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 change may lead to sudden packet losses and delays. Transport protocols like TCP, which have been designed for reliable fixed networks, misinterpret this packet loss as congestion and invoke congestion control, leading to unnecessary retransmissions and loss of throughput. To overcome this problem, a feedback scheme is proposed so that the source can distinguish between a route failure and network congestion. When a route is disrupted, the source is sent a Route Failure Notification packet, allowing it to invalidate its timers and stop sending packets. When the route is reestablished, the source is informed through a Route Reestablishment Notification packet, upon which it resumes packet transmissions. Simulation experiments show that in the event of route failures, as the route reestablishment time increases, the use of feedback provides significant improvements in performance.

A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks

KARTIK CHANDRAN, SUDARSHAN RAGHUNATHAN, SUBBARAYAN VENKATESAN, AND RAVI PRAKASH
UNIVERSITY OF TEXAS AT DALLAS

Ad hoc networks consist of a set of mobile hosts communicating among themselves using wireless links, without the use of any other communication support facilities (e.g., base stations). They are also called *mobile radio networks* or *multihop wireless networks*. Two mobile hosts (MHs) are said to be within range and neighbors of each other if each can receive the other's transmission. Every MH behaves in a cooperative 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 an MH's movement results in the establishment of new wireless links (an MH moves within range of another) or link disconnections (an MH moves out of range of another which was within its range). The rate of topology change is dependent on the extent of mobility and transmission range of the hosts. Routes are heavily dependent on the relative location of MHs. Hence, routes may be repeatedly invalidated in an unpredictable and arbitrary fashion due to the mobility of hosts. The mobility of a single node may affect several routes that pass through it.

In this article, we consider the problem of maintaining *reliable end-to-end communication* in ad hoc networks, similar to that provided by TCP over the Internet. It is desirable to use TCP directly even in ad hoc networks in order to provide seamless portability to applications like file transfer, e-mail, and Web 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 dynamic topology on TCP performance. Our studies and simulation results indicate that as a result of frequent and unpredictable route disruptions, TCP's performance is substantially degraded. We propose a *feedback mechanism* for overcoming this problem. Simulation experiments show that the use of feedback mechanisms along with TCP can result in substantial performance improvements.

The rest of the article is organized as follows. We describe existing work in related areas. We describe the ad hoc network model and assumptions. We discuss the behavior of TCP in ad hoc networks and explain the proposed approach for improving TCP performance. The simulation model, experiments, and observations are presented and discussed, followed by conclusions.

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.

A preliminary version of this article appears in the Proceedings of the 18th International Conference on Distributed Computing Systems, *Amsterdam, The Netherlands, May 1998.*

Related Work

The dynamic nature of topology in ad hoc networks poses many interesting problems in the domain of routing protocols. As a result, ad hoc networks have been studied extensively in the context of routing (i.e., at the network layer). In these networks, it is necessary to have a routing protocol that:

- Quickly provides relatively stable, loop-free routes
- Adapts to the mobility of the network

Conventional protocols like link state and distance vector do not match these requirements because they do not converge quickly enough or scale well as mobility increases. A number of alternative protocols have been proposed, including a dynamic sequenced distance vector (DSDV) based scheme [1], dynamic source routing [2], link reversal [3], TORA [4], routing using a virtual backbone [5], zone routing [6], cluster-based routing [7], and tree-multicast-based routing strategies [8].

There has been a lot of research in recent years on reliable transport protocols for cellular wireless networks. In cellular wireless networks, there are frequent breaks in communication due to handoffs, during which there is packet loss in the transition phase. This leads to the initiation of congestion control procedures at the source and loss of throughput. In addition, wireless links have lower bandwidth and are less reliable than wired links. This adversely affects end-to-end performance of transport protocols like TCP. Various solutions have been proposed for this problem.

In the split-connection method [9–11], an end-to-end connection between two MHs is split into separate connections. Each MH has a connection with its base station, and the base station establishes a proxy connection on behalf of the MHs. The Snoop protocol [12] follows the approach of making simple modifications at the base stations to improve performance. Packets are cached at the base station, and local retransmissions are performed to alleviate problems caused by high bit error rates. In the fast retransmit approach [13], a triplicate acknowledgment is sent by the destination MH to the TCP source to ensure that the source immediately resumes transmission after a handoff. Another approach suggested by Bakshi *et al.* [14] uses feedback packets from base stations to inform the TCP sender of wireless link errors and prevents loss of performance by resetting timers and avoiding timeouts.

The work closest to ours are Freeze-TCP [15] and MTCP [16], both in cellular networks. In Freeze-TCP, when a receiver senses an impending disconnection (say because of severe fading, hard handoff), it sends a few ACKs with advertised window size equal to 0, thereby freezing the source without reducing the source's window. As soon as the connection is reestablished, fast transmit is enabled by triple ACK. In simulated experiments, the performance improvement of Freeze-TCP with triple acks ranges from 25 to 51 percent over normal TCP. Brown and Singh [17] propose MTCP. This approach is like I-TCP, and the single TCP connection is split into two: one from the source to a supervisor host (SH) and the second from the SH to an MH. The SH is a machine that controls several base stations. When the SH receives a packet from the source, unlike in I-TCP, it does not send an ACK to the source. However, to improve TCP performance, when ACKs do not arrive (because of MH disconnection, etc.), the SH chokes the sender; when the MH reconnects, MTCP allows the sender to transmit at full throttle.

WTCP [18] is another method for reliable delivery in wireless WANs using a CDPD network involving last-hop wireless links. The receiver of a stream computes the rate by using interpacket delays and communicates them to the sender on ACKs. This ensures that ACKs traveling different paths, thereby having different rate characteristics, do not lead to incorrect computation of sending rate. Also, using the history of packet loss, when there is a new interval of packet losses, WTCP predicts if the loss is due to congestion or blackout. In addition to these, the authors use many other techniques for increased throughput (selective ACKs, controlling ACK frequency, sender using probe packets, etc.).

All of the above mechanisms heavily depend on the presence of the wired base station network, and hence cannot be directly applied in ad hoc networks, where all nodes are mobile and all links wireless.

Holland and Vaidya [16] provide a comprehensive analysis of the effect of mobility on TCP throughput.

The work described above is instructive in understanding the effect of packet loss on TCP performance and serves as a good starting point for our analysis.

Model and Assumptions

For the forthcoming discussions, we make the following assumptions:

- An ad hoc network consists of n MHs, each equipped with wireless communication capability. Each MH broadcasts its packets on the wireless channels. No assumption is made about the medium access protocol.
- The wireless links are bidirectional. This implies that all the MHs 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.
- A reliable data link layer protocol is implemented over the unreliable wireless links by using, say, link-level acknowledgment and retransmission. Therefore, we assume that if a packet cannot be delivered at the link layer, the network layer will be duly informed.
- A suitable routing protocol 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 include sending feedback messages to transport entities. The routing protocol may maintain redundant routes between different sources and destinations.
- 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 all routes. We assume that when a route failure occurs, *a finite time elapses until the route is restored* and communication can be resumed.
- All packets carry the source and destination ids so that the network layer can identify the source and destination address of each packet.

TCP in Ad Hoc Networks

TCP is a reliable, stream-oriented transport layer protocol which was designed for use over fixed low-error networks like the Internet. Route failures and disruptions are very infrequent since the network is fixed. Therefore, packet loss, which is detected by TCP as a timeout, can be reliably interpreted as a symptom of congestion in the network. In response, TCP invokes congestion control mechanisms [19–21]. Thus, *TCP does not distinguish between congestion and packet loss due to transmission errors or route failures*. This inability of TCP to distinguish between two distinct problems exhibiting the same symptom results in performance degradation in ad hoc networks, 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 more serious 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 reestablishment of the route by the underlying routing protocol takes a finite amount of time. During this period of time, no packets can reach the destination through the existing route. This results in the queuing and possible loss of packets and acknowledgments. This in turn leads to timeouts at the source which are interpreted by the transport protocol as congestion.

Consequently the source:

- *Retransmits* unacknowledged packets upon timing out
- Invokes *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

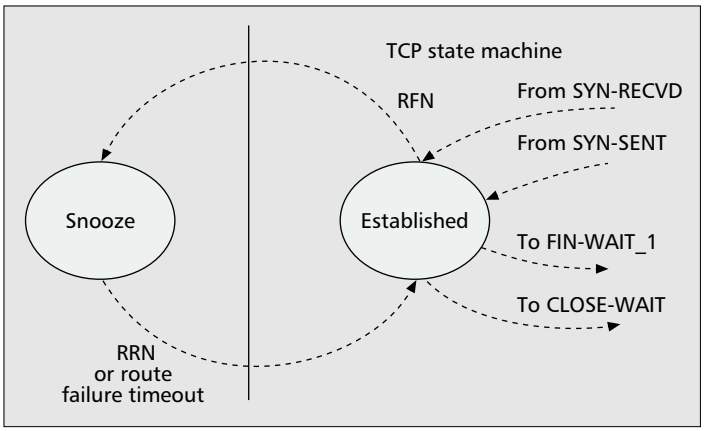

Figure 1. *The TCP-F state machine.*

- Enters a *slow start recovery phase* to ensure that the congestion is 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 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 may be no congestion in the network.

A Feedback-Based Approach

From the preceding discussion, it is clear that treating route failure as congestion (and invoking congestion control) is not advisable because *congestion 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) [22] for fixed networks and EBSN [14] 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. Our scheme, which we call *TCP-Feedback* (TCP-F), 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, 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 the particular route and prevents incoming packets intended for the 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 toward the source.

On receiving the RFN, the source goes into a *snooze* state (Fig. 1) and performs the following:

- It completely stops sending further packets (new or retransmissions).
- It then
  - –Marks all of its existing timers as invalid
  - –Freezes the send window of packets
  - –Freezes values of other state variables such as retransmit timer value and window size
  - –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 reestablishment 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 previously stored). All further RRNs received by this intermediate node for the same source-destination connection are discarded. Any other node that receives the RRN simply forwards it toward the source.

As soon as the source receives the RRN, it changes to an active state from the snooze state. It then flushes out all unacknowledged packets in its current window. Since most packets in transit during the failure period would have been affected, packets can be flushed out without waiting for acknowledgments from the receiver. The number of retransmitted packets directly depends on the current window size.

These steps in effect reduce the effect of TCP's congestion control mechanism when transmission restarts. Communication now resumes at the same rate as before the route failure occurred, ensuring that there is no unnecessary loss of throughput in this period. TCP's congestion control mechanism can now take over and adjust to the existing load in the system. 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 to compare the performance of TCP-F and basic TCP. Our results, described in the next section, suggest that this approach is indeed beneficial in preventing performance degradation.

Simulated Performance

Simulation Model

To study the behavior of TCP and compare its performance with that of the proposed TCP-F mechanism, we have simulated a single unidirectional bulk transfer session. The simula-

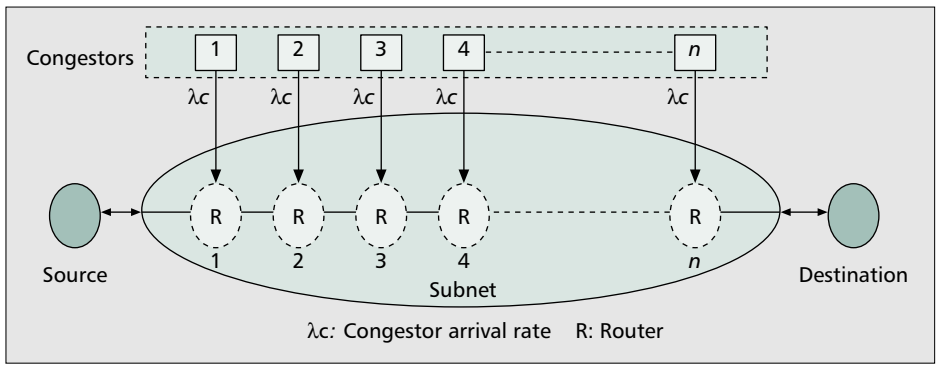

Figure 2. *The simulation model.*

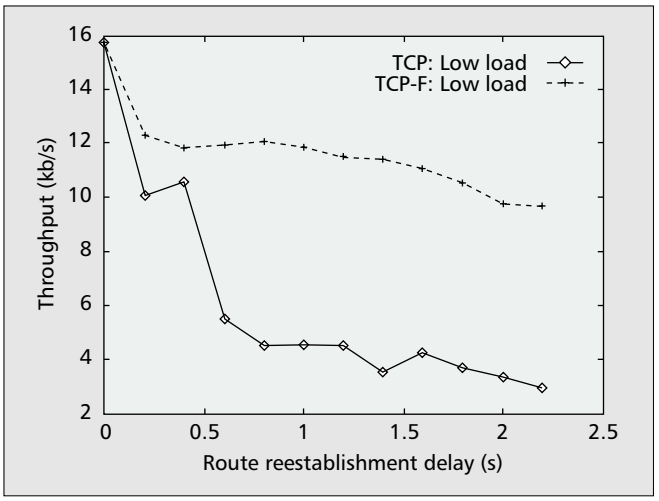

Figure 3. *Throughput vs. RRD for low load.*

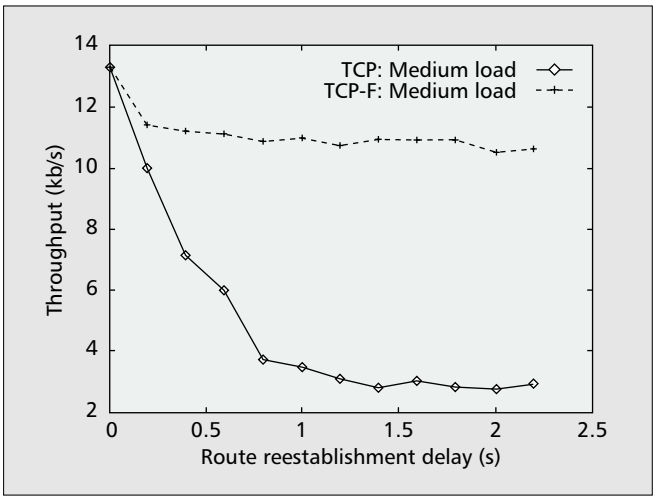

Figure 4. *Throughput vs. RRD for medium load.*

tion model, consisting of the source, destination, intermediate routers, and congestors, is illustrated in Fig. 2.

The source sends a continuous stream of data to the destination through the simulated ad hoc network, which consists of a fixed number of routers connected in a chain. A *congestor* is connected to each router, which models the traffic of the rest of the network that is flowing through this router. Congestors inject dummy packets into the routers at a specified arrival rate, which controls the load in the network. The routers are connected through bidirectional links which are independently controlled by a *link controller*. The link controller periodically "breaks" links by setting their state to "down" and back to "up" after a specified delay. The link is chosen randomly — each link has an equal probability of failing. Whenever a router sees a failed link while forwarding a packet, the packet is "dropped." If feedback is enabled, the router sends feedback messages to the source. These messages are assumed to be high priority; hence, they are not queued up with the regular packets. TCP with the feedback mechanism is implemented at the source and destination. The implementation of the TCP portion is based on source code from the network simulator ns from Lawrence Berkeley Labs. The feedback mechanism is disabled while studying the performance of basic TCP.

This model captures the effect of topology change of the network from the point of view of a transport entity and abstracts the other aspects of the network layer like routing. The load on the routers is controlled by varying the packet generation rate of the congestors. The route reestablishment delay (RRD) reflects the ability of the network to respond to topology changes. By varying the RRD and congestor arrival rates, we can study the performance of transport protocols as a function of the network layer performance at different loads without actually implementing a full-fledged network layer. Since we do not make any other assumptions about the network layer, our results are applicable for any network layer protocol.

The limitation of this model is that it does not account for the interaction of traffic due to different connections that share links, since we have assumed that there is a single source and destination and that the congestors generate traffic independent of one another. However, the same assumption enables us to accurately control the load of the network and hence closely study the performance at different load conditions.

Implementation

Simulation Parameters — For our experiments we have used a fixed packet size of 200 bytes and data rate of 12.8 kb/s 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). The input parameters to the simulation are the packet generation rate for the congestors and RRD. We ran the simulations for three values of arrival rate, corresponding to low, medium, and high loads, and for RRD values ranging from 0 to 2.4 s. Each data point is taken as the mean of 20 values.

Observations

Figures 3, 4, and 5 show the comparative behavior of TCP and TCP-F as a function of RRD for different network loads. From the figures, it is evident that the throughput of TCP degrades rapidly with increasing RRD. This is because as the RRD value increases, more packets and acknowledgments are lost due to route failure. This in turn leads to more timeouts at the source, as a result of which there is greater exponential backoff (Fig. 7). Even when the route is reestablished, the backoff value remains high since only unretransmitted packets are used to calculate RTT estimates and adjusting timeout values. Since window size is small in the period after the route failure, we can be faced with a situation where we have a full window of outstanding packets with very high timeout values, in which either the packets or the ACKs are lost. In such a case, the source remains idle for a period corresponding to this high timeout value, as can be seen from Figs. 6, 7, and 8.

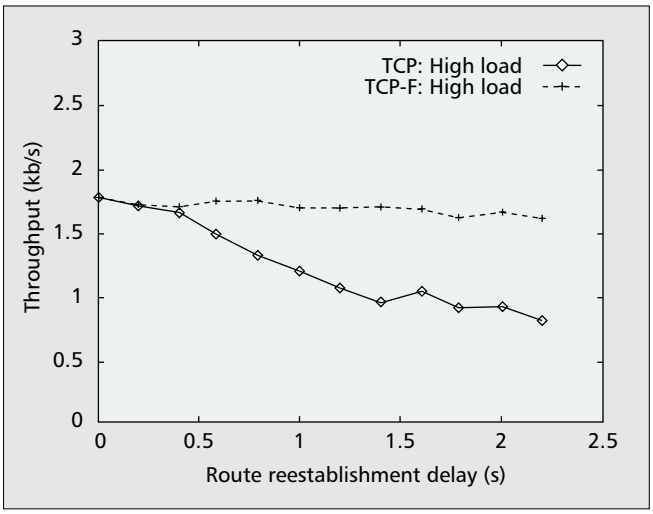

Figure 5. *Throughput vs. RRD for high load.*

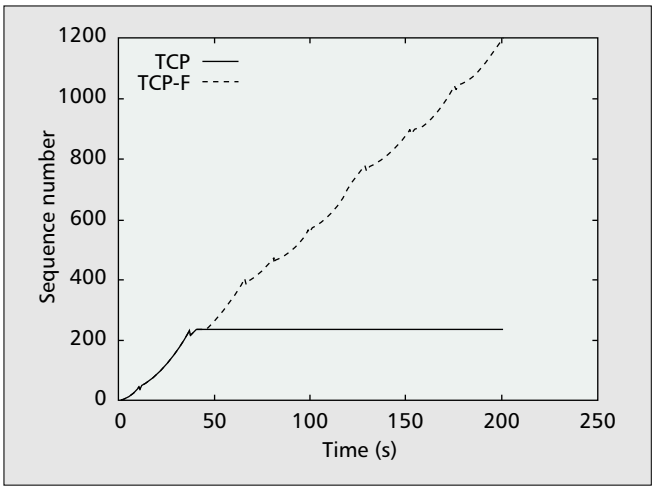

Figure 6. *Sequence number vs. time at medium load and RRD = 0.7 s.*

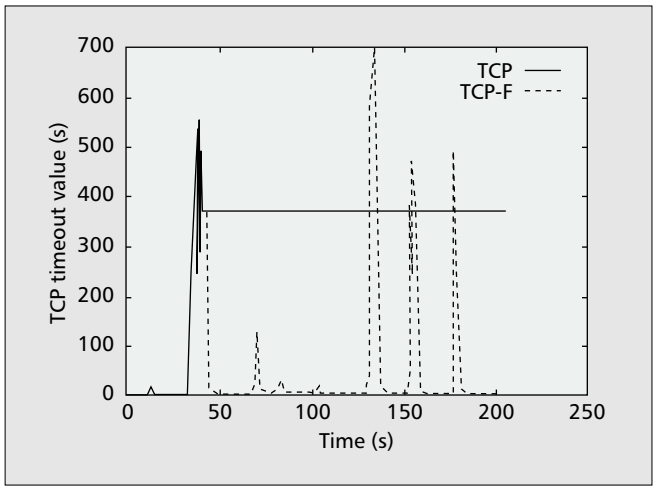

Figure 7. *TCP timeout value vs. time at medium load and RRD = 0.7 s.*

On the contrary, TCP-F throughput remains relatively stable even for high values of RRD. By using feedback, TCP-F is able to control this backoff by ensuring that the source state is frozen for a substantial portion of the route failure period, during which it ignores timeouts. In addition, after receiving the RRN, the source retransmits its existing window of packets, which enables very quick recovery. Therefore, even if backoff does occur, its effect is nullified and the source can resume transmitting packets at the full rate supported by the network. Also, as the route failure frequency increases (due to high MH mobility), the throughput improvement in TCP-F is even more pronounced. It is interesting to note that for all values of load, the relative improvement of TCP-F over TCP is almost the same. In other words, the feedback mechanism works irrespective of the load on the network.

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 s, 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*.

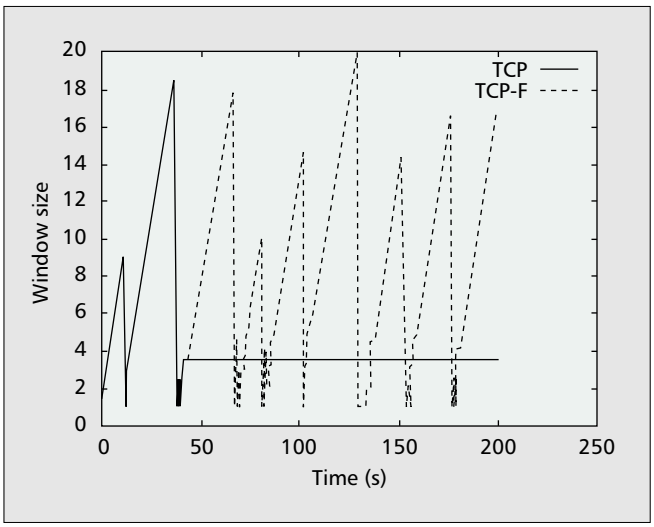

Figure 8. *Window size vs. time at medium load and RRD = 0.7 s.*

Discussions

The use of TCP-F raises some questions that merit further discussion:

- *Overhead on routers*: The success of TCP-F is crucially dependent on the ability of the router to detect route failures and reestablishments, and quickly provide feedback to the source. When an RFN is sent in response to a failure, each router that forwards the RFN needs to record the source id of this RFN so that it can send an RRN when a route is found. This information needs to be cached in the routing tables and discarded when it becomes obsolete. This can be integrated into the routing table maintenance modules in the routers.
- *The effect of multiple failures on the same route*: It is always possible that multiple route failures occur independently on different links of the route. This, however, is not a serious concern in the case of TCP-F since the source will then receive an RFN from the nearest FP and behave accordingly. Communication will then resume only after the source receives an RRN, which means that the route has been restored.
- *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 since 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.
- *The 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:
  - –How do we determine all sources affected by the failure?
  - –How do we inform these sources?

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 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 disrupted route.

Conclusions

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

Extensions and Future Work

Throughout this article, we assume 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 one, the RRN message can be used to flush out the buffers (i.e.,, 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:

- It will save packet retransmissions, and packet flow can resume even before the source learns about the route reestablishment.
- Since buffering is staggered across the intermediate hops, the buffering overhead at each node is expected to be low.

Another area that merits further investigation is alternative acknowledgment policies. When a route failure occurs, a number of packets and/or acknowledgments may be lost while subsequent packets may reach the destination after the route has been reestablished. This is likely to result in gaps in the receiver's window, which adversely affects TCP's cumulative acknowledgment scheme. Therefore, it may be worthwhile exploring alternative end-to-end acknowledgment schemes such as selective acknowledgment (SACK) [23] and comparing their performance with cumulative acknowledgment.

Acknowledgments

The authors wish to thank Sridhar Alagar and Ramki Rajagopalan for various comments and discussions. The authors would also like to thank the Networks Research Group at Lawrence Berkeley Laboratories for making their network simulator ns available in the public domain.

References

[1] C. E. Perkins and P. Bhagwat, "Destination Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proc. ACM SIGCOMM Conf. Commun. Architectures, Protocols and Apps.*, Aug. 1994, pp. 234–44.
[2] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks," T. Imielinski and H. Korth, Eds., *Mobile Computing*, Kluwer, 1996.
[3] S. Corson, J. Macker, and S. Batsell, "Architectural Considerations for Mobile Mesh Networking," Internet draft, v. 2, May 1996.
[4] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proc. IEEE INFOCOM '97*, Apr. 1997.
[5] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in Ad-Hoc Networks Using a Virtual Backbone," *Proc. IEEE IC3N*, 1997.
[6] Z. J. Hass, "A New Routing Protocol for the Reconfigurable Wireless Networks," *Proc. ICUPC*, Oct. 1997.
[7] P. Krishna *et al.*, "A Cluster-Based Approach for Routing in Ad-Hoc Networks," *2nd USENIX Symp. Mobile & Location-Independent Comp.*, Apr. 1995.
[8] M. Gerla, C. Chiang, and L. Zhang, "Tree Multicast Strategies in Mobile," *Multihop Wireless Networks*, ACM MONET, 1999.
[9] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. Int'l. Conf. Dist. Comp. Sys.*, Oct. 1994.
[10] R. Yavatkar and N. Bhagwat, "Improving End-to-End Performance of TCP over Mobile Internetworks," *Proc. Wksp. Mobile Comp. Sys. and Apps.*, Dec. 1994.
[11] Z. J. Haas, and P. Agrawal, "Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems," *ICC '97*, June 8–12, Montreal, Canada.
[12] H. Balakrishnan *et al.*, "Improving TCP/IP Performance over Wireless Networks," *Proc. 1st ACM Conf. Mobile Comp. and Net.*, Nov. 1995.
[13] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE JSAC*, Special Issue on Mobile Computing Networks, 1994.
[14] B. S. Bakshi *et al.*, "Improving Performance of TCP over Wireless Networks," *Proc. Int'l. Conf. Dist. Comp. Sys.*, May, 1997.
[15] T. Goff *et al.*, "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," *Proc. IEEE INFOCOM*, 2000.
[16] G. Holland and N. H. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," 5th Annual Int'l. Conf. Mobile Comp. and Net., Aug. 1999.
[17] K. Brown and S. Singh, "M-TCP:TCP for Mobile Cellular Networks," *ACM Comp. Commun. Rev.*, 1997.
[18] P. Sinha *et al.*, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *Proc. ACM Mobicom*, Seattle, WA, Aug. 1999.
[19] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM*, Aug. 1988, pp. 314–29.
[20] P. Karn and C. Patridge "Estimating Round Trip Times in Reliable Transport Protocols," *Proc. ACM SIGCOMM*, Aug. 1987.
[21] S. Shenker and L. Zhang, "Some Observations on the Dynamics of Congestion Control Algorithms," *ACM Comp. Commun. Rev.*, Oct. 1990, pp. 30–39.
[22] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Comp. Commun. Rev.*, vol. 5, no. 5, Oct. 1994.
[23] M. Mathis *et al.*, "TCP Selective Acknowledgement Options," Internet draft RFC 2018, Oct. 1996.

Additional Reading

[1] C.-C. Chiang *et al.*, "Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel," IEEE Singapore Int'l. Conf. Networks, 1997.
[2] D. Comer, *Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture*, 2nd ed., Prentice Hall, 1991.
[3] S. West and N. H. Vaidya, "TCP Enhancements for Heterogeneous Networks, Tech. rep. 97-003," Comp. Sci., Texas A&M Univ., Apr. 1997.

Biographies

KARTIK CHANDRAN is currently working as a software engineer with Cisco Systems Inc., San Jose, California. He is involved with the design and development of QoS features to support high-quality voice and video on cable modem networks. He has also been a contributing member of the DOCSIS specification team, on which most currently deployed data over cable networks are based. He has a Bachelor's in computer engineering from the Government College of Engineering, Pune, India, and a Master's in computer science from the University of Texas at Dallas. His current technical interests are mainly centered around efficient algorithms and advanced system architectures to support QoS on broadband IP networks.

SUDARSHAN RAGHUNATHAN received his M.S. degree in computer science from the University of Texas at Dallas in 1998. For the past two years, he has been working at Ericsson on advanced wireless networks and data networks. His interests are in wireless networks, QoS, packet networks, and routing algorithms.

SUBBARAYAN VENKATESAN (venky@utdallas.edu) received his M.S. and Ph.D. degrees in computer science from the University of Pittsburgh. He joined the Computer Science faculty at the University of Texas at Dallas in 1989 and is currently an associate processor of computer science. Since August 2000 he has been with Cisco Systems in the wireless business unit, where he is currently technical leader. His interests are in distributed algorithms and their applications in reliable networks and mobile wireless networks.

RAVI PRAKASH received a B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi, in 1990, and M.S. and Ph.D. degrees in computer and information science from The Ohio State University in 1991 and 1996, respectively. He joined the Computer Science program at U.T. Dallas in July 1997. During 1996–1997 he was a visiting assistant professor in the Computer Science department. His areas of research are mobile computing and distributed systems.