A Feedback Based Scheme For Improving TCP Performance In Ad-Hoc Wireless Networks *

Kartik Chandran **Sudarshan Raghunathan** **S. Venkatesan**
Ravi Prakash
Computer Science Program
University of Texas at Dallas
Richardson, TX 75083-0688
e-mail: {chandran,rsud,venky,ravip}@utdallas.edu

Abstract

Ad-hoc networks are completely wireless networks of mobile hosts, in which the topology rapidly changes due to the movement of mobile hosts. This frequent topology may lead to sudden packet losses and delays. Transport protocols like TCP have been built mainly for reliable, fixed networks. Hence, when used in ad-hoc networks, TCP misinterprets this loss as congestion and invokes congestion control. This leads to unnecessary retransmissions and loss of throughput. To overcome this problem, a feedback scheme is proposed, so that the source can distinguish between route failure and network congestion. When a route is disrupted, the source is sent a Route Failure Notification(RFN) packet, allowing it to freeze its timers and stop sending packets. When the route is re-established, the source is informed through a Route Re-establishment Notification (RRN) packet, upon which it resumes by unfreezing timers and continuing packet transmissions. The simulated performance of TCP on ad-hoc networks with and without feedback is compared and reported. It is observed that in the event of route failures, as the route re-establishment time increases, the use of feedback provides significant gains in throughput as well as savings in unnecessary packet transmissions. Several further enhancements and directions for future work are also sketched.

1 Introduction

Ad-hoc networks consist of a set of mobile hosts communicating amongst themselves using wireless links, without the use of any other communication support facilities (such as base stations). They are also called mobile radio networks or multi-hop wireless networks. Two mobile hosts (MHs) are said to be within range and said to be neighbors of each other if each can receive the other's transmission. Every MH behaves in a co-operative fashion by acting as a router allowing packets destined to other MHs to pass through it.

The topology of an ad-hoc network changes every time a mobile host's movement results in establishment of new wireless links (mobile host moves within the range of another) or link disconnections (mobile host moves out of the range of another which was within its range). The rate of topology change is dependent on the extent of mobility of the hosts and the transmission range of the hosts. Routes are heavily dependent on the relative location of MHs. Hence, routes may be repeatedly invalidated in a sporadic and arbitrary fashion due to the mobility of hosts. The mobility of a single node may affect several routes that pass through it.

Ad-hoc networks have been studied extensively in the context of routing and a number of routing protocols have been proposed for ad-hoc networks including distance vector schemes [16], link reversal [6], TORA [15], dynamic source routing [11], routing using a virtual backbone [7], zone routing [9] and cluster based routing [13].

In this paper, we consider the problem of maintaining **reliable end-to-end communication** in ad-hoc networks, similar to that provided by TCP over the Internet. The end-to-end communication problem has been studied in the context of cellular wireless networks and a number of modifications and extensions to regular TCP for such networks have been proposed [1, 2, 3, 4, 18, 19]. It is desirable to use TCP directly even in ad-hoc networks in order to provide seamless portability to applications like file transfer, email

*This work was supported in part by the Texas Advanced Technology Program under Grant No. 9741-052 and by NSF under Grant No. CCR-9796331.

and WWW browsers written using standard TCP libraries. Hence, it is of interest to study the behavior of TCP in the context of ad-hoc networks and evaluate the effect of the dynamic topology on TCP's performance in order to determine whether it is feasible to use TCP as it is in the ad-hoc network domain. Our preliminary studies and results indicate that as a result of frequent and unpredictable route disruptions, TCP's performance is indeed substantially degraded both in terms of throughput and goodput (the ratio of useful data received at the destination and the total data transmitted by the source) [3]. We therefore propose a *feedback based* scheme for overcoming this problem. Our simulation experiments show that the use of feedback mechanisms along with TCP can result in substantial performance improvements.

Section 2 describes the ad-hoc network model and assumptions. Section 3 discusses the behavior of TCP in ad-hoc networks and Section 4 explains the proposed approach for improving TCP performance. In Section 5, the simulation model, experiments and observations are presented and discussed, followed by conclusions and proposed future extensions in Section 6.

2 Model and Assumptions

For the forthcoming discussions, we make the following assumptions:

1. An ad-hoc network consists of n MHs, each of which is equipped with wireless communication capability. Each MH broadcasts its packets on the wireless channels. No assumption is made about the medium access protocol.

2. The wireless links are bidirectional. This implies that all the mobile hosts have the same transmission range. Otherwise, the *weaker* hosts can receive the transmissions of *stronger* hosts but not vice versa, if they are sufficiently far away.

3. A reliable data link layer protocol is implemented over the unreliable wireless links by using, say, link level acknowledgement and retransmission. Therefore, we assume that if a packet cannot be delivered at the link layer, the higher layers will be duly informed.

4. A suitable routing protocol from among those mentioned in Section 1 is implemented to establish and maintain routes between a source and a destination. We will require the routing protocol to perform certain actions in addition to forwarding packets, which may include sending feedback messages to transport entities. The routing protocol may maintain redundant routes between different sources and destinations.

5. When a packet cannot be sent on a link despite the presence of a reliable link layer protocol, we treat the situation as the failure of the link due to mobility. The routing protocol is then responsible for adapting and maintaining routes between all sources and destinations. We assume that when a route failure occurs, a *finite time elapses until the route is restored* and communication can be resumed.

6. All packets carry the source and destination id's so that the network layer can identify the source and destination address of each packet.

3 TCP in Ad-hoc Networks

TCP is a reliable, stream-oriented transport layer protocol which has been designed for use over fixed, low error networks like the Internet. Route failures and disruptions are very infrequent as the network is fixed. Therefore, packet loss, which is detected by TCP as a timeout, can be reliably interpreted to be a symptom of congestion in the network; in response, TCP invokes congestion control mechanisms [10, 12, 17]. In other words, *TCP does not distinguish between congestion on the one hand and packet loss due to transmission errors or route failures on the other.* This inability of TCP to distinguish between two distinct problems exhibiting the same symptom results in performance degradation in ad-hoc networks, as described later in the section.

In an ad-hoc network, packet losses are frequent in the error-prone wireless medium, but the effect of these losses can be reduced using reliable link layer protocols. The greater problem is that of route failures, which can occur very frequently and unpredictably during the lifetime of a transport session, depending on the relative motion of MHs in the network. In general, whenever the mobility of an MH invalidates a route(s), the re-establishment of the route by the underlying routing protocol will take a finite amount of time. During this period of time, no packets can reach the destination through the existing route. This will result in the queuing and possible loss of packets/acknowledgements. This in turn will lead to timeouts at the source which will be interpreted by the transport protocol as congestion.

Consequently the source will:

1. *Retransmit* unacknowledged packets upon timing out.
2. Invoke *congestion control* mechanisms that include exponential backoff of the retransmission timers and immediate shrinking of the window size, thus resulting in reduction of the transmission rate.
3. Enter a slow start recovery phase to ensure that the congestion has reduced before resuming transmission at the normal rate.

This is undesirable for the following reasons:

- When there is no route available, there is no need to retransmit packets that will anyway not reach the destination.
- Packet retransmission wastes precious MH battery power and scarce bandwidth.
- In the period immediately following the restoration of the route, the throughput will be unnecessarily low as a result of the slow start recovery mechanism even though there is actually no congestion in the network.

4 A Feedback Based Approach

From the preceding discussion, it is clear that treating route failure as congestion and invoking congestion control is not advisable as *congestion control and route failure are disparate phenomena which have to be handled independently and separately.* Therefore, we propose a scheme by which the source is informed of the route failure so that it does not *unnecessarily invoke congestion control and can refrain from sending any further packets until the route is restored.* Feedback based schemes for TCP have already been proposed in the form of explicit congestion notification (ECN) [8] for fixed networks and EBSN [3] in cellular networks. As we do not have a reliable backbone in case of ad-hoc networks, neither of these methods is directly applicable in our case. Therefore, we propose a feedback based scheme for handling route failures in ad-hoc networks termed as **TCP-Feedback** or TCP-F, which is described below:

Consider, for simplicity, a single bulk data transfer session, where a source MH is sending packets to a destination MH. As soon as the network layer at an intermediate MH (henceforth referred to as the failure point or FP) detects the disruption of a route due to the mobility of the next MH along that route, it explicitly sends a **Route Failure Notification (RFN)** packet to the source and records this event. Each intermediate node that receives the RFN packet invalidates that particular route and prevents incoming packets intended for that destination from passing through that route. If the intermediate node knows of an alternate route to the destination, this alternate route can now be used to support further communication and the RFN is discarded. Otherwise, the intermediate node simply propagates the RFN towards the source.

On receiving the RFN, the source goes into a *snooze* state and performs the following steps:

1. It completely stops sending further packets (new or retransmissions).
2. It freezes (i) all of its timers, (ii) the send window of packets (iii) values of other state variables such as retransmit timer value and window size and (iv) starts a *route failure timer* which corresponds to a worst case route reestablishment time. The timeout value of this timer can be a parameter whose value depends on the underlying routing protocol.

The source remains in this snooze state until it is notified of the restoration of the route through a **Route Re-establishment Notification (RRN)** packet as explained below:

Let one of the intermediate nodes that has previously forwarded an RFN to the source, learn about a new route to the destination (through a routing update). This intermediate node then sends an RRN packet to the source (whose identity it had previously stored). All further RRNs received by this intermediate node to the same source are discarded. Any other node that receives the RRN simply forwards it towards the source.

As soon as the source receives the RRN, it changes to an active state from the snooze state - it restarts the timers from their frozen values (and does not *reset* the timers) and resumes the transmission based on the stored sender window and timeout values. These steps in effect reduce the effect of TCP's congestion control mechanism when transmission re-starts. Communication now resumes at the same rate as that before the route failure occurred. The route failure timer ensures that the source does not indefinitely remain in

the snooze state waiting for an RRN which may be delayed or lost.

We have conducted simulation experiments based on TCP-F and have compared it with basic TCP. Our results, which are described in the next section, suggest that our approach results in significant improvements in throughput as well as goodput.

5 Simulated Performance

5.1 Simulation Model

To study the behavior of TCP and compare its performance with that of the proposed TCP-F mechanism, we simulated a single unidirectional bulk transfer session. The source sends a continuous stream of data to the destination through the ad-hoc network. Since we are only interested in studying transport protocol behavior, we view the network as a *black box* (containing a fixed number of nodes between source and destination) that emulates the behavior of an ad-hoc network from the viewpoint of a transport entity. The reconfiguration of the network (due to the movement of MHs) manifests itself in the form of route failures and route re-discoveries. The transport entity sees this effect in the form of sudden packet losses and delays. The actual location of the route failure is a random parameter and route re-establishment time and frequency of failures are parameters of the simulation. No assumption is made about the routing protocol used by the underlying network layer.

5.2 Implementation

Simulation Parameters: For our experiments, we use a fixed packet size of 200 bytes and data rate of 12.8 Kbps which are reasonable values for wireless networks [3].The number of hops between the source and destination is set to 10 and the corresponding window size is 4 KBytes (20 packets). We have assumed that the network in our simulation does not suffer from congestion. Consequently, any packet loss is attributed to route failure only. The input parameters to the simulation are the **failure rate** (number of failures in the total time of simulation) and **route re-establishment delay (RRD)**. The simulation is carried out for a period of 100 seconds.

The simulation is event-driven and proceeds as follows: "Transmission" of a packet leads to a SEND event at the source which in turn schedules an ACK event and a TIMEOUT event. The timestamp of the ACK event is calculated as follows: $t_{ACK} = t_{SEND}$ + delay based on number of hops, data rate and packet size + random variance (of upto 10% of total time to account for packet delays). The timestamp of the TIMEOUT event is based on TCP's timeout estimation mechanism. Depending on the conditions as described below, one of the above events (ACK/TIMEOUT) will be valid and the other event is then discarded. In order to simulate the reconfiguration of the network due to mobility, route failure events are periodically generated. The route is "re-established" after a delay corresponding to the RRD. Based on the current time instant, location and duration of the failure and the timestamp of a SEND event, we can determine whether a packet reached the destination successfully and if an ACK was successfully received at the source. Once we have determined which packets are "lost", their ACK events are then deleted from the event queue.

Over this model, we implemented basic TCP (along with it's retransmission scheme) and the proposed TCP-F. We then ran the simulation for both cases using different values of failure interval (i.e. number of failures in the period of observation) and route re-establishment delay (RRD). It was ensured that both TCP and TCP-F were simulated under the same conditions.

5.3 Observations

Figure 1 shows the behavior of TCP with and without feedback as a function of RRD. The graph indicates that TCP-F performs much better than TCP as the value of RRD increases. This is because as the RRD value increases, more packets and acknowledgements are lost due to route failure, leading to more timeouts and greater exponential backoff in case of TCP, which drastically affects performance. By using feedback, TCP-F is able to control this backoff by ensuring that the sender's state is frozen for a substantial portion of the route failure period, during which it does not respond to timeouts. Figures 4 and 5 for an RRD value of 2 seconds and with 5 failures distributed uniformly during the simulation time, clearly illustrate the above behavior. We also found that the *improvement in performance of TCP-F over regular TCP increases with data rate*. The intuitive explanation for this is that for a given time interval, the number of packets passing through the network increases with data rate. Consequently, if we consider a failure that lasts t seconds, the number of packets that are lost in time t increases with data rate, which leads to further performance degradation in TCP. Therefore, *as data rates in wireless media increase, feedback is likely to provide even greater benefits.*

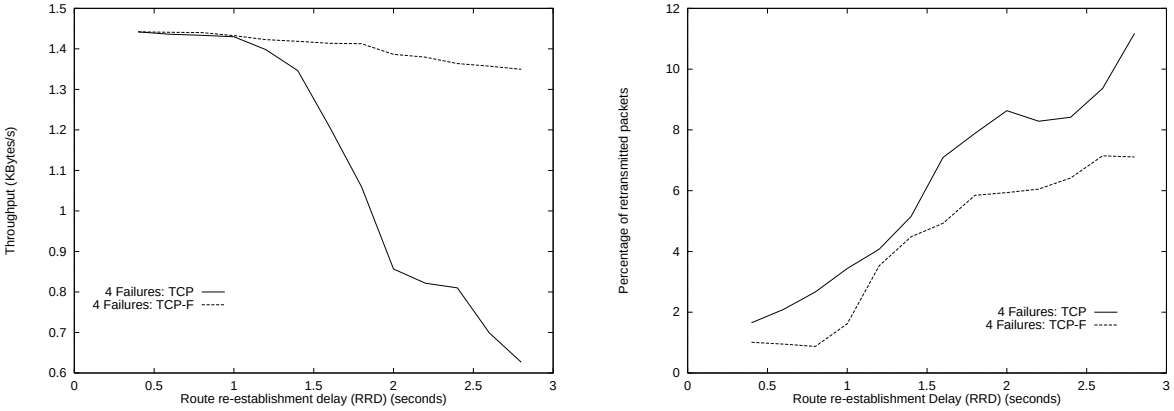

Figure 1: Throughput and retransmissions for 4 failures with data rate of 12.8 Kbps

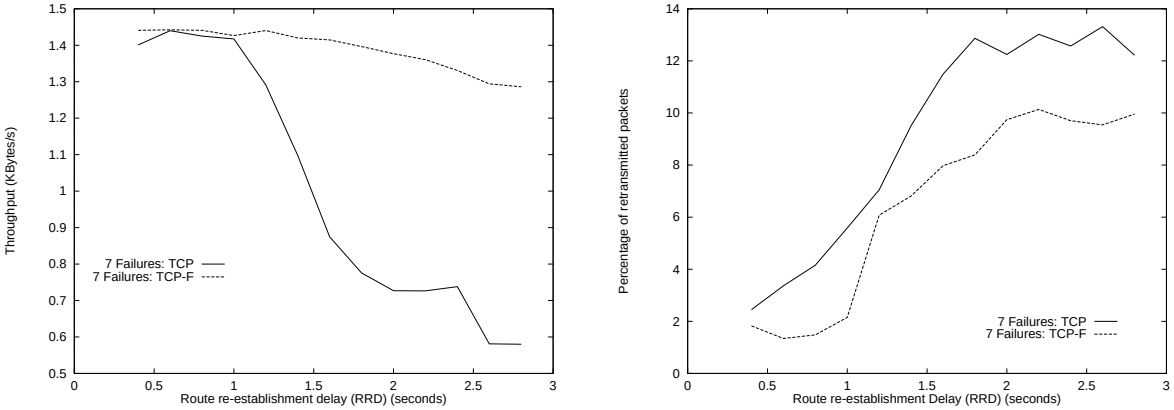

Figure 2: Throughput and retransmissions for 7 failures with data rate of 12.8 Kbps

5.4 Discussions

The use of feedback raises some interesting issues that are discussed below.

1. *Effect of multiple failures on the same route:* It is always possible that multiple route failures occur independently along different links of the route. This is however not a serious concern in case of TCP-F as the source will then receive an RFN from the nearest FP and behave accordingly. The communication will then resume only after the source receives an RRN from that FP, which means that the route has been restored.

2. *The effect of congestion on the feedback mechanism:* It is possible that in a congested network, the RFN and RRN packets may be lost or delayed. However, this is not a concern since basic TCP at the source will detect congestion and invoke congestion control. This behavior is desirable as we are only attempting to distinguish packet loss due to congestion from that due to route failure; we are not interfering with TCP's congestion control mechanism in any way.

3. *Effect of failure on multiple transport connections:* In the current discussion, we have considered a single transport connection in the simulation analysis. However, in a real life situation, there may be a number of transport connections that use the same route/link. Thus, a single route/link failure is likely to affect a number of transport connections. This raises the following questions:

   (a) How do we determine all sources affected by the failure ?

   (b) How do we inform these sources?

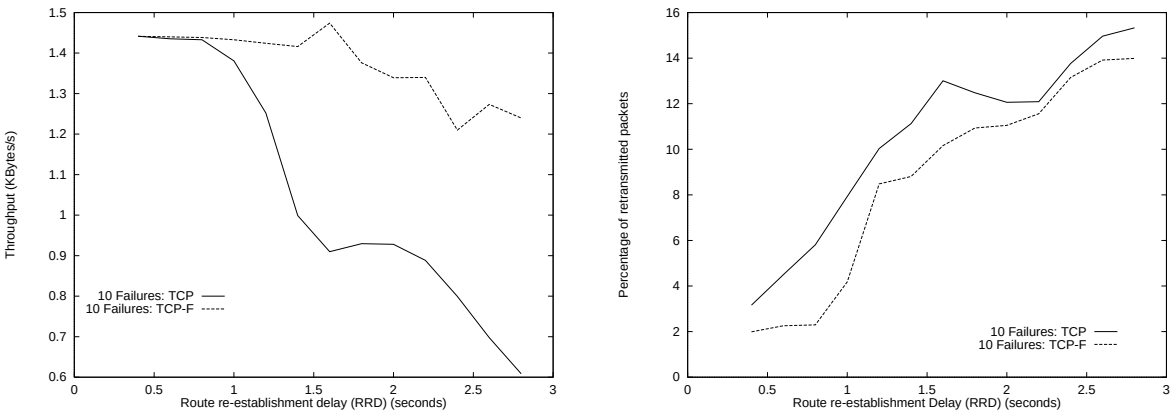

Figure 3: Throughput and retransmissions for 10 failures with data rate of 12.8 Kbps

Note that route failure is detected by the network layer at the failure point, which is completely unaware of any end-to-end transport connections. Therefore in order to detect and inform all of these sources, we can use the following approach: Whenever the failure point receives a packet, it sends an RFN in response to this packet to the source (assuming that the received packet carries the source and destination addresses). This also ensures that a source, which is not currently using this route, does not know about the route failures. We are currently exploring methods to efficiently inform all of the sources that use the route that is disrupted.

6 Conclusions

We have studied the effect of route failures, which are characteristic of ad-hoc mobile networks, on basic TCP's performance. In TCP, if the source is not aware of the route failure, the source continues to transmit (or re-transmit) packets even when the network is down. This leads to packet loss and performance degradation. Moreover, since packet loss is interpreted as congestion, TCP invokes congestion recovery algorithms when the route is re-established, leading to unnecessary throttling of transmission. We have proposed a feedback based scheme, in which the failure point notifies the source of route failure and route re-establishment, thus distinguishing route failures from congestion. We have studied the relative performance of basic TCP and TCP-F and found the results to be very encouraging. As the route re-establishment delay grows, TCP-F performs significantly better than TCP. This is attributed to the fact that we are able to reduce the number of unnecessary packet re-transmissions/timer back-offs during the route failure interval.

6.1 Extensions and Future Work

Throughout the paper, we have assumed that packets reaching the failure point are lost when the next link from the failure point is down. However, this is not so if the intermediate nodes can buffer these packets to a limited capacity. In this case, we could do the following: As the RFN propagates to the source from the failure point, all the intermediate nodes can temporarily buffer subsequent packets. If there is a substantial overlap between the newly established route and the old route, the RRN message can be used to flush out the buffers. That is, the buffered packets may be sent to the destination along the newly established route. Similarly, intermediate nodes may forward the buffered packets to the destination without waiting for an RRN on learning of new routes. This buffering scheme has the following advantages:

1. It will save packet retransmissions, and packet flow can resume even before the source learns about the route re-establishment.

2. Since the buffering is staggered across the intermediate hops, the buffering overhead at each node is expected to be low.

Another extension is to study the acknowledgement policies in TCP. When a failure occurs, a number of packets and/or acknowledgements may be lost while subsequent packets may reach the destination after the route has been re-established. This is likely to result in gaps in the receiver's window, thus affecting TCP's cumulative acknowledgement scheme. Therefore it may be worthwhile exploring alternative end-to-end acknowledgement schemes such as Selective Acknowledgement (SACK) [14] and comparing their per-

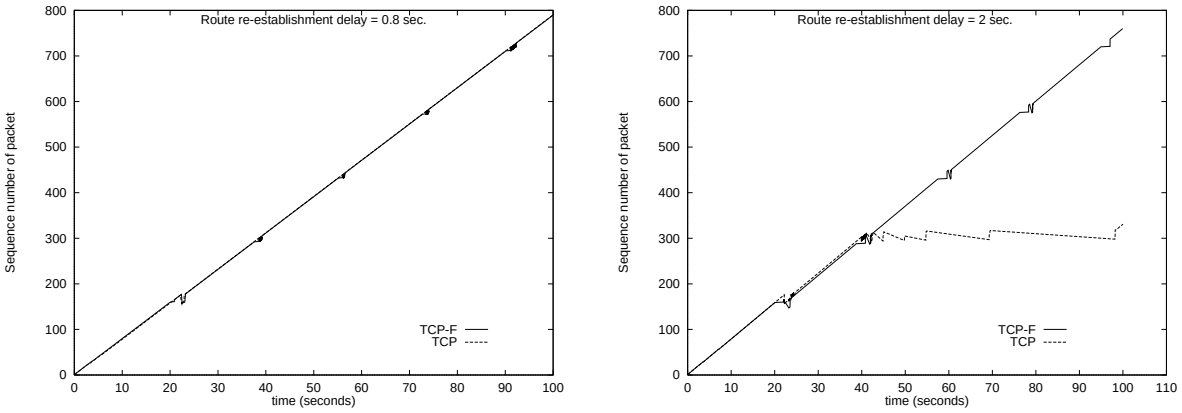

Figure 4: Sequence number of packets sent for 5 failures with data rate of 12.8 Kbps

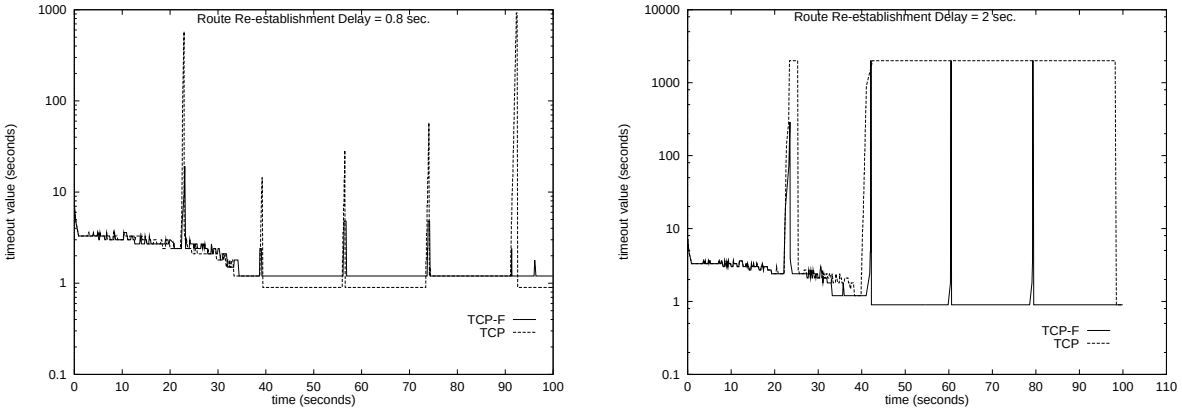

Figure 5: Timeout values for 5 failures with data rate of 12.8 Kbps

formance with the existing cumulative acknowledgement policy.

We plan to study the mutual effects of feedback and congestion on TCP performance in greater detail. We are currently working on a more extensive simulation involving multiple nodes and multiple connections to further validate our claims.

Acknowledgement

The authors wish to thank Sridhar Alagar and Ramki Rajagopalan for various comments and discussions.

References

[1] Balakrishnan, H., Seshan, S., Amir, E., and Katz, R. H. Improving TCP/IP Performance over Wireless Networks. In *Proceedings 1st ACM Conf. on Mobile Computing and Networking* November,1995.

[2] Bakre, A., and Badrinath, B.R. I-TCP: Indirect TCP for Mobile Hosts. In *Proceedings of the International Conference on Distributed Computing Systems* October, 1994.

[3] Bakshi, B.S., Krishna, P., Vaidya, N.H., and Pradhan, D.K. Improving Performance of TCP over Wireless Networks. In *Proceedings of the International Conference on Distributed Computing Systems* May, 1997.

[4] Caceres, R., and Iftode, L. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. In *Journal on Selected Areas In Communication, Spl Issue on Mobile Computing Networks* (1994),IEEE.

[5] Comer, D. Internetworking with TCP/IP. Volume 1 (Second Edition): Principles, Protocols and Architecture. Prentice-Hall Inc., 1991.

[6] Corson, S., Macker, J., and Batsell, S. Architectural Considerations for Mobile Mesh Net-

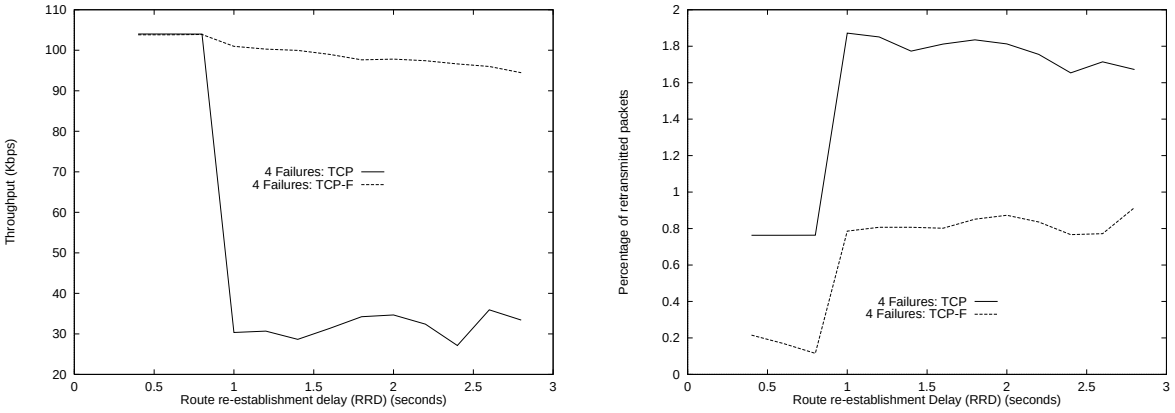

Figure 6: Throughput and % retransmissions vs. RRD for 4 failures, data rate 128 Kbps

working. *Internet DRAFT RFC Version 2*, May, 1996.

[7] Das, B., Sivakumar, R., and Bharghavan, V. Routing in Ad-Hoc Networks Using a Virtual Backbone. *Proceedings of IEEE IC3N*, 1997.

[8] Floyd, S. TCP and Explicit Congestion Notification. *ACM Computer Communication Review, Vol 5, No. 5* October, 1994.

[9] Hass, Z.J. A new routing protocol for the reconfigurable wireless networks. *Proceedings of International Conference on Universal and Personal Communication*, October 1997.

[10] Jacobson, V. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM* August, 1988, pp. 314-329.

[11] Johnson, D.B., and Maltz, D.A. Dynamic Source Routing in Ad-Hoc Wireless Networks. *Tomasz Imielinski and Hank Korth, eds, Mobile Computing* Kluwer Academic Publishers, 1996.

[12] Karn, P., and Patridge, C. Estimating Round Trip Times in Reliable Transport Protocols. In *Proceeding of ACM SIGCOMM* August, 1987.

[13] Krishna, P., Chatterjee, M., Vaidya, N.H., and Pradhan, D.K. A Cluster-based Approach for Routing in Ad-Hoc Networks. *2nd USENIX Symp. on Mobile & Location-Independent Computing* April, 1995.

[14] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. TCP Selective Acknowledgement Options. *Internet DRAFT RFC 2018* October 1996.

[15] Park, V.D., and Corson, M.S. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE INFOCOM'97* April, 1997.

[16] Perkins, C.E., and Bhagwat, P. Destination Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of ACM SIGCOMM Conference on Communication Architectures, Protocols and Apps.* August, 1994, pp. 234-244.

[17] Shenker, S., and Zhang, L. Some Observations on the Dynamics of Congestion Control Algorithms. *ACM Computer Communication Review* October, 1990, pp. 30-39.

[18] West, S., and Vaidya, N.H. TCP Enhancements for Heterogenous Networks. *Technical Report, #97-003, Comp. Sc.,Texas A&M Univ.* April, 1997.

[19] Yavatkar, R., and Bhagwat, N. Improving End-to-End Performance of TCP over Mobile Internetworks. In *Proceedings of Workshop on Mobile Computing Systems and Apps.* December, 1994.