A Microscopic Analysis of TCP Performance over Wireless Ad-hoc Networks

Vaidyanathan Anantharaman Raghupathy Sivakumar
School of Electrical and Computer Engineering, Georgia Institute of Technology
vaidy@ece.gatech.edu, siva@ece.gatech.edu

1. INTRODUCTION

Ad-hoc networks are multi-hop wireless networks that can operate without the services of an established backbone infrastructure. While such networks have obvious applications in the military and disaster relief environments, more recent works that have motivated their use even in regular wireless packet data networks have increased their significance. The focus of this paper is to study the performance of the TCP transport layer protocol over ad-hoc networks.

Recent works in transport protocols for ad-hoc networks have investigated the impact of ad-hoc network characteristics on TCP's performance, and proposed schemes that help TCP overcome the negative impact of such characteristics as random wireless loss and mobility. The primary mechanism proposed involves sending an explicit link failure notification (ELFN) to the source from the point of link failure. The source, upon receiving the ELFN *freezes* TCP's timers and state, re-computes a new route to the destination, and either releases the timers and state or re-starts them from their respective initial values. While the goal of ELFN based approaches is to prevent the route disruption time from adversely impacting TCP's performance, in this paper we contend that there are several other factors that influence TCP's performance degradation. We briefly outline the different factors below:

- *TCP Losses:* Every route failure induces upto a TCP-window worth of packet losses. While the losses have an absolute impact on the performance degradation, the TCP source also reacts to the losses by reducing the size of its window. Note that ELFN will prevent this negative impact on TCP's performance by appropriately freezing TCP's state.
- *MAC Failure Detection Time:* Since the MAC layer (802.11) has to go through multiple retransmissions before concluding link failure, there is a distinct component associated with the time taken to actually detect link failure since the occurrence of the failure. Importantly, the detection time increases with increasing load in the network. While an external mechanism to detect link failures (e.g. through periodic beacones at the routing layer) would solve this problem, it comes at the cost of beacon overheads and associated trade-offs.
- *MAC Packet Arrival:* When a failure is detected as described above, the link failure indication is sent only to the source of the packet that triggered the detection. If another source is using the same link in the path to its destination, the node upstream of the link failure will wait until it receives a packet from that source before informing it of the link failure. This also contributes to the magnitude of the delay after which a source realizes that a path is broken.
- *Route Computation Time:* Once a source is informed of a path failure, the time taken to recompute the route also increases with increasing load. With ELFN, for a load of 25 connections, the per-flow average of the aggregate time spent in route computation during a 100 second simulation was as high as 15 seconds. In addition to the absolute impact of the idle periods, TCP is also likely to experience timeouts, especially in the heavily loaded scenarios where the route computation time can be high.

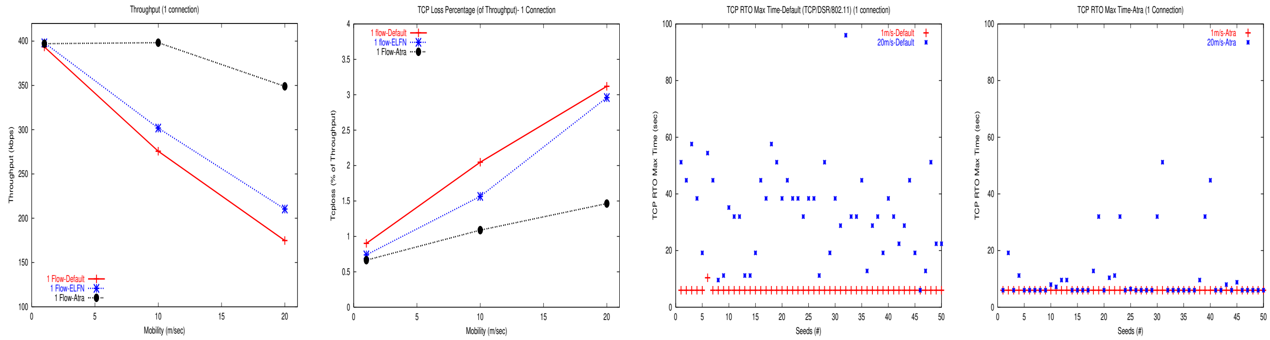
In the next section, we present a framework of mechanisms called *Atra* targeted toward addressing each of the above components. We show through representative simulation results that the proposed mechanisms outperform both the default protocol stack and an ELFN-enabled protocol stack substantially. We assume the default protocol stack to comprise of the IEEE 802.11 MAC protocol, the Dynamic Source Routing (DSR) routing protocol, and TCP-NewReno as the transport layer protocol. For a more detailed analysis of TCP performance in mobile ad-hoc networks, and description of the Atra framework, please see [1][1].

2. THE ATRA FRAMEWORK

The *Atra* framework consists of the following three mechanisms targeted towards addressing the components identified in Section 1:

- *Symmetric Route Pinning (SRP):* The DSR routing protocol, by default, does not require the use of symmetric routes between the source and the destination, i.e. the route taken from the source to the destination can be different from that taken from the destination to the source. Since, TCP relies on the correct delivery of acknowledgment (ACK) packets

[1]This work was supported in part by funds from the Yamacraw Research Program



(a) TCP Throughput (b) TCP Loss % (c) TCP Max RTO - Default (d) TCP Max RTO - Atra

Figure 1: Atra vs. ELFN vs. Default (1 Connection)

for the smooth progress of a connection, in a dynamic environment such as mobile ad-hoc networks, the use of asymmetric routes leads to an increased probability of a TCP connection suffering from a stall due to link failures. The SRP mechanism, implemented at the DSR layer, ensures that the ACK path of a TCP connection is always kept the same as the data path. The route pinning is performed only for non-piggybacked ACKs. SRP thus reduces the probability of a connection stalling through route failures. While a plausible drawback of the above scheme is the increased contention experienced by the data traffic due to the reverse path ACK traffic, we found in our evaluations that in a loaded network, such an impact was minimal.

- *Route Failure Prediction (RFP):* In RFP a node predicts the occurrence of a link failure based on the progression of signal strengths of packet receptions from the concerned neighbor. Maintaining a history of signal strengths enables nodes to dynamically profile the speed at which neighbors are moving away by observing the slope of the progression. If the *look-ahead time* threshold to trigger a prediction is set to be sufficiently large enough to compute an alternate route before the current path fails, connections remain unaffected by mobility induced route failures. When a source receives a predicted route failure message it issues a new route request but continues to use the current path until either a new route is computed, or the current route fails and a normal route error is received. Route requests are suppressed based on the same thresholds used to predict route failures. Hence, a route that is close to failure will not be chosen during any route computation process.

- *Proactive Route Errors (PRE):* The third mechanism in *Atra* is targeted toward minimizing the latency involved in the propagation of link failure information to the sources that use the concerned link. In the default set-up, DSR issues a route error only to the source of the packet that triggered a link failure at the MAC layer, irrespective of the actual number of flows that traverse that link. If multiple flows from multiple sources are using the same link, a packet from each source will have to cause a failure at that link, before route errors are sent back that source. This increases the latency involved in sources being informed of the link failure. The latency is further inflated because subsequent packets, arriving from other sources, also have to go through the *MAC failure detection time* before the link failure is inferred. *Atra* uses a mechanism called PRE wherein each node maintains a cache of the source identifiers of TCP connections that have used a particular link in the past T seconds (T set to a 1 second in our simulations). When a link failure is detected, all sources that have used the link in the past T seconds are informed about the link failure through normal route errors. This reduces the latency involved in the route failure information delivery, which consequently triggers earlier alternate route computations and reduces the number of losses.

2.1 Performance Evaluation

In Figures 1(a)-1(d) we present a few representative comparison results for Atra, ELFN, and the default protocol stack. The scenario used consists of 100 nodes distributed in a 1000mx1000m grid and moving using the way-point mobility model. One TCP connection between a randomly picked source-destination pair is used. The results in Figures 1(a) and 1(b) are averaged over 50 simulation runs. From Figure 1(a), *it can be observed that* Atra *outperforms the default protocol stack and ELFN by a large margin*. *Atra* exhibits throughput improvement of around 100% over the default case at a mobility rate of 20 m/s. The improvement is around 70% when compared to the ELFN approach. The TCP loss rate experienced by the *Atra* framework shown in Figure 1(b), is significantly lower than that of both ELFN and the default protocol stack. The prevention of route disruptions is also evident from the TCP RTO values shown in Figures 1(c) and 1(d) for a particular scenario. While the RTO values for the default scenario is highly dispersed, the values for *Atra* predominantly remain around the default value of the TCP retransmission timer, even with an increase in mobility.

3. CONCLUSIONS

In this paper, we contend that *existing approaches to improve TCP performance over mobile ad-hoc networks have focused only on a subset of the factors* affecting TCP performance. We identify a comprehensive set of factors, and propose a framework called *Atra* to improve TCP performance. *Atra* is a suite of mechanisms at the MAC and routing layers, aimed at improving performance by (i) reducing the number of route failures, (ii) predicting route failures before they occur and (iii) minimizing the latency for route error propagation. We show that *Atra* improves on the throughput performance of a default protocol stack by 50-100%.

4. REFERENCES

[1] V. Anantharaman and R. Sivakumar, "Atra: A Framework for Improving TCP Performance over Ad Hoc Networks, Technical Report, GNAN Research Group - http://www.ece.gatech.edu/research/GNAN," March 2002.