Boosting TCP with Cross-Layer InformatioAwareness in Wireless Networks

Mouhcine Guennoun
School of Information Technology and Engineering
University of Ottawa
800 King Edward Avenue Ottawa, ON, Canada
mguennou@uottawa.ca

ABSTRACT: *It has been demonstrated through simulation and experimental studies that TCP protocol suffers from a significant drop of performance when communicating over wireless networks. In general, packets can be lost for three main reasons: 1) Congestion 2) Link failure due to node mobility, and 3) Losses due to transmission errors. However, TCP doesn't distinguish between these three types of packet losses. It considers any packet loss as a sign of network congestion; and therefore triggers its exponential back off as a mean to control congestion. In this paper, we study the impact of providing TCP with explicit information extracted from the physical, link and network layers to make it possible to identify the cause of the packet loss. There are three information of interest to this study: maximum probability to drop a packet, signal strength of the wireless node, and explicit handover notification when the mobile node transits to a new access point. We show in the experimental section that by providing TCP sender with only the information regarding the queues states, the performance can be improved by 500% over a lossy wireless link (with 60% error rate).*

Keywords: Boosting TCP;Wireless Networks; Cross-Layer Information

© 2011 DLINE. All rights reserved

1. Introduction

Many solutions have been proposed to improve the performance of TCP protocol over wireless networks. In [7], Holland et al. showed an improvement of the TCP throughput when explicit link failure notification (ELFN) is used to notify the sender of route failures. In [5], the authors suggested to freeze the TCP state upon receiving an ELFN notification. Therefore, after a new route is established, TCP sender resumes operation using the old parameters instead of the default parameters of the congestion window and RTT. While this prevents undesirable timeouts from occurring, we believe that this solution may not be suitable in case the new route has different characteristics (BER, delay).

Explicit feedback has been widely proposed in literature as a mechanism to provide the TCP sender with information about congestion [6], transmission error in wireless links [2], and link failures due to mobility [3]. These solutions enhance TCP with mechanisms that can distinguish congestion losses from random errors and link failures. A particular solution that is of interest to this work is presented in [9]. Sundaresan *et al.* proposed ATP protocol and compared its performance with regular TCP and TCP-ELFN. ATP relies on feedback not only from the receiver, but also from the intermediate nodes in the connection path. Intermediate nodes set the value of the maximum delay experienced by the packet at intermediate nodes. This information is sent back to the sender as a field in the ACK message. Using this extra information, the TCP sender can provide efficient connection and congestion controls. According to their simulation studies, ATP outperforms regular TCP and TCP-ELFN. However, we believe that providing the sender with the maximum delay experienced at intermediate nodes is of little value since the RTT takes into considerations all delays experienced by the packet.

This paper is organized as follows: After a brief introduction in section I, section II presents few techniques to improve TCP performance. We discuss the proposed scheme in section III. Experimental results are provided in Section IV and finally, conclusions and plans for future work are provided in Section V.

2. Related Work

Mobility has the most significant impact on TCP performance. According to [8], TCP achieves only about 10% of the reference throughput in highly mobile scenarios where node speed is 20 m/s. On the other hand, congestion and moderate channel errors have less impact on TCP, with less than 10% performance drop compared with the standard TCP. This is due to the fact that TCP is unable to distinguish between congestion, link failure and transmission errors, and hence reduces its sending rate for all errors, assuming congestion exists in the connection path. Various TCP schemes have been proposed to improve TCP performance over complex wireless networks including end to end solutions and cross layer information awareness. In most of these solutions, the receiver sends back feedback to the sender to improve the "visibility" of TCP regarding the cause of packet losses.

We present in this section a brief literature review to present few techniques to improve TCP performance over wireless links. Most of the works discussed in this section proceed by sending cross layer information back to the sender. This information is usually collected by the transmitted packet and sent back to the sender piggybacked on the ACK packet.

In [4], based on the fact that the smaller TCP ACK packets consume channel resource, the authors proposed a new ACK technique to reduce the number of ACKs sent by the receiver. The sender appends the current CWND to the TCP packet. The receiver sends an ACK only if: ACK-delay timeout occurs or number of received-but-unacknowledged packets equals cwnd. According to their simulation results, this solution improves TCP performance by 205%.

In [10], the authors observed that the change of the RTT between consecutive packets can represent the situation of network congestion. The receiver determines the reason of packet loss according to the cumulative change of time interval of consecutive packet, and notifies the information to the sender by setting Explicit Loss Notification (ELN) bit of ACK packet. The sender decides to backoff if ELN=0, otherwise it considers the packet is lost because of link errors. This technique improved TCP performance by 30%.

Instead of cutting the congestion window to half (like standard TCP), the proposed scheme in Congestion Window Action delays the cut decision until TCP receive all duplicate acknowledgment for a given window of data (packets on flight) [1]. This will give TCP a clear image about the number of drops from this window. The congestion window size is reduced only by number of dropped packets. Experimental results showed an improved TCP performance improved by 50%.

In [11], Yu presents two mechanisms: early packet loss notification (EPLN) and best-effort ACK delivery (BEAD). In EPLN, nodes in the path of the TCP packet notify the sender about losses so that TCP can retransmit lost packets earlier. In the second mechanism, BEAD, ACKs for lost ACKs are retransmitted in a best-effort way. Compared with TCP-ELFN, simulation results showed that EPLN and BEAD not only significantly improve TCP throughput but also considerably reduce TCP timeouts.

3. Boosting TCP with Cross Layer Information

In order to enhance TCP, we propose to provide TCP sender with the following information:

- Maximum probability of dropping a packet.
- Signal Strength of the mobile node.
- Explicit Notification of a hand over.

These information is piggybacked as options in the ACKpackets.

3.1 Maximum Probability of Dropping a Packet

Upon reception of a packet, an intermediate node (router) calculates the probability to drop the packet according to a specific congestion control scheme (RED, DSRED, etc). Based on this calculated probability the packet is marked for queuing or discarding. In our scheme, the intermediate node updates the field that stores the maximum probability to drop the packet if the value of this field is lower than the calculated probability.

3.2 Signal Strength

The signal strength is available with each received packet (pre-pended to the frame as a control header by the firmware). This information can be used by the sender to infer the error rate of the link at the receiver's vicinity.

3.3 Explicit Notification of a hand over

Before a mobile node enters the hand-over phase, it sends a notification to the sender. Using this information, the sender can freeze its state until a new notification is received from the mobile host.

Combining these three crucial information (maximum probability to drop a packet, signal strength and hand-over information), TCP protocol can efficiently estimate the rate of the communication (Congestion Window) and the timers (RTO) associated with it.

We studied the performance of TCP protocol when the maximum probability to drop a packet is sent back to the receiver. The algorithm to react to a packet loss is illustrated in Figure 1

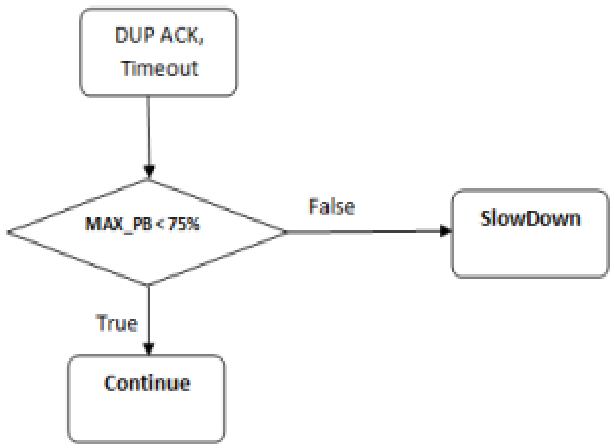

Figure 1. Modified TCP Sender Slowdown Algorithm

Upon reception of a duplicate ACK packet, the TCP sender checks the value of maximum probability to drop a packet that was collected by the transmitted packet in its way to the destination. If this value is higher than a certain threshold, 75% in our experimental study, it will trigger the slowdown algorithm. If the maximum value is lower than the threshold, the TCP sender assumes that the packet loss is due to link errors, and therefore there is no need to trigger its exponential back-off algorithm.

4. Experimental Analysis

The simulation environment is illustrated in Figure 2.

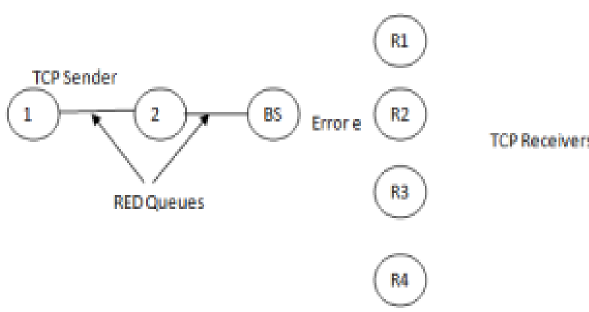

Figure 2. Simulated Network

We simulated the network with two TCP versions: Tahoe and Improved Version based on the maximum probability to drop a packet.

We introduced gradually different error rates and measured the achieved throughput. Table 1 and 2 show the values of throughput for different error rates. We note that we repeated the measurements 5 times with different random seed values.

Error Rate	#1 MB/S	#2 MB/S	#3 MB/S	#4 MB/S	#5 MB/S	Average KB/S
0	0.139533	0.139556	0.139549	0.139487	0.13951	139.527
0.05	0.115377	0.113134	0.110248	0.113372	0.114985	113.4232
0.1	0.089711	0.091408	0.090495	0.089077	0.09056	90.2502
0.15	0.06724	0.068502	0.068013	0.06814	0.069651	68.3092
0.2	0.047957	0.049662	0.049529	0.049673	0.049318	49.2278
0.25	0.034948	0.033567	0.033915	0.034658	0.033835	34.1846
0.3	0.019559	0.022738	0.023624	0.023634	0.022423	22.3956
0.35	0.014775	0.013932	0.011675	0.014183	0.011904	13.2938
0.4	0.007093	0.00451	0.006048	0.002846	0.006911	5.4816
0.45	0.001205	0.000689	0.000453	0.001318	0.001214	0.9758
0.5	0.000184	0.000307	0.000126	0.000145	0.000184	0.1892
0.55	0.000061	0.000022	0.000037	0.000057	0.000046	0.0446
0.6	0.000006	0.000025	0.000015	0.000015	0.000013	0.0148

Table 1. Achieved TCP Throughput (Original TCP Tahoe)

Error Rate	#1 MB/S	#2 MB/S	#3 MB/S	#4 MB/S	#5 MB/S	Average KB/S
0	0.140171	0.14017	0.140158	0.140154	0.140172	140.165
0.05	0.115141	0.11365	0.115228	0.114312	0.112121	114.0904
0.1	0.088512	0.089134	0.088867	0.087875	0.08913	88.7036
0.15	0.066948	0.066674	0.066253	0.066285	0.065654	66.3628
0.2	0.048243	0.045478	0.04973	0.043628	0.049261	47.268
0.25	0.034402	0.033279	0.03398	0.031768	0.035818	33.8494
0.3	0.018343	0.02155	0.020715	0.022068	0.021633	20.8618
0.35	0.012799	0.010292	0.016388	0.012323	0.012182	12.7968
0.4	0.004282	0.004414	0.005243	0.005931	0.004832	4.9404
0.45	0.000802	0.002182	0.00206	0.001633	0.001575	1.6504
0.5	0.000252	0.001002	0.000311	0.000747	0.000626	0.5876
0.55	0.000356	0.000378	0.000252	0.000047	0.000102	0.227
0.6	0.000092	0.000078	0.00002	0.000142	0.000014	0.0692

Table 2. Achieved TCP Throughput (Boosted TCP Tahoe)

As illustrated in figure 3, the performance of the original TCP and Boosted TCP are comparable for small error rates. This can be explained by the fact that channel errors are not main source of packet losses. However, when the channel error rate increases, the gain in performance is considerable. According to the values, the performance is improved by more than 5 times.

5. Conclusion and Future work

In this paper, we analyzed the impact of cross layer information on the performance of TCP protocol. We observed that by simply appending the maximum probability to drop a packet calculated at the level of each router and sending it back to the source, the TCP performance can be increased by 500% for highly lossy wireless links.

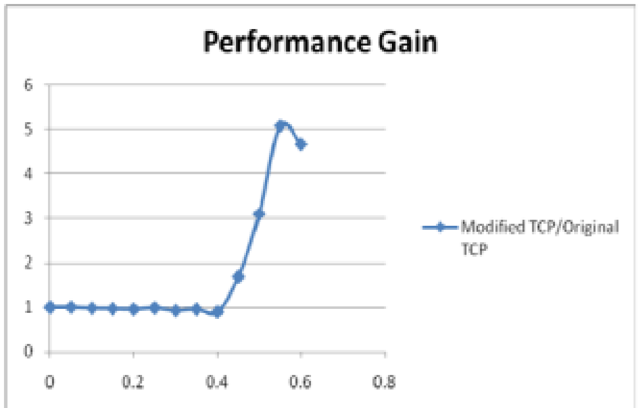

Figure 3. Performance Gain of Boosted TCP

As future work, we are interested to study the impact of the remaining two cross layer information on the performance of TCP protocol.

References

[1] Alnuem, M., Mellor, J., Fretwell, R., (2009) New algorithm to control TCP behavior over lossy links, International Conference on Advanced Computer Control, ICACC '09, 22-24 January.

[2] Balakrishnan, H., Padmanabhan, V., Seshan, S., Katz, R., (1996) A comparison of mechanisms for improving TCP performance over wireless links, in ACM SIGCOMM, Stanford, August.

[3] Chandran, K., Raghunathan, S., Venkatesan, S., Prakash, R., (1998) A feedback based scheme for improving TCP performance in ad-hoc wireless networks, *In*: Proceedings of International Conference on Distributed Computing Systems, Amsterdam.

[4] Chen, B., Ivan Marsic, Shao, H., Miller, R., Improved Delayed ACK for TCP over Multi-hop Wireless Networks, *In*: Proceedings of the 2009 IEEE conference on Wireless Communications & Networking Conference.

[5] Dyer, T. D., Bopanna, R. (2001). A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks, *In*: Proceedings of ACM MOBIHOC 2001, Long Beach, CA, Oct.

[6] Floyd, S. (1994). Tcp and explicit congestion notification, *ACM Computer Communication Review*, 24.

[7] Holland, Gavin., Vaidya, Nitin (1999) Analysis of TCP Performance over Mobile Ad Hoc Networks, *In*: Proceedings of ACM MOBICOM, Seattle, WA, Aug.

[8] Fu, Z., Meng, X., Lu, S., (2002). How ba TCP can perform in mobile ad hoc networks. *In*: Proceedings of 7th IEEE ISCC.

[9] Sundaresan, K., Anantharaman, V., Hsieh, H., Sivakumar, R. (2005). ATP: A Reliable Transport Protocol for Ad Hoc Networks, IEEE Transactions on Mobile Computing, November/December (vol. 4 no. 6)

[10] Xiao, L., Li, Z., Zhao, N. (2009). A TCP Performance Improved mechanism for Wireless Network, Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, Brisbane, Australia,July.

[11] Yu, X. (2004) Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-Layer Information Awareness, Proceedings of the 10th annual international conference on Mobile computing and networking.