Journal of Communications Technology, Electronics and Computer Science, Issue 4, 2016
ISSN 2457-905X

Techniques for Throughput Enhancement in Wireless Networks

A. M. Miyim
Department of Computer Science
Federal University Dutse (FUD)
Dutse Jigawa State, Nigeria
Abubakarm.miyim@gmail.com

***Abstract*— In wireless systems, unnecessary signal degradation is caused by spurious timeouts from highly variable round-trip times (RTTs) for Transmission Control Protocol (TCP). This paper proposes two methods of improving TCP throughput in wireless networks. Selecting a threshold higher than standard for retransmission timeout (RTO) and use of select-N-repeat (SNR) and trace-N-retransmit (TNR) as retransmission policies upon packet timeout are the two techniques adopted in this study. Simulation reveals that the former method helps reduce timeouts and provides relative throughput gain in a simulated network environment. However in the later method, TNR was found to improve throughput gain by about 8% over the SNR policy when packets at the receiver end arrive out-of-order.**

Keywords*— Throughput; Performance; Timeout; TCP; wireless networks.

I. Introduction

The Internet has tremendously benefited from the use of Transmission Control Protocol (TCP) [1] as a reliable and widely used data transport protocol today that establishes connection between the two ends (transmitter and receiver). A timeout mechanism starts from the transmitter end with the round-trip times (RTTs) being constantly monitored to determine the appropriate timeout period when sending a packet to the receiver. An explicit or implicit acknowledgement at the transmitter is received by each packet, the failure of which the packet is considered to be lost and therefore requires retransmission. In order to regulate the traffic flow (from transmitter to receiver), a congestion window is used by the TCP protocol to dynamically adjust the window size.

Although TCP was in the initial stage designed for wired networks, the tremendous increase in popularity of wireless data applications has inspired 4G (WiMAX and LTE) wireless networks to extend TCP to cover wireless communications as well. The objective of the TCP is to effectively utilize the available bandwidth of a network as well as avoid congesting the network (reason for packet losses) by controlling the transmission rates from senders. This has made TCP performance in wireless networks to be unsatisfactory because of the fluctuation in the quality of the radio link due to channel fading and user mobility that causes high variability of transmission time and delay of which requires improvement [2]. Data packet retransmissions and system scheduling causes signal delay at the link level due to terminals with relatively poor radio quality. Further delays have easily incurred during the handover process of resources from one network or cell to another. To transport data by employing TCP results in high RTTs and spurious delay timeouts. Regardless of the causes of timeout, the TCP congestion window is reduced to unity, thereby causing unnecessarily throughput degradation.

This paper proposed the study of two efficient methods of improving TCP throughput in wireless networks. One of the techniques relies on selecting a threshold for the retransmission timeout (RTO) a little bit higher than the usual value which results in producing some throughput gain. The other technique is the adoption of select-N-repeat (SNR) and trace-N-retransmit (TNR) methods acting as retransmission policies when packet timeout. TNR was found to improve the network throughput by about 8% over the SR policy from RTT measurements.

The rest of this paper is organized as follows: Section II covers related works in this area; Section III describes throughput improvements using the method of increased timeout threshold; Section IV discusses a proposed second method of trace-N-retransmit as a strategy to improve throughput performance and compare the relative performance of trace-N-retransmit and select-N-repeat. The results of performance study of the two methods are analyzed and discussed in section V to show the merits of the proposed methods; Section VI concludes the paper with directions for future improvements.

II. Related works

The performance of TCP in wireless networks has been studied extensively by many researchers where the effects of delay were identified as a shortcoming. In [3], Amnai et al., observes how delay variability influences on the performance of TCP. Performance degradation due to delay spikes for various types of TCPs has been studied in [4] by Chakravorty et al. According to a survey of these proposals [5], different solutions have been proposed to overcome the TCP deficiency for radio links. There are two ways of improving TCP performance in wireless networks; technique that requires

change of the existing protocol and the others that oppose change in TCP protocol. Fu et al., [6] in their work propose TCP Eifel for detecting viral timeouts and retransmissions using time stamping. Further usage of Eifel method to enhance TCP has been proposed by Gurtov and Ludwig in [7], in order to avoid spurious timeouts.

A proxy is adopted as a common point by using split-TCP [8] as a solution in which the end-to-end TCP connection is used to divide the terminal from the corresponding host as two separate, independent connections. The idea is to reduce the intensity of packet errors and delay on the wireless link as compared to that of wired connection do not suffer channel quality fluctuation as a result of TCP congestion control, timeout and retransmission mechanisms in the wired link.

For further enhancement, delay-jitter algorithm has been proposed and studied in [9] and [10]. With clear knowledge that delay causes spurious timeouts, one way of maintaining throughput performance is to absorb the delay variability by a large timeout value. This method (delay-jitter algorithm) is adopted in order to raise the TCP timeout value by arbitrarily injecting additional delay to some of the packets so as to increase the mean deviation of the RTTs as this method does not require any TCP stack modification. It is difficult to point out one single universal technique suitable for a variety of network and application environments. The split-TCP solution in particular, is likely to violate the security protection between transmitter and receiver. Furthermore, TCP performance for the connection between the terminal and the may not be satisfactory and therefore requires more improvement. Finally the delay-jitter algorithm requires appropriate selections of control parameters to minimize the negative performance impacts due to the increase of RTTs.

III. Improved Timeout Threshold

We start the discussion of RTO value selection in standard TCP operations [1], after which the new enhancement is then presented through an increased RTO value. Packets from the RTTs are closely being monitored by TCP sender. A timeout mechanism is utilized to trigger retransmissions of packets once the ACK is not received prior to expiry of time. However, with factor *m* set to 4, TCP sender uses is the sum of the average RTT and factor *m* multiplied by the mean deviation of RTTs as the RTO value [1] for incoming packets. Let $\aleph(k)$ be the *k*-th value of RTT, with its value as the time interval from the beginning of packet transmission until an ACK for the packet is received by the sender. With $\Re(k)$ as the smoothed average RTT, its value could be calculated from:

$$\Re(k)=(1-\delta)\Re(k-1)+\delta\aleph(k) \quad (1)$$

where δ represents the exponential smoothing parameter of typical value 1/8. Similarly, $\Im(k)$ denote the smoothed mean deviation of the RTTs and is computed as:

$$\Im(k)=(1-\gamma)\Im(k-1)+\gamma|\ \aleph(k)-\Re(k)\ | \quad (2)$$

where γ is the smoothing parameter, with typical value of ¼. Finally, the retransmission timeout (RTO) value is obtained from equation 3 as:

$$R(k)=\Re(k)+m\Im(k) \quad (3)$$

where *m* is the standard with its value set to 4.

According to TCP protocol standard, the sender keeps record of the starting time of forwarding packets to the receiver until the sender eventually receives an ACK associated with the packet before computing the RTT for the packet. Equations 1 through 3 are used to compute the average and mean deviation of RTTs as well as the new RTO value respectively. The RTO value is used for setting the timeout period for the next packets sent by the TCP sender, until the next RTT measurement is obtained and the RTO is updated according to the equations. It should be noted that until the time-stamping option is activated, a TCP sender does not track RTT for every packet, instead, only one packet among the outstanding packets is tracked for RTT at any time. As a result, equations 1 through 3 are invoked to update the RTO value when the ACK for a tracked packet is received by the sender.

By setting *m* to 4 in equation 3 in trying to determine the RTO value, almost all spurious timeouts could be avoided with a probability of nearly 1, so long as the RTTs doesn't show a high level of variability as is seen with wired networks. However, because of ARQ retransmissions, handovers, packet scheduling, channel degradation and so forth, RTTs in wireless networks tend to have much higher variability than their counterparts in wired networks. As a result of this, m value larger than 4 is proposed to compute the RTO threshold value.

Any increase of *m* in equation 3 is reciprocated by an increase of RTO value for the given smoothed average and mean deviation of RTTs. In wireless networks, it is evident that a bigger part of the RTO value helps maintain high variability of RTTs leading to curbing unnecessary spurious timeouts and maintain throughput performance. However, on no account should the RTO value be arbitrarily chosen in order to facilitate speedy recovery of actual packet losses. Furthermore, exists a tradeoff between throughput gain where spurious timeouts are being avoided and throughput degradation due to delay in recovering the actual packet losses. While specifics of network conditions and parameters determines the tradeoff, the performance study of the new method proves increasing *m* from 4 to 10 results in a better throughput gain for a given packet loss probability and remains relatively constant over a large range of *m* value.

Although spurious timeouts could be avoided in wireless networks by raising the RTO value, this newly enhanced method of throughput performance has not been presented elsewhere before now for the simple reason that the usual method of computing the RTO value (Equation 1 through Equation 3) has been widely implemented. Therefore, it is

difficult to modify the value of *m* on existing systems and devices. The new method demonstrates practicability with the existing networks on downlink transmissions because calculations for changing the RTO are executed only on the proxy, without modifying the TCP stack. Furthermore, the TCP sender at the proxy server can be modified once the need arises for split-TCP solution in a network by increasing the *m* value. More so the objective of both methods is to avoid spurious timeouts as a solution to sustaining throughput performance, the jitter-algorithm [9, 10] achieves an increased RTO value through injection of extra delay to RTTs as a way of increasing the mean deviation (Equation 3), while the increase of smoothed average RTT is kept minimal. Although the proposed method in this paper is to increase the *m* value in equation 3 and thereby achieve the increase in RTO. Nevertheless, this proposed technique could be seen as a future solution of TCP implementation for wireless applications.

One of the advantages of the jitter algorithm is that it does not require any change to the TCP entity. However, the introduction of additional delay to the communication path invariably increases the average RTT, while the increased mean deviation for RTT helps greatly in reducing the timeout frequency. Thus, a tradeoff from throughput losses as a result of increased RTT and the gain due to avoidance of timeouts bring about the performance gain for the delay-jitter algorithm. Such tradeoff is not applicable to this new method of increasing the *m* value, but instead, the proposed method provides throughput gain so long the *m* value is not excessively large. The modification of the entire TCP entity poses as the disadvantage of the new method.

IV. Trace-N-Retransmit mechanism

Known as a retransmission policy, TCP adopts *select-N-repeat* (SNR) when a timeout is considered [1, 11]. It means that TCP sender retransmits only the timeout of an outstanding packet that occurs to the receiver. Another well understood policy is the *trace-N-retransmit* (TNR) policy [11]. Under the TNR policy, when a timeout occurs, the TCP sender resends all unacknowledged packets, including even those packets that caused the timeout. Considering some factors like packet loss characteristics and bandwidth-delay, either of these policies (SNR or TNR) might produce a better throughput performance. In the case of TNR policy, even if the loss involves only one single packet, all outstanding packets are retransmitted, thereby incurring additional delay and bandwidth wastage. This is not the case with SR policy which usually performs better than the TNR policy.

In another scenario, consider the quality of a radio link to be a correlation between moderate to strong in such a way that, packets, encounter long delay whenever a packet requires a large number of transmissions prior to a successful reception due to unstable link quality in which successive packets are said to be correlated. Whenever packets are timed, it is likely that a significant TCP throughput degradation will occur. Additionally, it is assumed that packets change routes when they are resend by the TCP transmitter. This phenomenon will incur an additional delay different from the initial delay with the possibility of arriving earlier (out-of-order) than the original packet at the receiver end. Besides, handover connections in wireless networks can equally result in out-of-order arrivals at the destination. This is the motivation for adopting the TNR protocol as the retransmission policy for TCP in environments with correlated packet delays and early arrival of later packets. The use of TNR policy enables TCP sender to reset all timeout periods and resends all awaiting packets as soon as unacknowledged packet times out. Consequently, the rest of TCP operations (congestion window, acknowledgements (ACK), retransmissions and so forth) remain unchanged. Contrary to SNR policy, this is one of the network scenarios under which the TNR policy tends to show better performance.

V. Performance study

Two sets of RTT measurements are proposed in this study and are referred to as Traces 1 and 2. While a set of measured RTTs is given in Trace 1 where user equipment (UE) issues a repetitive 1.5 Kbyte ping messages to the nearest router in a 4G network, for Trace 2, the RTTs are downlink simulations of multiple wireless channel users. Apart from the distance-based path loss, Jakes model has been adopted to capture the channel fluctuations as a result of shadowing and fast fading. Proportional fairness algorithm (PFA) by definition is the scheduling task used in traditional networks in which users are directly proportional to the rate they can achieve and inversely proportional to the amount of data they have already transferred. The aim is to allow users with better propagation conditions to transmit more data, without starving others (some fairness guaranteed). Congestion control procedures such as acknowledgment and packet timeout are implemented by TCP simulator so as to trace every RTT and quantify the number of timeouts and throughput for 1 Mbyte file transfers. With the factor m considered as an input parameter, the average and mean deviation of RTTs is produced by the simulator using an exponential-smoothing method [1] as seen in the equations (1) through (3). It should be noted that out-of-order arrival of the packet is associated with Trace 1, while guaranteeing in-sequence delivery of packets and their retransmitted copies are represented with Trace 2. This could be as a result of first-come, first-serve nature of the scheduling algorithm (for packets of the same user).

As the factor *m* and the RTO threshold increases, the number of timeout occurrences decreases as shown by the simulations of Figure 1. Furthermore, the SNR policy has proven that the TCP throughput shows some improvements as shown in Fig. 1 for Trace 1. As the factor *m* increases, a corresponding increase in throughput is expected for a given loss probability P_L with the throughput reaching a peak at the factor value of 10 to 15. It is interesting to note that the throughput remains relatively flat over a wide range of values

for the factor *m*. By raising the factor *m*, the throughput is likely to improve from 27.0 to 30.7 Kbps, without packet loss. This represents a relative gain of 13.7% as depicted in the figure with similar gains to be achieved for P_L between 5% and 10%.

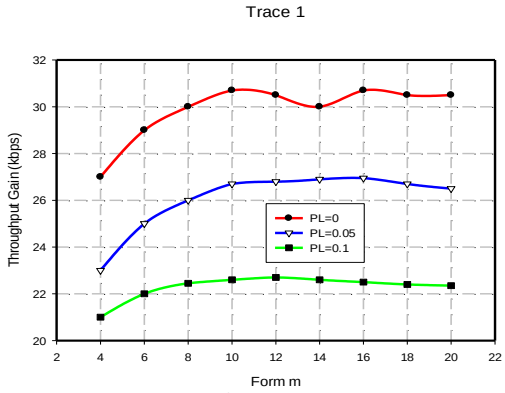

Fig. 1. TCP Throughput vs. Factor *m* for Trace 1.

Correspondingly, in Fig. 2, about 10% throughput improvement was obtained for Trace B. With the congestion window for Trace 2 being larger than that of Trace 1, the throughput for P_L=0 is having the least value and for all considered values of *m*, the throughput is uniformly increased at P_L=5%. However, retransmission timeouts are triggered due to received duplicate ACKs by the TCP sender. Upon receiving the duplicate ACKs, the congestion window reduces by half, and not reset to unity (1) as experienced under timeout occurrence. Throughput is further reduced, the moment packet loss probability P_L is increased (P_L=10%). These results confirm the effectiveness of using the RTO value higher than the standard value of 4 (10 instead of 4 as the factor *m*) that helps improve TCP performance in wireless networks. Furthermore, duplicate ACKs could be triggered by dropping packets deliberately to avoid spurious timeouts.

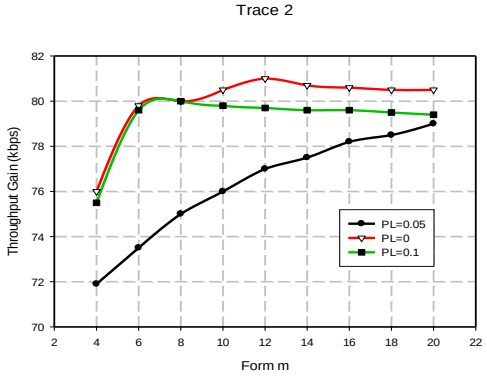

Fig. 2. TCP Throughput vs. Factor *m* for Trace 2.

When the factor *m* is less than 9, the TNR retransmission policy yields a throughput, higher than the SNR policy for Trace 1 without packet loss as shown in Fig. 3. For the simple reason that the RTTs in Trace 1 are moderately correlated (i.e., an autocorrelation coefficient of 0.42), the subsequently retransmitted packets might timeout. Since TNR retransmits packets starting from the timeout packet, timeouts for the subsequent packets are likely to be avoided.

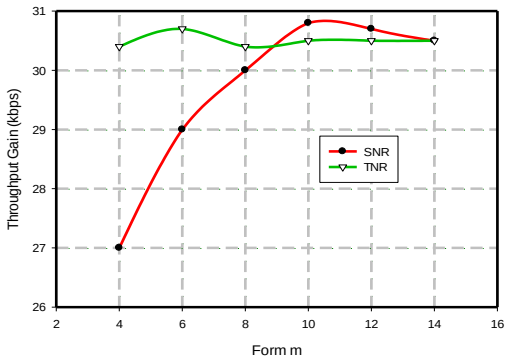

Fig. 3. Throughput Gain for the TNR over SNR policy

Additionally, knowing fully well that packets do arrive out-of-order for Trace 1, the retransmitted packets reach the receiver earlier than the originally transmitted packets (as long as they are not lost), thereby maintaining data throughput. In Fig. 3, TNR has an edge of about 8% throughput gain over the SNR policy for the factor *m* of 4. The two policies seem to achieve similar throughputs when avoiding spurious timeouts with the factor *m* exceeding 9. Due to the first-come-first-serve nature of the PFA scheduler in Trace 2, packets arrive at the receiver end in order because the TNR policy was applied. Generally, the SNR policy shows some higher throughput than the TNR policy for Trace 2 which reveals the strength of TNR policy in providing clear throughput gain over the SNR policy.

VI. Conclusion

The paper did propose, investigated and discussed the TCP performance using two techniques to avoid or minimize the impact of spurious timeouts. The first technique increases the RTO threshold through an appropriate choice of factor *m*. The second method, however, deals with choice of retransmission policy that was based on the characteristics of the underlying network (temporal correlations and in-sequence delivery of packets). The results show significant performance gains for both policies from simulated RTT traces in wireless networks.

Acknowledgment

The author would like to thank the Federal University Dutse (FUD) for the continued support of this research. Also acknowledged are the reviewers for their useful comments and advices in improving the quality of this paper.

References

[1] D. Gupta and A. K. Sharma, "On Performance Evaluation of WSN Routing Protocols for MICA and MICAz using Different Radio

Models", International Journal of Energy, Information and Communications, , 2(4): 181-194, November 2011.

[2] X. Wang & Q. Liang, "On the Throughput Capacity and Performance Analysis of Hybrid Wireless Networks over Fading Channels", IEEE Transactions on Wireless Communications, 2013, 12(6):2930 - 2940.

[3] M. Amnai, Y. Fakhri, J. Abouchabaka, "Impact of Mobility on Delay-Throughput Performance in Multi-Service Mobile Ad-Hoc Networks," Int'l Journal of Communications, Network and System Sciences, 2011, (4): 395-402.

[4] R. Chakravorty, S. Katti, J. Crowcroft and I. Pratt, "Flow Aggregation for Enhanced TCP over wide-area wireless", submitted to IEEE INFOCOM 2003, source: http://www.cl.cam.ac.uk/users/rc277/gprs.htmlS.

[5] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," Proc. 1st ACM Int'l Conf. on Mobile Computing and Networking (Mobicom), pp. xxx-xxx, November 1995.

[6] Fu, et. al., "Effect of Delay Spike on SCTP, TCP Reno and Eifel in a Wireless Mobile Environment," Proc. of International Conf. on Computer Communications and Networks, pp. 575-578, Oct. 2002.

[7] A. Gurtov and R. Ludwig, "Responding to Spurious Timeouts in TCP," in Proc. of IEEE Infocom, Mar. 2003.

[8] F. Xie, J. L. Hammond and D. L. Noneaker, "Evaluation of a Split-Connection Mobile Transport Protocol," Wireless Networks, Vol. 9, 2003, pp. 593-603.

[9] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, "The Impact of Multi-hop Wireless Channel on TCP Throughput and Loss," Proceedings of IEEE INFOCOM 2003, pp. xx-xx

[10] T. E. Klein, et. al., "Avoiding Spurious TCP Timeouts in Wireless Networks by Delay Injection," submitted for publication, 2004.

[11] L. T. Nguyen, R. Beuran, Y. Shinoda, "Performance Analysis of IEEE 802.11 in Multi-hop Wireless Networks", Proceedings of Third International Conference, MSN 2007 Beijing, China, December 12-14, pp326-337.