Analysis of TCP Performance on Ad Hoc Networks Using Preemptive Maintenance Routing*

Tom Goff†, Nael B. Abu-Ghazaleh‡ and Dhananjay S. Phatak†

†CSEE Dept,
University of Maryland Baltimore County
Baltimore, Maryland 21250
tomgoff@ibm.net, phatak@umbc.edu

‡Computer System Research Laboratory
Dept. of CS, Binghamton University
Binghamton, NY 13902–6000
nael@cs.binghamton.edu

Abstract

In mobile Ad hoc networks, the topology of the network is constantly changing as nodes move in and out of each others range, breaking and establishing links. TCP performs poorly in such networks because packets that are lost due to path disconnections trigger TCP's congestion avoidance mechanisms. In this paper, we investigate the effect of preemptive routing protocols, where an alternative path is found before an actual disconnection occurs, on the performance of TCP. Preemptive routing should perform well for TCP traffic because it reduces the delays caused by TCP's unnecessary use of congestion avoidance when paths break. We observe this behavior under some, but not all scenarios. Specifically, it appears that when the network is saturated, the additional traffic introduced by preemptive routing causes small degradation in performance. In the analysis process, we encountered an unfairness problem resulting from interaction between the routing protocol and the MAC layer under multiple continuous transmission cases. Similar unfairness problems were encountered by other studies – however, the observations of those studies related those problems to the number of hops, and not the routing effects as we observed. This motivates the study of fairer wireless MAC protocols for multi-hop and ad hoc networks.

1 Introduction

Since TCP was designed for wired networks, it assumes that packet losses are due to congestion. Therefore, when a packet is lost, TCP applies "congestion avoidance" mechanisms and slows its transmission rate (by reducing the congestion window and exponentially backing off its retransmit timers [12]). Unfortunately, this causes TCP to perform poorly in wireless environments where packet losses due to transmission errors are frequent (even without congestion) [2, 3]. In addition, in a mobile last-hop environments, packets can be lost due to hand-offs as a mobile node moves out of range of a base station and into the range of another [5, 9]; packets lost during such transitions also initiate TCP's congestion avoidance. Several researchers have addressed optimizing TCP in wireless last-hop environments [2, 5, 9, 25].

In mobile ad hoc networks, the network topology is in continuous flux – existing paths are broken and new paths are made as the nodes move. Thus, the losses due to mobility (as a hand-off to a new path from an expired one occurs) are more frequent than those in last hop wireless environment. These losses can cause significant degradation of TCP performance especially if the mobility is high [11]. Since the routing algorithm is responsible for finding paths between communicating nodes, it has a direct influence on the frequency of packet losses due to mobility.

Routing in ad hoc network is a challenging problem that has received wide interest [10, 13, 15, 18, 19, 20, 21, 23]. In existing protocols, an alternative path is sought only after the current active path fails. The delay required to detect a path failure is very high in comparison to typical packet latencies (several retries have to time-out before a path is "pronounced dead"). Elsewhere [8], we investigated adding preemptive maintenance to ad hoc routing protocols by finding alternative paths when a link is in danger of breaking. Adding preemptive maintenance to the Dynamic Source Routing Protocol (DSR) [15] and the Ad Hoc On Demand Distance Vector Protocol (AODV) [20] resulted in a drastic reduction in the number of broken paths for Constant Bit Rate scenarios (uniform UDP traffic) and improved the overall latency and jitter of the packet delivery. In this paper, we investigate the effect of using preemptive routing

*This work was partially supported by NSF grants EIA–9911099 and ECS–9875705.

0190-3918/01 $10.00 © 2001 IEEE

on the performance of TCP in ad-hoc networks. Since the preemptive approach reduces the number of broken paths substantially, the number of lost packets due to mobility is significantly reduced. There is reason to believe that preemptive routing will benefit the performance of TCP by avoiding unnecessary congestion avoidance due to packets dropped when a path is broken. We study the effect of different levels of preemptive maintenance on the performance of several TCP traffic scenarios (telnet, ftp and http) and different mobility patterns.

The remainder of this paper is organized as follows. Section 2 overviews preemptive routing. Section 3 overviews TCP's congestion avoidance mechanisms and their effect on the performance of wireless networks. Section 4 presents a simulation study of TCP using preemptive DSR for different traffic scenarios. Finally, Section 5 presents some concluding remarks.

2 Preemptive Routing

In traditional routing algorithms, a change of path occurs when a link along the path fails. A link failure is costly since: (i) multiple retransmissions/timeouts are required to detect the failure; (ii) a new path must be found (in on-demand routing). Since paths rarely fail in wired networks, this is not an important cost. Routing protocols in mobile ad-hoc networks follow this model despite the significantly higher frequency of path disconnections that occur in this environment.

Existing ad-hoc routing protocols can be classified into two categories: **Table-driven** (proactive) algorithms maintain information about the links for all the network while **Source initiated on-demand** (reactive) protocols initiate a route discovery when a path to a destination is needed (to establish a new flow, or because the current path is broken). On demand protocols might experience delays to discover a path when one is needed. However, the traffic overhead is less than Table-driven algorithms where many of the updates are for unused paths [4, 14]. Many routing protocols have been suggested, including proactive (e.g., [6, 13, 17, 19, 21]), reactive (e.g., [15, 18, 20]) and hybrids (e.g., [10]). In both types of protocol, recovery from a broken path is initiated only after a path is broken.

A preemptive routing algorithm initiates recovery action early by detecting that a link is likely to be broken soon and finding and using an alternative path before the cost of a link failure is experienced. More specifically, the algorithm consists of two components: (i) detecting that a path is likely to be broken soon (and informing the source); and (ii) finding a better path and switching to it before the active path breaks. Reducing the number of broken paths improves the average latency and jitter of the packets (fewer packets are dropped/delayed because of path disconnections).

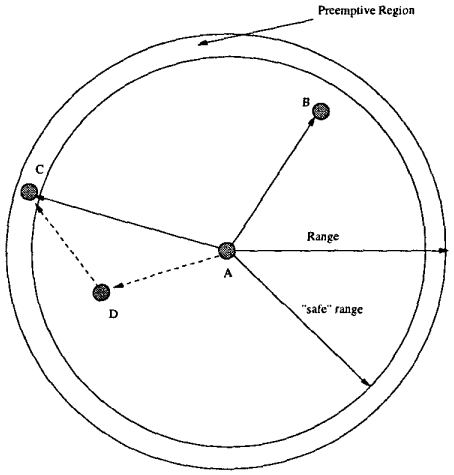

Figure 1. Preemptive Region

Ideally, a preemptive route rediscovery should complete before the path is broken; the nodes would then seamlessly switch to a new path with no "dead-time". Thus, depending on the velocities of the mobile nodes, a warning should be generated when the distance between them approaches the transmission range. A preemptive region of width w can be derived by relating the average node velocities to an estimate of the recovery time. Figure 1 demonstrates a preemptive region around a source. As node C in the figure enters this region, the signal power of received packets from the source A falls below the preemptive threshold, generating a warning packet to A. A initiates route discovery action, and discovers a route through D; A switches to this route avoiding the failure of the path as C moves out of direct range of A.

In [8] we related w to the signal power of the received packets. More specifically, if the signal power falls below a preemptive threshold, the receiver concludes that it has entered the preemptive region. The estimate of the packet power level must be immune to transient power fluctuations due to fading and/or multi-path effects [22]. This is accomplished by initiating a number of null message exchanges across a hop where a packet is received below the preemptive warning (at a minimum, checking the warning packet power level resulting in 2 checks).

The preemptive ratio (δ) is the ratio of the preemptive threshold to the minimum detectable power:

$$\delta = \frac{P_{threshold}}{P_{range}} = \frac{\frac{P_0}{(range-w)^4}}{\frac{P_0}{range^4}} = (\frac{range}{(range-w)})^4. \quad (1)$$

WaveLAN cards have a range of 200 meters in open environments in the 2.4GHz band [1]. The preemptive ratio for a preemptive region of width 4 meters is $(\frac{200}{(200-4)})^4 =$

1.08. This value corresponds to a preemptive threshold of $1.08P_{range} = 3.96 \cdot 10^{-10}$ Watts. If a packet is received with a signal power below this value, a warning is generated.

3 TCP in Ad hoc Networks

In addition to transmission errors at the link level, Ad hoc environments also suffer "path breaks" due to node mobility. When packets are lost due to either of these reasons, there is no need to initiate a congestion avoidance/control procedure. Unfortunately, TCP invokes the congestion avoidance/control procedures on any lost packet. Thus, a path break leads to under-utilizing the bandwidth due to the following reasons:
(1) When packets are lost, the re-transmission timers are exponentially backed off as part of the standard congestion avoidance procedures implemented in TCP. Upon a path break, the source initiates a route discovery. If a route is found (i.e., a reply is received in response to the route-query) just when TCP has entered a long re-transmit back-off period, no packets will be sent until the back-off is complete.
(2) In response to packet losses, TCP drops its congestion window (which determines how fast packets can be sent). In most TCP implementations, the window size can be dropped to one segment and the slow start mechanism invoked. This effect can be seen in Figure 2.

Avoiding path breaks prevents TCP from invoking congestion avoidance, which in turn prevents losses due to either of the above mechanisms. Preemptive routing significantly reduces the number of path breaks, and hence should lead to better TCP performance. Note that if a path break cannot be avoided, then it is still possible to minimize the effect of both the above mechanisms using other mechanisms [9, 11].

4 Experimental Study

The Dynamic Source Routing (DSR) protocol [15] was modified for preemptive maintenance (we call this version PDSR). In DSR, each packet carries a full route to the destination in its header (specified by the source). The route is obtained by a route discovery process: when a node has a packet to send with no path available to the destination, it broadcasts a route-request. The route-request is propagated until it reaches the destination node – at that stage, the destination reverses the route taken by the request packet and sends the discovered route to the sender.

In modifying DSR the following changes were made: (i) a preemptive route warning is generated when a packet is received with signal power lower than the preemptive threshold. When the warning is received by the source it initiates a route request; (ii) route-requests are propagated only if they are received with a signal power above the threshold (thus, only paths where every link is above the preemptive threshold are discovered); and (iii) a new path received in reply to a preemptive discovery is used immediately when it is received. In addition, because of problems with stale paths in the DSR caching scheme (as was noted by other studies [11, 16]) DSR caches were disabled (only one active path maintained).

The NS-2 network simulator was used for the study, with CMU Monarch mobility extensions (some ad-hoc routing protocols; an implementation of IEEE 802.11 and a radio propagation model). NS-2's radio model was modified to introduce fading roughly according to a Rayleigh distribution including deep fades of up to 20dB. A low overhead stable power estimate algorithm was used; further details of the radio model can be found in [8]. Scenarios from the set studied in [4] were selected and simulated for DSR and PDSR with different levels of preemptive threshold. More specifically, we considered scenarios with a set of 35 nodes in an area of 700 meters by 700 meters. Nodes randomly pick a location within the simulated area and start moving towards it. Two mobility scenarios were considered: low (10 m/sec) and high (20 m/sec).

We considered three types of TCP traffic: (i) **telnet**: pairs of nodes simulate telnet sessions where small messages are exchanged with "human delays" between them. Here, latency is the main performance metric; (ii) **ftp**: a sender sends a continuous data stream to a receiver at the maximum rate possible for the duration of the experiment. In this case, throughput is the appropriate measure of performance; and (iii) **http**: several http servers and clients are initiated, with each client generating requests of exponentially distributed sizes to any of the servers. This case represents a middle ground between the first two cases.

Figure 3 shows sequence of events that occur when a path is broken. On this diagram, arrows that do not reach from one end of the diagram to the other represent dropped packets (or failed route requests). Note that the send times shown on the figure represent the TCP send times and not the actual time that the packet leaves the node. The first "send" of the packet fails because the route is broken. After the sender is notified of the path failure it has to wait until the TCP back-off timer expires; this takes 0.2 seconds in this case (the initial back-off timer value is twice the conservative estimate of RTT as measured by the coarse grain 0.1 second TCP timer). After the timeout, the packet is resent thereby triggering a route request. After the route reply is received, the packet is delivered. In this case, the ack also suffers a path failure. The packet is retransmitted 0.4 seconds after the first one (twice the previous back-off value). Finally, the ack for this retransmission causes a route discovery, after which the packet is acked. In this

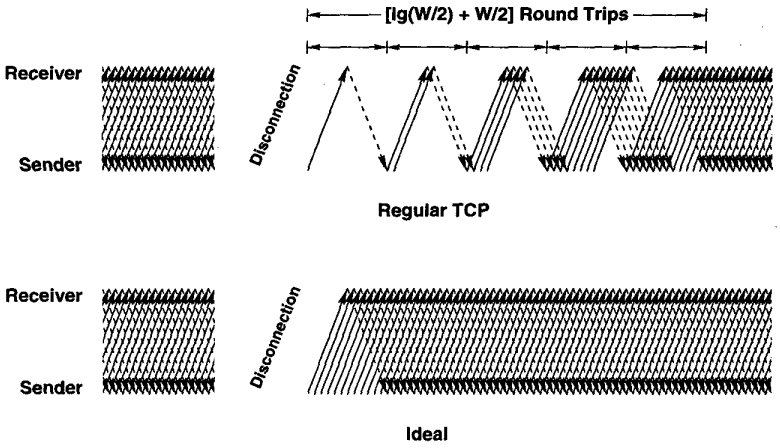

Figure 2. Illustration of down-time due to gradual window re-growth

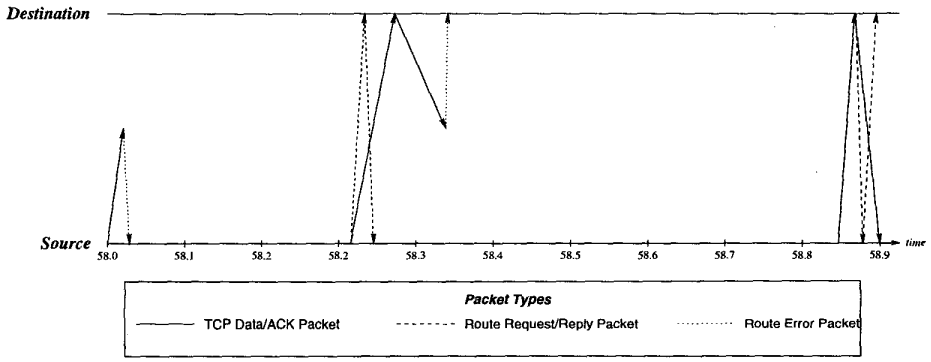

Figure 3. Effect of TCP Back-off

case, TCP back-off consumed about 0.6 seconds of the total delay (66%).

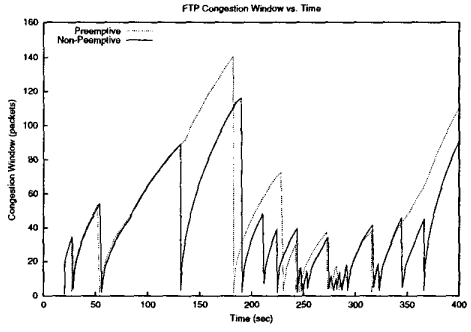

Figure 4. TCP Congestion Window

Figure 4 shows the congestion window size for PDSR with and without preemptive maintenance.[1] As can be seen in the diagram the congestion window size is, on average, higher for the preemptive case.

Figures 5 and 6 show the average packet latency in the telnet scenarios. Baseline DSR is the the left most point on each plot. The performance of baseline DSR is significantly lower than PDSR with no preemptive maintenance (preemptive factor 1.0) due to the bad caching behavior observed in DSR. The preemptive ratio was used as a simulation variable to isolate its effect; in [8], we discuss adaptively converging on optimal values for the preemptive ratio. Adding preemptive maintenance further improved the latency by up to 40%. An interesting observation about the behavior of baseline DSR is that the latency improved as the number of senders increased. A possible explanation is that the cache behavior is improved as nodes listen to traf-

[1] We chose preemptive ratio 1.0 rather than baseline DSR to eliminate the effects of stale cache entries observed in DSR thereby isolating the effect of preemptive maintenance.

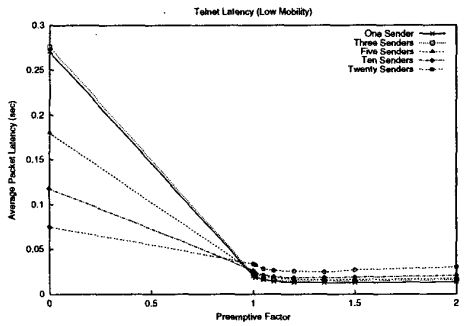

Figure 5. Telnet Latency – Low Mobility

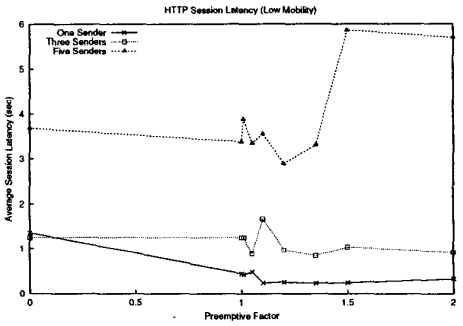

Figure 7. HTTP Latency – Low Mobility

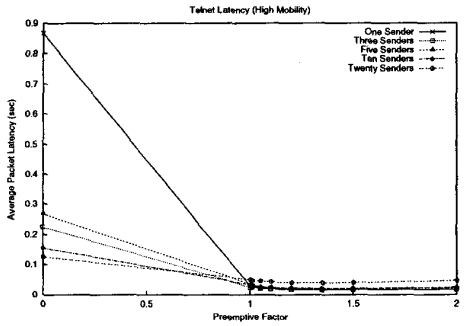

Figure 6. Telnet Latency – High Mobility

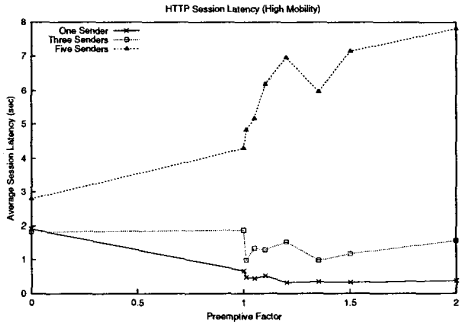

Figure 8. HTTP Latency – High Mobility

fic/route requests from the other paths – this makes their caches fresher. Thus, the caches benefit from a better sampling of the network state. The throughput was almost identical in all cases since the offered load is relatively light.

The **session latencies** for the http scenarios are shown in Figures 7 and 8. The session latency includes the time between sending a request and receiving the response. This time includes whatever processing time is necessary at the server; this time cannot be improved. The size of the response is exponentially distributed (but size is consistent across compared scenarios). At the optimal preemptive values, the improvement from preemptive maintenance over preemptive ratio of 1.0 is significant (around 40% in the best case). An exception is the case of 5 clients and servers in high mobility. This case corresponds to a very high network load, with 25 open paths in this small region. The preemptive overhead starts to significantly interfere with the data traffic. The behavior of baseline DSR is better than the telnet case; this can also be explained by the higher number of active paths allowing better updates of the cache states (making DSR's cache useful vs. PDSR which does not use any caching). We remark that turning off caching is not an inherent property of preemptive maintenance; our preemptive AODV extension does not interfere with AODV's caching behavior because AODV maintains significantly fresher paths than DSR.

The packet latency for the FTP scenarios are shown in Figure 9. The latency is marginally improved in the one sender case with preemptive maintenance. The small improvement relative to the http and telnet cases can be explained by the small number of packets affected by the path disconnects (relative to the very high ftp data traffic). FTP represents a very high load on the network – a single ftp connection has been observed to saturate a wireless LAN in experimental settings [24]. Thus, the higher frequency of path discovery in PDSR can increase congestion in this high load scenario. This is especially true as the number of active ftp connections increases beyond 1. Moreover, we observed a fairness problem in multiple-sender ftp case which skews these results; this problem is further addressed later in this section.

Figure 10 shows the throughput for the FTP scenario. The multiple-sender fairness problem again distorts the 3-sender case. A small improvement (around 10%) in throughput is achieved due to preemptive maintenance in the single sender case.

The overhead is shown in Figure 11 for the high mobility telnet scenarios. Note that this overhead is not a function of

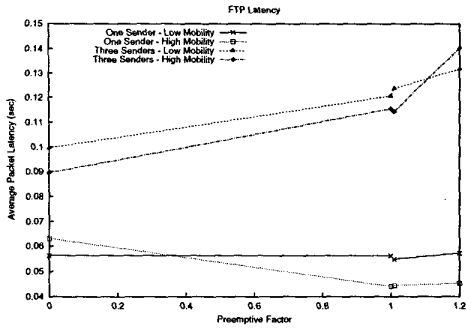

Figure 9. FTP Latency

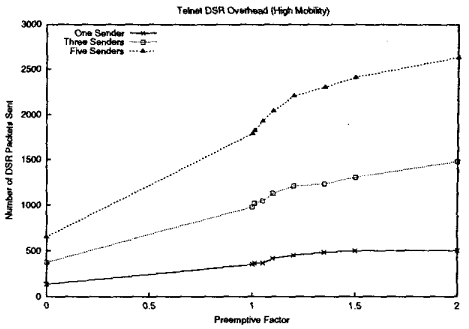

Figure 11. Routing Overhead – High Mobility

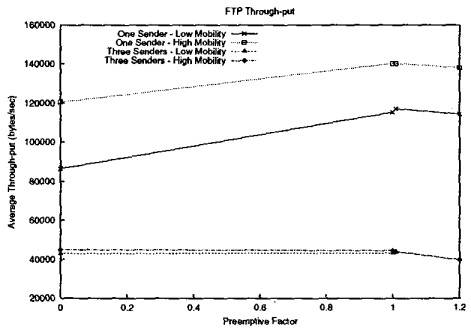

Figure 10. FTP Throughput

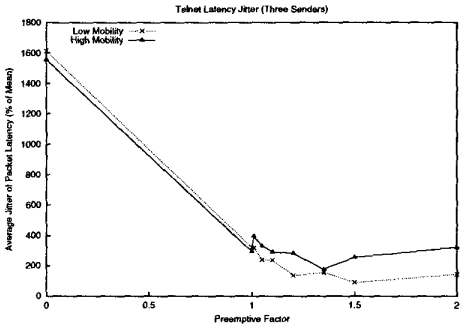

Figure 12. Packet Jitter

the traffic pattern; for a given mobility pattern, PDSR overhead consumes a constant and low portion of the bandwidth. The overhead is significantly higher than baseline DSR, but most of the increase is due to disabling the caching (the largest increase is seen going from DSR to the no cache version). Figure 12 shows the "packet jitter" (the standard deviation of the latency as a percentage of the average latency). Jitter is significantly reduced by PDSR because it reduces the number of broken paths and the associated delays.

Figure 13 shows the sequence numbers of TCP packets as a function of time for an FTP scenario with three sender-receiver pairs. There are extensive periods where some flows are not able to deliver any packets (the lower two flows on Figure 13) with at least one flow dominating the bandwidth. While TCP suffers from known fairness problems (favoring short RTT paths [12]), that alone is not sufficient to explain the gaps. Further analysis showed that, during these gaps, the sender does not have a route to the receiver. Furthermore, the route requests generated were not successful (no replies are received). This behavior is illustrated in Figure 14 which shows the events occurring during a typical inactive period.[2] Several failed route requests can be seen (with successive re-discovery attempts exponentially backed-off with an upper limit of 20 seconds, as per DSR). Meanwhile, TCP exponential retransmit backoff can be seen to be in progress. After examining our scenarios, we discovered that the network was **not** partitioned; path(s) were available for the blocked send-receive pair(s). We conjecture that some flows monopolize the bandwidth and block the route request packets generated by the other flows. The likely reasons for this unfairness are the MAC layer effects similar to those observed by Gerla et al [7]. However, their study considered scenarios with no mobility. Therefore, the large gaps in throughput observed in those experiments cannot alone explain the failures at the routing level – when a path was available, no gaps were observed. We are investigating this issue.

Consistent with observations in [11], the inactivity periods were observed even in the case of a single sender/receiver pair for the baseline DSR (Figure 15). In examining the detailed traces, we noticed that the inactivity was due to DSR repeatedly switching to paths from the cache that turn out to be stale. This also caused the TCP

[2] Again, on this diagram, arrows that do not reach from one end of the diagram to the other represent dropped packets (or failed route requests). Also, the send times shown on the figure represent the TCP send times and not the actual time that the packet leaves the node.

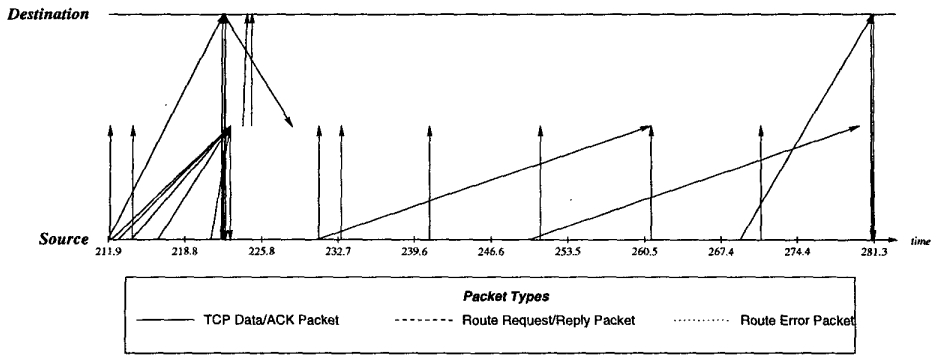

Figure 14. Behavior in Inactive Period

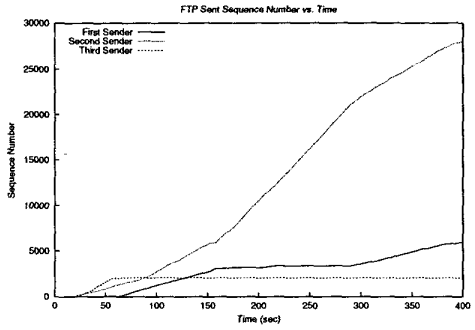

Figure 13. TCP Sequence Numbers

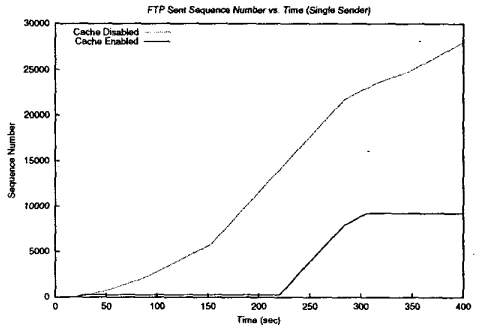

Figure 15. Single Sender Inactive Period

timer to back-off exponentially, further exacerbating the problem. As can be observed in Figure 15, once caching was turned off (PDSR with preemptive ratio 1.0), the inactivity periods were eliminated. This argues for active management of the cached paths.

5 Discussion and Conclusions

In ad-hoc networks, the topology of the network changes continuously – new links are established and existing ones broken as nodes move in and out of range of each other. Since the cost of detecting and recovering from a disconnected path is high, traditional routing protocols are inefficient in ad-hoc networks. In addition, the large disconnection recovery time may cause TCP to: (i) enter the lengthier stages of its exponential back-off; (ii) drop its congestion window. Consequently, when a path is eventually re-discovered, additional delays are incurred until TCP's back-off is complete. Furthermore, the TCP "slow start" causes additional inefficiencies as the window is gradually re-grown. Recently, we proposed a preemptive routing scheme where the number of disconnections is substantially reduced. In this paper, we compared the performance of TCP using this preemptive scheme (PDSR) to a conventional ad-hoc routing scheme (DSR).

For the telnet scenarios, the latency improved substantially. However, this might be due to the fact that the number of disconnections is relatively high with respect to the number of data packets; thus, the recovery cost is amortized over this small number favoring the preemptive scheme. Conversely, in the ftp case, there was no appreciable difference in latency (and a small improvement in throughput) of PDSR at optimal preemptive ratio relative to PDSR 1.0 (no preemptive maintenance). We attribute this to the following reasons: (i) For ftp, the number of packets transmitted is very high compared to the number of packets affected by path breaks. Hence, the effect of the large delays experience by few delayed packets is lost when averaged over the large total number of packets; (ii) Preemptive maintenance can add congestion to the network due to the more frequent route discoveries; (iii) preemptive maintenance can force us to accept longer paths (for higher signal quality); path length has been shown to be inversely proportional to TCP throughput [11].

The http traffic scenarios behave as expected: the performance enhancements are in between those for FTP and

telnet cases. In these scenarios, the communication pattern is all-to-all leading to a much higher number of active paths and potential congestion/interference. For ftp traffic, when there are multiple senders and receivers, it was observed that one flow can block others. It was observed that the route-query requests of the blocked flows were not successful, despite the existence of a physical path between the nodes. This suggests that the *fairness* issue occurs due to MAC layer effects: if multiple traffic flows are competing, fair arbitration of bandwidth is needed to share the network resources. We are currently investigating this issue.

References

[1] WaveLAN/PCMCIA Card User's Guide – Lucent Technologies.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. Katz. A Comparison of Mechanisms for Improving TCP performance over wireless links. In *Proceedings of ACM SIGCOMM'96, Palo Alto, CA*, pages 256–269, Aug 1996.

[3] S. Biaz and N. H. Vaidya. Distinguishing Congestion Losses from Wireless Transmission Losses. In *7th International Conference on Computer Communications and Networks (IC3N), New Orleans*, Oct 1998.

[4] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'98)*, Oct. 1998.

[5] K. Brown and S. Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM Computer Communications Review (CCR)*, 27(5), 1997.

[6] C. Chiang, M. Gerla, and L. Zhang. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In *Proceedings of IEEE SICON'97*, pages 197–211, Apr. 1997.

[7] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang. TCP over Wireless Multi-hop Protocols: Simulation and Experiments. In *Proceedings of IEEE ICC'99*, June 1999.

[8] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu. Preemptive maintenance routing in ad-hoc networks. In *Proceedings of the 7th International Conference on Mobile Computing and Networking (MobiCom'01)*, July 2001. (forthcoming).

[9] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments. In *Proceedings of the IEEE INFOCOM'2000, Tel-Aviv, Israel*, volume 3, pages 1537–1545, Mar. 2000.

[10] Z. Haas, M. Pearlman, and P. Samar. Zone routing protocol (zrp). Internet Draft, Internet Engineering Task Force, Jan. 2001. http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-ierp-00.txt.

[11] G. Holland and N. H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'99)*, Aug. 1999.

[12] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of the ACM SIGCOMM'88*, pages 314–329, Aug. 1988.

[13] P. Jacquet, P. Mulethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen. Optimized link state routing protocol. Internet Draft, Internet Engineering Task Force, Mar. 2001. http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-04.txt.

[14] P. Jacquet and L. Viennot. Overhead in mobile ad hoc networks. Technical report, Institut National De Recherche en Informatique Automatique (INRIA), June 2000.

[15] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks. Internet Draft, Internet Engineering Task Force, Mar. 2001. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt.

[16] D. A. Maltz, J. Broch, J. Jetcheva, and D. B. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, Aug. 1999.

[17] S. Murthy and J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal*, pages 183–197, Oct. 1996.

[18] V. Park and S. Corson. Temporally-ordered routing algorithm (TORA) version 1 functional specification. Internet Draft, Internet Engineering Task Force, Nov. 2000. http://www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-03.txt.

[19] G. Pei and M. Gerla. Fisheye state routing in mobile ad hoc networks. In *Proceedings of ICC'2000*, pages D71–D78, 2000. Internet Draft available at: http://www.ietf.org/internet-drafts/draft-ietf-manet-fsr-00.txt.

[20] C. Perkins, E. Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. Internet Draft, Internet Engineering Task Force, Mar. 2001. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt.

[21] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM Computer Communications Review*, pages 234–244, Oct. 1994. SIGCOMM '94 Symposium.

[22] S. S. Rappaport. *Wireless Communication Systems*. Prentice Hall, 1996.

[23] E. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, pages 46–55, Apr. 1999.

[24] D. Tang and M. Baker. Analysis of a local-area wireless network. In *Proceedings of the International Confernece on Mobile Computing and Networks (MobiComm'00)*, pages 1–10, 2000.

[25] N. Vaidya. TCP for wireless hosts tutorial. `http://www.cs.tamu.edu/faculty/vaidya/seminars/tcp-tutorial-aug99.ppt`.