slow-start phase. In general, if the link exhibits an error rate of 1 in 10^8 bits or higher, significant improvements in TCP performance are experienced by implementing Snoop.

5. Further Research

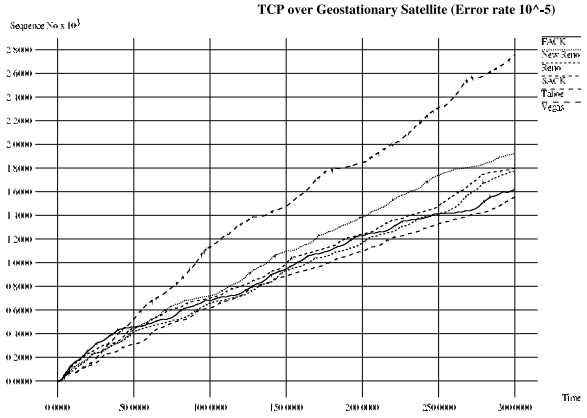
As discussed in section 4, each of the TCP variants behaved extremely differently as the error rate was varied. Also, the introduction of Snoop made some of the best performing variants perform the worst and vice versa. The interaction between each variant and Snoop could be investigated further, along with the exact effect of a specific error rate. These complex relationships may provide further insight into ways to increase TCP performance and to refine the congestion control algorithms. Additionally, the effect of multiple TCP connections traversing the same geostationary satellite link could be explored.

Given that the results from this research are highly promising, implementation and testing in a real environment would be extremely beneficial, as this would confirm the accuracy of the simulated model and prove or disprove the applicability of Snoop to geostationary satellite links. The authors plan to pursue this work and document the findings in a future paper.

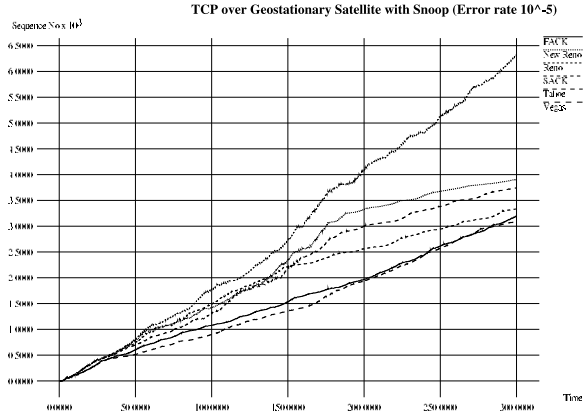
The error model used for these simulations was a fixed rate, bit distributed model, whereas radio based transmission tend to exhibit error bursts and variances in error rate, due to channel fading, noise and interference. The development of a more realistic error model, either based on a two-state Markov model or simple multi-state model, would be advantageous for this research. A fixed rate, bit distributed model provides for simple analysis, hence the reason for its use. Additionally, it would appear that the two-state Markov error model is currently broken in *ns2*. The simulations could be re-executed with a more appropriate error model and the results compared to those documented in this paper.

The implementation of Snoop in *ns2* is currently difficult to use for the model simulated in this research. Further development and refinement of *ns2* could be undertaken to simplify the process of simulating Snoop on a full duplex link.

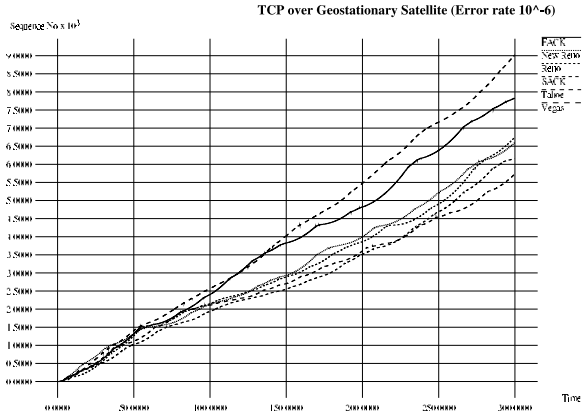
Finally, Snoop is rendered useless when network layer encryption is deployed, as this prevents the transport layer from being accessed or modified. As a result, Snoop is unable to view and cache TCP packets, nor is it able to drop acknowledgments. This for example, becomes a problem when a geostationary satellite link or wireless link forms part of a Virtual Private Network (VPN). The use of a finer grained network layer encryption protocol, such as ML-IPSec [13], could be configured to allow the terrestrial gateway access to the TCP headers, in turn allowing Snoop to operate correctly. This is another area for future research and one that is of significant interest to the authors.



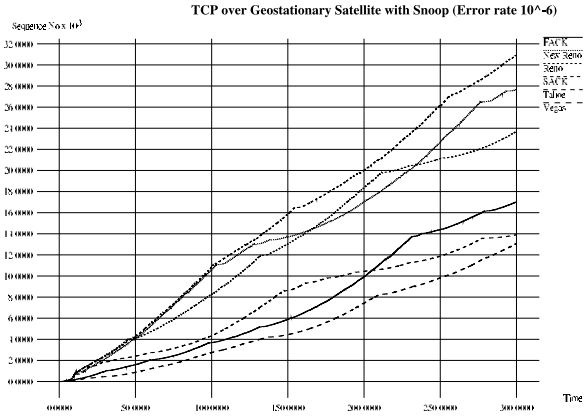
(a) TCP Throughput without Snoop (10**-5)



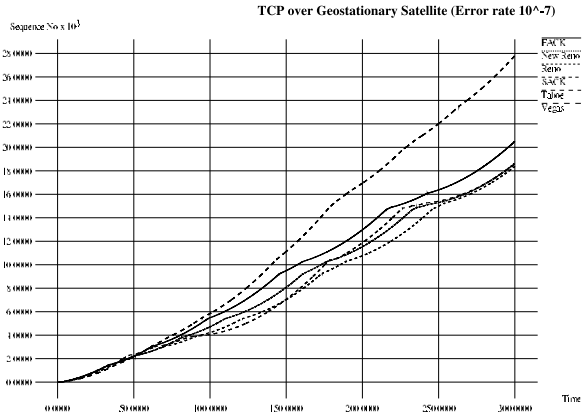
(b) TCP Throughput with Snoop (10**-5)



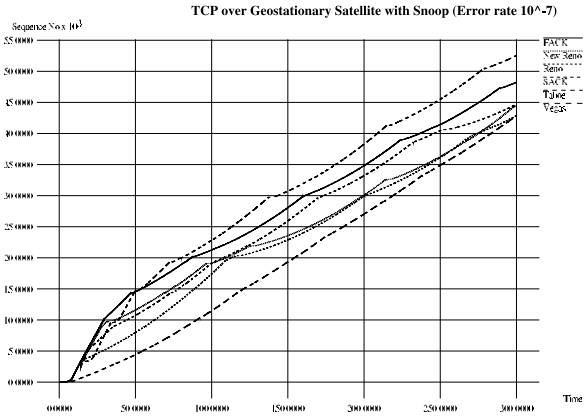
(c) TCP Throughput without Snoop (10**-6)



(d) TCP Throughput with Snoop (10**-6)



(e) TCP Throughput without Snoop (10**-7)



(f) TCP Throughput with Snoop (10**-7)

Figure 2. TCP Throughput Graphs

6. Conclusion

This research shows that the implementation of Snoop at a terrestrial gateway to geostationary satellite link will provide a significant increase in performance when the error rate of the satellite link is 1 in 10^8 bits or higher. Furthermore, with the exception of TCP Vegas, the implementation of Snoop does not appear to decrease performance under any conditions. This would suggest that a Snoop implementation may still be beneficial for links that typically have an error rate of 1 in 10^9 bits or lower. If the error rate did increase due to channel fading, noise or interference, the Snoop implementation would assist to maintain the performance of TCP until the error rate returned to normal levels.

If Snoop is not implemented at the terrestrial gateway, the use of TCP Vegas may provide an increase in throughput, particularly for link error rates between 1 in 10^5 bits and 1 in 10^7 bits. If Snoop is implemented at the terrestrial gateway the best performing variant changes with the error rate. At an error rate of 1 in 10^5 bits, TCP Reno achieves the highest throughput. At an error rate of 1 in 10^6 bits TCP SACK achieves the highest throughput, and at error rates of 1 in 10^7 bits and 1 in 10^8 bits TCP Tahoe achieves the highest throughput. The use of Snoop results in TCP SACK performing between 8% and 402% better when the error rate is 1 in 10^8 bits or higher. Given the ubiquitous use of TCP SACK, the deployment of Snoop seems to be a logical and simple way to increase TCP performance over geostationary satellite links.

7. Acknowledgment

This work was supported by the Australian Research Council (ARC) along with industry partners, Telstra Corporation and Ursys, under a research grant. We thank the ARC and our industry partners for their support and assistance.

This project would not have been possible without the hard work, dedication and support of numerous people. In particular, we would like to thank Jean Armstrong, David Ensor, Michael Feremez, Mike Flavel, Peter Goddard, Robert Hinrichs, Jennifer Jones, Tony Knox, Mary Martin, Phil Scott, Jeremy Swift and the beautiful Nikki Borchard.

References

[1] The Network Simulator 2 (ns2). `http://www.isi.edu/nsnam/ns/`.

[2] M. Allman. Ongoing TCP research related to satellites. Informational, RFC 2760, Internet Engineering Task Force, 2000.

[3] M. Allman, D. Glover, and L. Sanchez. Enhancing TCP over satellite channels using standard mechanisms. BCP 28, RFC 2488, Internet Engineering Task Force, 1999.

[4] M. Allman, C. Hayes, H. Kruse, and S. Ostermann. TCP performance over satellite links. In *Proceedings of the 5th International Conference on Telecommunication Systems*, 1997.

[5] H. Balakrishnan, S. Seshan, and R. Katz. *Improving reliable transport and handoff performance in cellular wireless networks*, pages 469–481. Wireless Networks, J.C.Baltzer AG, December 1995.

[6] H. Balkrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 1997.

[7] D. Glover and H. Kruse. *TCP Performance in a Geostationary Satellite Environment*. Annual Review of Communications 1998, International Engineering Consortium, April 1998.

[8] C. Metz. TCP over satellite... the final frontier. *IEEE Internet Computing*, 1999.

[9] C. Partridge and T. Shepard. TCP/IP performance over satellite links. *IEEE Network*, 1997.

[10] J. Postel. Transmission Control Protocol. RFC 793, USC/Information Services Institute, 1981.

[11] J. Sing and B. Soh. TCP performance over geostationary satellite links: problems and solutions. In *Proceedings of the 12th IEEE International Conference on Networking*, pages 14–18. IEEE, November 2004.

[12] S. Vangala and M. Labrador. Performance of TCP over wireless networks with the Snoop protocol. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, 2002.

[13] Y. Zhang and B. Singh. A multi-layer IPsec protocol. In *Proceedings of the 9th USENIX Security Symposium*. USENIX, August 2000.