Transport Protocol Mechanisms for Wireless Networking: A Review and Comparative Simulation Study

Alper Kanak, Öznur Özkasap

Koç University, Department of Computer Engineering, Istanbul, Turkey
{akanak, oozkasap}@ku.edu.tr

Abstract. Increasing popularity of wireless services has triggered the need for efficient wireless transport mechanisms. TCP, being the reliable transport level protocol widely used in wired network world, was not designed with heterogeneity in mind. The problem with the adaptation of TCP to the evolving wireless settings is because of the assumption that packet loss and unusual delays are mainly caused by congestion. TCP originally assumes that packet loss is very small. On the other hand, wireless links often suffer from high bit error rates and broken connectivity due to handoffs. A range of schemes, namely end-to-end, split-connection and link-layer protocols, has been proposed to improve the performance of transport mechanisms, in particular TCP, on wireless settings. In this study, we examine these mechanisms for wireless transport, and discuss our comparative simulation results of end-to-end TCP versions (Tahoe, Reno, NewReno and SACK) in various network settings including wireless LANs and wired-cum-wireless scenarios.
Keywords. TCP, wireless transport protocols, end-to-end protocols, split-connection protocols, wired-cum-wireless.

1. Introduction

With the growth and increasing popularity of wireless services, need for efficient wireless connections to the existing network infrastructures will be become crucial in future internetworks. TCP, being the reliable transport level protocol widely used in wired network world, was not designed with heterogeneity in mind. As an example, when a data packet is lost, in a wired network it is relatively safe to assume that this is most likely due to congestion, that is, too many packets are contending for network resources. However, when a wireless link or sub-network is involved it could as well be because of bad reception at the location of the user.

It is not unusual today, and will certainly become increasingly more common in forthcoming years, that a given TCP connection would pass through networks with varying latency and bandwidth characteristics. For example, it may pass through a network with high latency/low bandwidth (e.g. a wireless WAN) and then through a low latency/high bandwidth (wired) network. The problem with the adaptation of TCP to the evolving wireless settings is because of the assumption that packet loss and unusual delays are mainly caused by congestion. TCP is thus tuned to adapt to such congestion losses by slowing down the amount of data it transmits. It shrinks its transmission window and backs off its retransmission timer, thus reducing the load and congestion on the network. However, in wireless networks, packet loss is often caused due to other factors besides congestion. Wireless channels often suffer from high bit error rates and broken connectivity due to handoffs. Moreover, the Bit Error Rate (BER) may vary continuously during a session. Unfortunately, TCP assumes that these losses are due to congestion and invokes its congestion control algorithms. Consequently, effects of packet losses in wireless links on TCP performance and enhancements to the classical TCP versions are significant research topics to study. Based on the studies in this area, a question to be answered would be “Would TCP, in particular, be the transport layer of preference for a wired-cum-wireless world?”

A range of schemes has been proposed to improve the performance of transport mechanisms, in particular TCP, on wireless settings. They can be broadly categorized into three groups: end-to-end protocols, split-connection protocols and link-layer protocols. These approaches are revisited in the next section. In this study, we examine mechanisms for wireless transport, and conduct simulations and analysis of end-to-end TCP versions (Tahoe, Reno, NewReno and SACK) for various network settings including wireless LANs and wired-cum-wireless scenarios. Tahoe is an old TCP protocol that is rarely used currently. It implements slow start algorithm and has no fast recovery algorithm. In TCP Reno, the fast retransmit operation has been modified to include fast recovery. It prevents the communication path from going empty after fast retransmit, thereby avoiding the need to slow start to refill it after a single packet loss. Reno significantly improves the behavior of Tahoe when a single packet is dropped from a window of data, but can suffer from performance problems when multiple packets are dropped from a window of data. The NewReno TCP includes a small change to Reno algorithm at the sender that eliminates Reno’s wait for a retransmit timer when multiple packets are lost from a window. NewReno can recover without a retransmission timeout. The TCP SACK uses the TCP Extensions for high performance. The congestion control algorithms are a conservative extension of Reno’s congestion control. SACK differs from Reno when multiple packets dropped from a window of data. During fast recovery, SACK maintains a variable called pipe that represents the estimated number of packets outstanding in the path. The sender only sends new or retransmitted data when the estimated number of packets in the path is less than the congestion window [1].

The outline of the paper is as follows. In Section 2, we review mechanisms for wireless transport, namely end-to-end protocols, split-connection protocols and link-layer protocols. Section 3 gives details of our simulation study and analysis results on various network settings. Section 3.1 discusses comparisons of TCP performances over wired and wireless LAN scenarios. Section 3.2 and 3.3 discuss results for end-to-end protocols and split-connection protocols in wired-cum-wireless scenarios respectively. Finally, our conclusions and future work are stated in Section 4.

2. Review of Mechanisms for Wireless Transport

In contrast to wired LANs, a wireless LAN exhibits typical characteristics, such as low and highly variable throughput, high and highly variable latency, bursty and random packet losses unrelated to congestion, irregular blackouts, and asymmetric uplink and downlink channels. Under such conditions, when packets are lost due to errors other than congestion, this leads to unnecessary reduction in end-to-end throughput and hence degraded performance. In order to improve the transport protocol performance, in particular TCP, in wireless settings, two methods are common in systems experiencing packet loss:

- ⊙ Hiding any non-congestion-related losses from the TCP sender which therefore requires no changes to existing sender implementations.
- ⊙ Attempt to make the sender aware of existence of the wireless hops and realize that some packets losses are not due to congestion

For implementing these methods, several schemes are proposed which can be examined in three broad categories, namely, *end-to-end protocols* where sender is aware of the wireless link, *link layer protocols* that provide local reliability, and *split-connection scheme* where the connection is partitioned into two, wired and wireless, at the base station [2]. Next, we review these approaches.

2.1 End-to-end protocols

End-to-end protocols, in which the sender is aware of the wireless link, retain a single TCP connection from sender to receiver and they handle losses through the use of selective acknowledgements (SACKs). This allows the sender to recover from losses within a window without resorting to timeout. This scheme performs well with TCP SACK but also applicable for Tahoe and Reno protocols.

The common TCP implementation currently employed on the Internet is Reno TCP. Reno TCP algorithm uses a combination of slow-start, congestion avoidance, fast retransmit and fast recovery methods. It utilizes cumulative acknowledgments for providing reliable delivery. NewReno TCP is an extended version of Reno TCP which is proposed for improving performance after multiple packet losses. SACK protocols add selective acknowledgments allowing the sender to handle multiple losses within a window of outstanding data more efficiently. Another approach referred to as explicit loss notification (ELN) is used to prevent unnecessary throttles applied to congestion window.

2.2 Data Link Layer Protocols

Proposals in this category are based on the following argument: Since the problem of high link error rates in a wireless environment lies at the physical layer, the data link layer (LL) has more control over it than any other layer and can sense a problem faster. Hence, the problem can be solved at the LL that aims to offer a good communication channel to the layers above, in particular to TCP. This does not require any change on current TCP implementations. The way to do this is to ensure reliable delivery of packets at all times, for example using an Automatic Repeat reQuest (ARQ) scheme. However, studies have shown that this could lead to degraded performance. This is because TCP also attempts to ensure reliable transmission, thus leading to a lot of duplicate effort. If such a solution is to be accepted, it has to be tightly coupled with TCP.

The most prominent proposal in this category is the SNOOP protocol [3]. SNOOP lies in an intermediate position between the end-to-end and split-connection proposals, trying to utilize the ability of a LL protocol to respond fast, and at the same time using the available information to keep TCP “happy” with the existing network connection. For example, SNOOP would favor local LL retransmissions instead of TCP retransmissions. This is achievable because a SNOOP agent would be aware of duplicate acknowledgments traveling from the receiver to the sender and suppress them, locally re-transmitting the potentially lost segment instead. Without SNOOP, the receipt of a certain number of duplicate acknowledgments would have caused a TCP segment retransmission.

2.3 Split-connection protocols

The basic idea behind the proposals in this category, such as Indirect TCP (I-TCP) [4], is that since there exist two completely different classes of sub-networks, namely wired and wireless, each TCP connection could be split into two connections at the point where the two sub-networks meet. As an example, suppose a mobile user browses a conventional web site using his laptop. The TCP connection will be split into two: one between the mobile host and the base station and one between the base station and the web server. The advantage is that TCP implementation in the sender does not need to be modified to deal with the enhanced functionality required for the wireless hop. The best transport protocol for each type of network can be utilized. This is very similar to using a HTTP proxy server. Splitting the TCP connection is essentially the technique used under the Wireless Application Protocol (WAP) architecture. The most significant disadvantage of splitting TCP connections is the loss of end-to-end semantics of TCP. In addition, the performance may also be degraded by the fact that it could end up splitting a particular connection several times, if different combinations of sub-networks are involved. Handoffs are also not handled as efficiently, and crashes in the base station result in TCP connection termination.

3. Simulation Results

The underlying platform for our simulation model is ns-2 network simulator [5]. Wireless model in ns-2 supports the effectiveness of mobile nodes by different routing mechanisms, routing protocols, network components, movement/traffic scenario generation and transmission protocols. Among the ad-hoc routing protocols currently available for mobile networking in ns-2, we chose DSR (dynamic source routing) as the routing agent. The DSR agent checks every data packet for source-route information. It forwards the packet as per the routing information. In case it does not find routing information in the packet, it provides the source route if route is known, or caches the packet and sends out route queries if route to destination is not known. Routing queries, always triggered by a data packet with no route to its destination, are initially broadcast to all neighbors. Route replies are sent back either by intermediate nodes or the destination node, to the source, if it can find routing info for the destination in the route query. Simulation parameters common to all set of runs are frequency of 2.4GHz, 802.11b Mac layer, 10Mbps channel, 100m distance between neighbor nodes in the topology, RED queue management algorithm, and 1400 byte-packet size with sender's rate of 200 packets per second. Versions of TCP protocols simulated are Tahoe, Reno, NewReno and SACK.

3.1 Comparison of TCP performances over wired and wireless LAN scenarios

Scenario 1: Simple two-node connection: The first scenario consists of a pair of nodes communicating either in wireless or wired settings that is used to simulate wireless LAN without collision. This is because there is only a pair of nodes communicating, and no additional node exists which could affect the communication. Besides, note that there is an error model on the link which is described when discussing our simulation results. The second set is for the two nodes connected via a wired link.

The measurement metrics are throughput and queue delay. Queue delay measurement is accomplished as follows: Suppose that a packet p is to be sent. There are two events associated: E1 and E2. E1 is scheduled to occur when the current sending operation has completed and E2 represents the packet arrival event. Here E1 is associated with a *Queue* object of ns-2, and queue delay is the time difference between these two events. On the other hand, average queue delay is calculated by tracing a set of packets during a time period. RED is used as the queuing algorithm, and buffer size is varied by using ns-2 arguments while we kept the buffer sizes of both sender and receiver the same.

According to the throughput results in Fig. 1 where x-axis represents the number of lost packets in a congestion window, SACK performs better in terms of throughput among all TCP versions while Tahoe has the lowest throughput values. The reason is that SACK uses a selective acknowledge and pipe scheme to ensure that when packets are dropped, the congestion window will recover fast. Thus the average throughput will be improved. Now let's consider the effect of buffer size on delay and throughput. As shown in Fig. 2, when the buffer size increases, different TCP versions have almost the same performance on average queue delay. However, we observed that the distinction of throughput for different TCP versions when buffer size increases is relatively large (Fig.3). The NewReno and SACK performs better than Tahoe and Reno. The reason is that when buffer size is small, multiple packets will be dropped. Tahoe and Reno has no fast recovery algorithm for dealing with multiple drops. Thus the TCP congestion window drops rapidly, hence leading low throughput. We observed similar results in wired connection as well (because of the page limitation, we do not include these graphs).

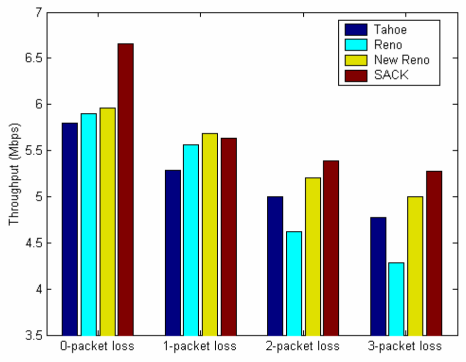

Fig. 1. Throughput for n-packet losses in simple wireless scenario

Scenario 2: Wireless N-node model: The second scenario is a wireless N-node model as illustrated in Fig. 4, which is used to simulate a wireless LAN with collisions. The network size N changes from 10 to 50. When the network size increases, the collisions become more frequent. Fig. 5 shows throughput analysis results as the network size increases for TCP versions. We observe that all four curves have the same trend of decrease when node number increases. The reason is that if node number increases, due to collisions taking place frequently, it becomes harder and harder to establish a connection link for each pair. However, SACK still presents the best performance among all. This is due to the fast recovery and fast retransmission algorithms implemented in SACK. Even when collisions occur and multiple packets are dropped, fast recovery algorithm could help TCP link to maintain a relatively good performance.

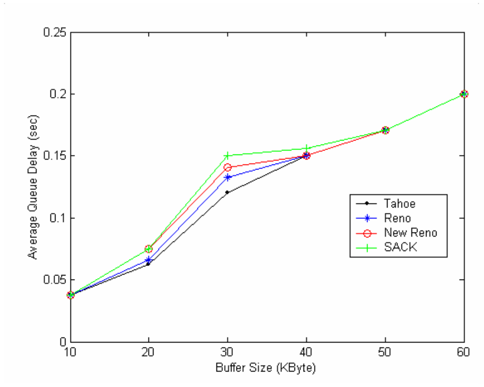

Fig. 2. Average queue delay as a function of buffer size in simple wireless scenario

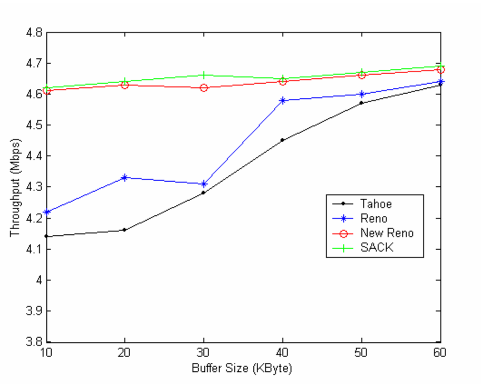

Fig. 3. Throughput as a function of buffer size in simple wireless scenario

3.2 End-to-End protocols: wired-cum-wireless scenario

TCP has been optimized over the years for wired networks. With the proliferation of wireless networking, TCP has to perform well in heterogeneous infrastructures, which exhibit different characteristics than the wired ones such as random segment corruption over wireless links. Several improvement proposals attempt to lessen TCP's inefficiencies in a wired-cum-wireless environment. In order to simulate the end-to-end schemes, we have constructed the topology shown in Fig. 6. Node 1 sends ftp packets to every node in the system without caring the link on the way to the destination. That could be either a wireless or wired link. End-to-end protocol maintains a single TCP connection from sender to receiver and attempts to make TCP aware of wireless losses so that it can deal with them. Protocols handle losses through the use of some form of selective acknowledgements (SACKs). As mentioned before, this allows the sender to recover from losses within a window without needing to timeout. Thus, end-to-end schemes involve modifying the TCP implementation in the sender to make it "wireless" aware. This may not always be feasible and is a potential problem for widespread implementation of end-to-end schemes.

Note that, in this scenario, a real base station is not constructed at node 3 but a non-mobile node is inserted to simulate the end-to-end protocol. In fact, there may be a base station to manage the communication among mobile nodes. End-to-end protocols do not worry about the communication path between the sender and the receiver even if there is a base station on the way. In other words, the mobile nodes in our simulation act as a base station. We construct a scenario with a base station for split-connection protocols, and evaluate combined performance results in the next section.

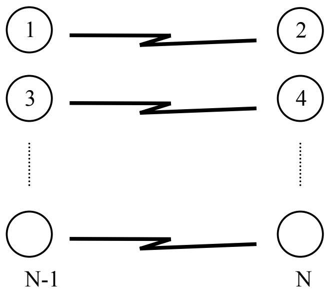

Fig. 4. Wireless N-node topology

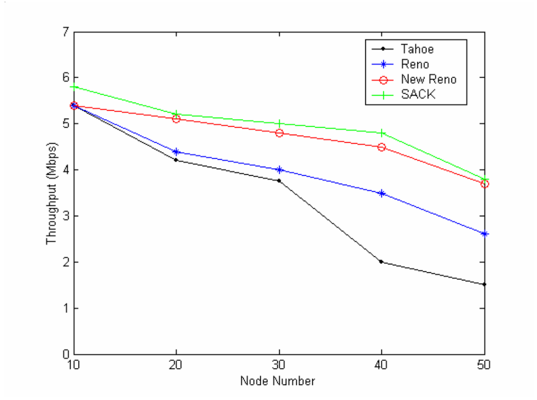

Fig. 5. Throughput for N nodes scenario

3.3 Split-Connection protocols: wired-cum-wireless scenario

In this scenario, we replace node 3 in Fig.6 with a base station to split the connection into wired and wireless parts. In other words, both parts are communicating via the base station. The nodes connected to each other via wired links are aware of the wireless link on the other end. Split-connection protocols hide the unreliability of the wireless link from the sender by terminating the TCP connection at the base station and using a separate protocol from the base station to the mobile host. This has the advantage that the TCP implementation in the sender does not need to be modified.

Throughput analysis results of end-to-end and split-connection cases for different error rates are given in Fig. 7. The rates represent the ratio of erroneous bits over the whole packet. For instance, 1/256 means there is 1 erroneous bit for every 256 bit in a packet. In all cases, SACK shows better performance in lossy wireless links. For high error rates, splitting schemes may perform better compared to end-to-end counterparts. Reno is the weakest protocol in all cases. As shown in Fig. 8, we observe that the bigger the queue size of the link, less congestion will occur and performance would be better. When queue size increases, a large proportion of total error is due to non-congestion losses. This is because of the fact that as the queue size on the link increases, the probability of congestion would decrease. Hence the throughput does not change much. Among all TCP versions simulated, SACK protocols perform better. Likewise, end-to-end case with TCP Reno shows degraded performance among the others.

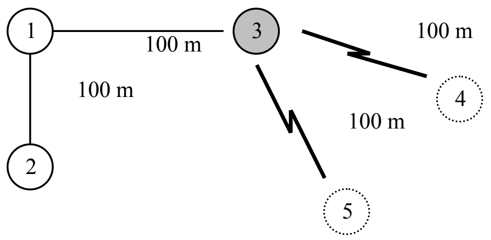

Fig. 6. Simulation topology for end-to-end and split connection protocols

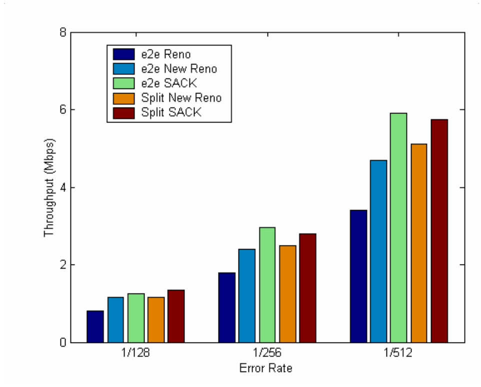

Fig.7. Throughput analysis of end-to-end and split protocols as a function of error rate

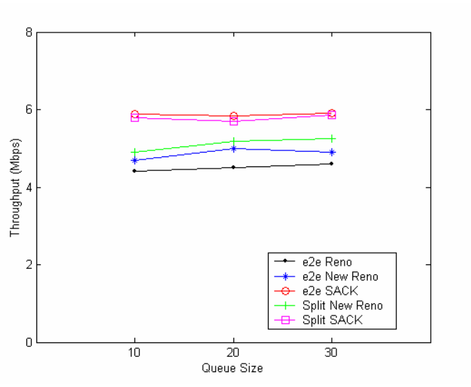

Fig. 8. Throughput analysis of end-to-end and split protocols as a function of queue size

4. Conclusions

In this study, we review three main approaches for transport mechanisms of wireless networking, namely end-to-end, split-connection and link-layer methods together with our comparative simulation results. Our simulation study yields some conclusions about the performance of end-to-end and split-connection transport protocol mechanisms particularly in wireless local area network settings. We conclude that, among the simulated TCP versions, TCP SACK has the best performance. When the buffer size increases, different TCP versions show almost the same trend on average queue delay, but the difference is apparent on average throughput behavior. In particular, for wireless multi-node model, throughputs of all TCP versions are inversely proportional to the node number. We have also simulated wired-cum-wireless scenarios for both end-to-end and split connection mechanisms and found out that TCP SACK performs best in both cases in terms of throughput versus different error rates and queue sizes. According to results, TCP SACK with end-to-end protocols performs a little better than the TCP SACK with split-connection protocols. An area for further study would be simulating and analyzing link-layer protocols and ELN schemes. There are also other solutions within this area such as WTCP (wireless TCP) or TCP-aware protocols that can be studied on the same network scenarios.

References

1. K. Fall and S. Floyd, Comparisons of Tahoe, Reno, and SACK TCP, ftp://ftp.ee.lbl.gov/papers/sack.ps.Z, December 1995.
2. H. Barakrishnan, V. N. Padmanabban, S. Seshan, R. H. Katz, A comparison of mechanisms for improving TCP performance over wireless links,Proc. *ACM SIGCOMM*, 1996.
3. H. Balakrishnan, S. Seshan, and R.H. Katz. Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. *ACM WirelessNetworks*, 1(4), December 1995.
4. 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.
5. K.Fall and K. Varadhan, NS Notes and Document. February 25, 2000.