Preemptive Routing in Ad Hoc Networks ⋆

Tom Goff [a] Nael Abu-Ghazaleh [b] Dhananjay Phatak [a]
Ridvan Kahvecioglu [b]

[a] *Electrical and Computer Engineering Dept.*
University of Maryland Baltimore County
Baltimore, MD 21250

[b] *Computer Science Dept.*
SUNY Binghamton
Binghamton, NY 13902

Abstract

Routing in Ad hoc networks is a challenging problem because nodes are mobile and links are continuously being created and broken. Existing on-demand Ad hoc routing algorithms initiate route discovery only ***after*** a path breaks, incurring a significant cost in detecting the disconnection and establishing a new route. In this work, we investigate adding proactive route selection and maintenance to on-demand Ad hoc routing algorithms. More specifically, when a path is likely to be broken, a warning is sent to the source indicating the likelihood of a disconnection. The source can then initiate path discovery early, potentially avoiding the disconnection altogether. A path is considered likely to break when the received packet power becomes close to the minimum detectable power (other approaches are possible). Care must be taken to avoid initiating false route warnings due to fluctuations in received power caused by fading, multipath effects and similar random transient phenomena. Experiments demonstrate that adding proactive route selection and maintenance to DSR and AODV (on-demand ad hoc routing protocols) significantly reduces the number of broken paths, with a small increase in protocol overhead. Packet latency and jitter also goes down in most cases. Because preemptive routing reduces the number of broken paths, it also has a secondary effect on TCP performance – unnecessary congestion handling measures are avoided. This is observed for TCP traffic under different traffic patterns (telnet, ftp and http). Additionally, we outline some problems in TCP performance in Ad hoc environments.

⋆ This work was supported by NSF Grants number ECS–9875705, CDA–800828, and EIA-9911099. Portions of this work appeared in the Proc. of the International Conference on Mobile Computing and Networking (MobiCom'01) and the International Conference on Parallel Processing (ICPP'01)

Email addresses: tgoff@umbc.edu (Tom Goff), nael@cs.binghamton.edu (Nael Abu-Ghazaleh), phatak@umbc.edu (Dhananjay Phatak), ridvan_kahvecioglu@non.hp.com (Ridvan Kahvecioglu).

1 Introduction

Routing in Ad hoc networks is a challenging problem because of node mobility and the scarcity of the bandwidth. Ad Hoc routing protocols fall into two categories: (1) Table-driven (proactive); and (2) On-demand (reactive). Table-driven protocols attempt to maintain consistent, up-to-date routing information among all nodes in the network. Thus, they require periodic route-update messages to propagate throughout the network. On the other hand, On-demand protocols initiate a route request flood whenever there is a path is needed between a pair of nodes. The advantage of the table-driven approach is that routes to any destination are always available without the overhead of a route discovery. In contrast, in On-demand routing, the source must wait until a route has been discovered, but the overhead is significantly less than Table-driven algorithms where many of the updates are for unused paths. A large number of routing protocols have been suggested including proactive (e.g., [13,21,26,29,31]), reactive (e.g., [23,28,30]) and hybrids (e.g., [17]). On demand protocols provide better routing performance (and potentially lower overhead) for networks were mobility is frequent; several simulation studies comparing Ad hoc routing protocols are available in literature [7,10,22].

An active path fails due to mobility when a pair of nodes forming a hop along the path move out of each other's range. In both types of routing algorithm, an alternative path is sought only *after* the current path fails. The cost of detecting a failure is high: several retries have to time-out before a path is "pronounced dead". Thus, when a path fails, packets experience large delays before the failure is detected and a new path is established. In this paper we investigate introducing preemptive route maintenance to Ad hoc routing protocols. More specifically, when two nodes, A and B, are moving out of each other's range, source nodes of active paths that use the hop A to B are warned that a path break is likely. With this early warning, the source can initiate route discovery and switch to a more stable path potentially avoiding the path break altogether.

Preemptive route maintenance implements a "soft-handoff" of active paths when they become endangered. Thus, it combines the best of on-demand and table-driven algorithms: the overhead is kept small since updates are only triggered by active paths that are likely to break, and hand-off time is minimized since corrective action is initiated early. Without preemptive maintenance, when a path break occurs, the connectivity of the flow is interrupted and a hand-off delay is experienced by the packets that are ready to be sent or in flight. This increases both the average and variance (jitter) of packet latency. Furthermore, this delay and the loss of any packets in flight causes TCP congestion avoidance mechanisms to take effect, further harming the performance of TCP connections. Our solution preemptively finds other paths, in many cases seamlessly switching to an alternative good path before a break, minimizing both the latency and jitter and avoiding inefficiencies due to unnecessary TCP backoff and congestion avoidance..

The effect of the suggested method is studied by extending Dynamic Source Routing (DSR) [23]; however, the proposed mechanism is general and can be used to enhance other routing algorithms. To illustrate this generality, we present preliminary results for preemptive extensions to AODV [30]. In addition, the method does not affect other Ad hoc routing optimizations such as location awareness [9,24] and query localization [12]. The signal strength is used as the preemp-

tive trigger; other warning criteria such as location/velocity and congestion can also be used. In fact, attributes such as path congestion, battery levels, and number of hops can be integrated into a "quality of path" measure to continuously maintain the "best" path. We also investigate the effect of using preemptive routing on the performance of TCP for different traffic scenarios. In the process, we encounter some inefficiencies in TCP performance that, to our knowledge, have not been discovered by other researchers in this area.

The remainder of this paper is organized as follows. Section 2 introduces the preemptive algorithm and discusses some possible optimizations to it. Section 3 discusses the generation of the preemptive warning and analytically derives the optimal signal power threshold and compares it to empirically observed values. Section 5 describes the extensions made to DSR to introduce preemptive maintenance. Section 4 discusses the effect of preemptive routing on TCP operation. Section 6 presents an experimental evaluation of the proposed mechanism. Finally, Section 7 presents concluding remarks.

2 Preemptive Route Maintenance

In traditional mobile and wired-network routing algorithms, a change of path occurs when a link along the path fails or a shorter path is found. A link failure is costly because multiple retransmissions/timeouts are required to detect the failure and a new path has to be found (in on-demand routing). Since paths fail so infrequently in wired networks, this is not an important cost. However, routing protocols in mobile networks follow this model despite the significantly higher frequency of path disconnections that occur in this environment.

We propose preemptive route maintenance extension to on-demand routing protocols. With preemptive maintenance, recovery is initiated early by detecting that a link is likely to break and finding and using an alternative path before the cost of a link failure is incurred. This technique is similar to soft-handoff techniques used in cellular phone networks as mobiles move across cells [32]. When extended with preemptive maintenance, an on-demand routing algorithm consists of two components: (i) detecting that a path is likely to be disconnected soon; and (ii) finding a better path and switching to it. Note the similarity to pure on-demand protocols: we replace path failure, with the likelihood of failure as the trigger mechanism for route discovery. Note that it is possible to add preemptive maintenance to table-driven protocols as well to avoid the cost of detecting a path failure.

A critical component of the proposed scheme is determining when path quality is no longer acceptable (which generates a preemptive warning). The path quality can incorporate several criteria such as signal strength, the age of a path, the number of hops, and rate of collisions. In this paper, we restrict the path quality (and hence the preemptive warnings) to be a function of the signal strength of received packets with the number of hops being used as secondary measure. Since most breaks can be attributed to link failures due to *node motion* in a typical network with mobility, the signal strength offers the most direct estimate of the ability of the nodes to reach each other. It is important that signal power fluctuations due to fading and other transient disturbances do not generate

erroneous preemptive warnings. The next section examines these issues in more detail describing our approach to mitigating the effects of transient signal fades.

Using preemptive route maintenance the cost of detecting a broken path (the retransmit/timeout time) is eliminated if another path is found successfully before the path breaks. In addition, the cost for discovering an alternate path is reduced (or eliminated) since the path discovery is initiated before the current path was actually broken. This can be expected to reduce the latency and jitter. Among the disadvantages, a higher number of path discoveries may be initiated since a path may become suspect but never break (for example, if the nodes change direction and move towards each other). However, if only high quality paths are accepted; they are likely to live longer reducing the number of re-discoveries needed.

3 Generating the Preemptive Warning

The preemptive warning is generated when the signal power of a received packet drops below a *preemptive threshold*. The value of this threshold is critical to the efficiency of the algorithm – if the value is too low, there will not be sufficient time to discover an alternative path before the path breaks. Conversely, if the value is too high, the warning is generated early with the following negative side-effects: (i) unnecessary discoveries: the frequency of the recovery action and the associated overhead increase; (ii) early switches to lower quality path: we may be forced to accept a path of a lower quality than the one we are currently using; and (iii) increasing the preemptive threshold effectively limits the range of the mobiles – a smaller range is now acceptable without generating a preemptive warning. If the threshold is too high, false network partitioning can also occur. Generating the preemptive warning is complicated due to fading that can cause sudden variations in the received signal power. The remainder of this section derives the criteria for selecting good threshold values under ideal conditions, then addresses link state estimation in the presence of channel fading and other random transient interferences.

3.1 The Preemptive Region

Figure 1 demonstrates the preemptive region around a source. For example, 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 this section, we develop an estimate for the optimal size of the preemptive region and, relate it to the signal power threshold under ideal conditions.

The recovery time from a broken path, $T_{recover}$, depends on the size and topology of the network, the load on the network as well as the path being recovered. In a realistic implementation, we assume that each node keeps a running estimate of this value (for example, as an exponential average of previous recovery times for all paths or more selectively, paths to a particular destination) and uses

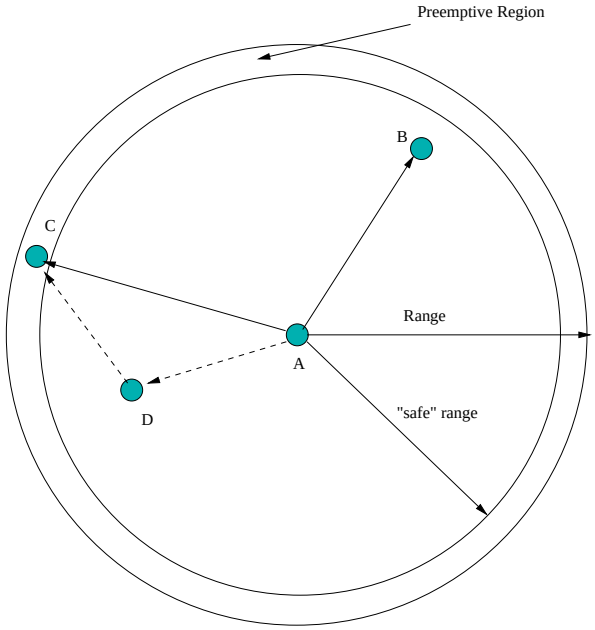


Fig. 1. Preemptive Region

that as its preemptive threshold. The optimal value for the signal threshold will warn the source $T_{recover}$ seconds before the path breaks; this allows just enough time to discover a new path. Hence, the warning interval T_w (which is the time between a warning and a break) should be set to $T_{recover}$.

Given two mobile nodes with a vector distance X between them, moving with vector speeds, V_1 and V_2, the distance between the two nodes is $X + t(V_2 - V_1)$. The time until the absolute distance between them becomes greater than the range of the source is a function of their relative location and velocity. In the worst case the sources are moving at their maximum speeds away from each other. This case can be used to derive a conservative estimate on the preemptive region.

Consider a typical land-based network where the maximum speed of a node is 20 m/s and a recovery time estimate of 0.1 sec (this value is used for illustration; it is in the typical range observed emperically). The preemptive region would start 4 meters from the maximum range; even if the two nodes are moving away from each other at maximum speeds (combined 40 m/s), the 4 meter distance will give the source the 0.1 second necessary to find a new path. If the nodes were actually drifting apart at a relative speed of 20 m/s, then 0.2 seconds would be available for the route discovery.

3.2 Relating the Preemptive Region to Signal Power

Because an explicit estimate of the preemptive region requires the nodes to exchange location and velocity information, we use the signal power of received packets to estimate the distance between them. Moreover, in a real environment, the distance between nodes does not correlate well with the received power due to obstacles, fading and interference. The recovery time can be related to the power threshold as follows. We consider devices operating in the ISM bands (such as Lucent

WaveLANs). The transmission power on such cards is restricted by the FCC to be less than 250 milliwatts at a distance of 3 meters from the transmitter (e.g., 280 milliwatts transmit power using omnidirectional antennas on the WaveLAN cards [1]). The signal power drops such that

$$P_r = \frac{P_0}{r^n} \tag{1}$$

at a distance r from the transmitter, where P_0 is the transmitted power and n is typically between 2 and 4.

The signal power at any point is the sum of the main signal transmitted by the antenna in addition to components of the signal that reflect off-of the surrounding features (multipath effect) [32]. In open environment, the main secondary component is the strong reflection of the transmitted signal from the ground (two-ray propagation model). Equation 1 represents an idealized model for the channel. Usually, $n = 2$ near the source until a certain distance where n becomes 4. Such an equation cannot account for channel fading in general (which can cause sharp and sudden fluctuations in signal power) because fading is highly dependent on the specific surrounding terrain; we shall consider stable power estimates in the presence of fading in the next subsection.

We assume the $\frac{1}{r^4}$ drop in signal power with distance model [4] throughout the preemptive region (since the preemptive region is near the maximum range of the devices). More specifically,

$$P_{received} = \frac{P_0}{r^4} \tag{2}$$

where P_0 is a constant for each transmitter/receiver pair, based on antenna gain and height. The minimum power receivable by the device is the power at the maximum transmission range, P_{range} is $\frac{P_0}{range^4}$. This value is characteristic of the device (e.g., $3.65 \cdot 10^{-10}$ Watts for WaveLANs [1]). Similarly, the preemptive signal power threshold – it is the signal power at the edge of the preemptive region. In addition, for a preemptive region of width of w, the signal power threshold is

$$P_{threshold} = \frac{P_0}{r^4_{preemptive}} \tag{3}$$

Note that $r_{preemptive}$ is equal to $(range - w)$ where $w = \text{relative\_speed} \cdot T_w$. The preemptive ratio, δ is defined as

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

For example, WaveLAN cards have a range of 250 meters in open environments in the 900MHz band [1]. The preemptive ratio for a preemptive region of width 4 meters is $(\frac{250}{(250-4)})^4 = 1.07$. This value corresponds to a signal threshold of $1.07 \cdot P_{range} = 3.9 \cdot 10^{-10}$ Watts.

3.3 Mitigation of Channel Fading and other transient interferences

In practice, the received signal power may experience sudden and substantial fluctuations due to channel fading, multipath effects and doppler shifts [32]. There is a concern that these fluctuations may trigger false preemptive route warning, causing unnecessary route request floods. The overhead of unnecessary route request floods can have adverse effect on performance as the network is saturated. Furthermore, route switches to lower quality routes may be initiated.

The problem is not as significant as may appear at first. For most networks, the preemptive region is so narrow that the probability of a transient fade changing the received power to be within the preemptive range is tiny. Moreover, there are established mechanisms to produce stable power estimates that were developed for power control in in cellular networks. For example, maintaining an exponential average of the signal power (rather than triggering the mechanism based on a single packet) can be used to verify that the signal power drop was not due to fading. However, if the traffic is bursty or infrequent, the preemptive region may be fully crossed by the time enough packets are received to drop the average below the threshold. Alternatively, quicker power estimates can be achieved by sending a warning whenever the instantaneous power drops below the threshold, and checking the warning packet received power when it is received by the source. If the warning packet power is also below the threshold, there is a good probability that the warning is real. More generally, a more stable average can be generated by having any number of ping pong rounds (these "query" packets are of minimal size) to check if the warning is true. The number/duration of pings should be related to the expected duration of fading phenomena – this depends on the channel and the surrounding environment. This is the approach we have used in our experiments. The details of this implementation are presented in Sections 5 and 6.

The two mechanisms can be mixed by using the exponential average if the packet reception rate is high, defaulting to ping-pong rounds if it is not. Finally, for mobiles equipped with GPS systems, the warning packet can register the location/velocity of the mobile so that the source can compute whether it is truly moving out of range or not. This approach is interesting since cellular phones will be soon required to provide location information due to the FCC's "911" mandate [14]. The source can also apply a dead reckoning calculation (using its own location and velocity information) to estimate when the path will be broken and when the optimal time to start corrective action is. However, we believe that the use of geographical information is idealized due to the effect of the environment (non-uniform topology and obstacles).

Another potential problem occurs when the transmission rate along a path is low or bursty. A node may move into the preemptive region during a quiet interval. No warning will be generated until the next packet is sent (because route warnings are triggered only when the signal strength of a "received packet" falls below the threshold). By that time the path may be already broken or not enough time is left for a route discovery. In order to avoid this situation, a null (empty) packet can be sent along idle but active paths. The period of this ping can be related to the width of the preemptive region to balance overhead against recovery time. The preemptive region can be extended to account for the sampling period effect in such flows.

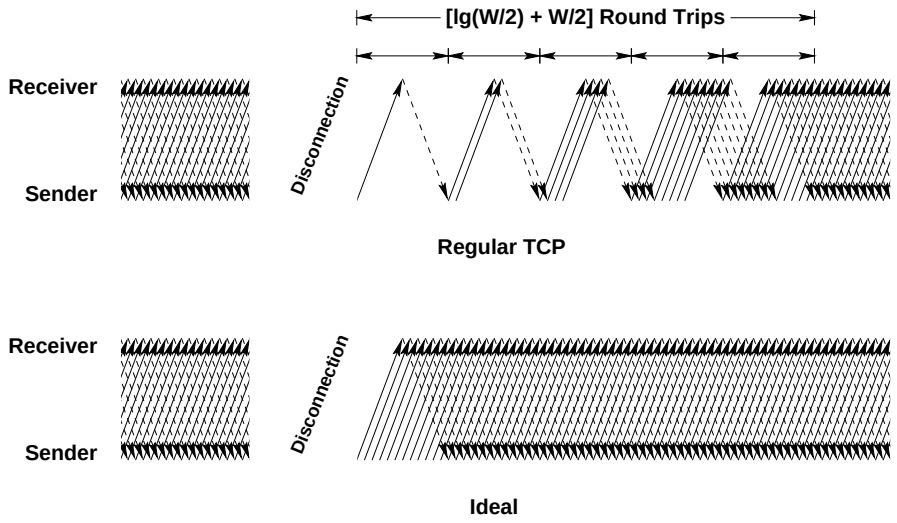


Fig. 2. Illustration of down-time due to gradual window re-growth

For example, consider the case where the sources generate traffic at a fixed rate (CBR model). The inter-packet interval is therefore

$$T_{\text{pkt}} \approx \frac{1}{\text{CBR}}. \tag{5}$$

No preemptive warning will be generated unless a packet happens to be received when a node is in the preemptive region (see Figure 1). This indicates that to be able to "sample" the preemptive region, the CBR should satisfy the constraint:

$$\begin{aligned} T_{\text{pkt}} &= \frac{1}{\text{CBR}} < \text{time to traverse the preemptive region} \\ &= \frac{w}{\text{average relative speed}} = T_{recover} \\ \text{or} \quad \text{CBR} &> \frac{1}{T_{recover}} \end{aligned} \tag{6}$$

4 TCP in Ad hoc Networks

Packets in flight along the paths that break often get dropped (unless some form of intermediate salvaging is applied). In addition, a period of disconnection is suffered while an alternative path is found. These two factors can cause significant degradation of TCP performance especially if the mobility is high [18]. 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 [20]). The disconnection time may result in multiple failed retransmits, and a large exponential backoff on the timer. Thus, when the path is eventually established, a long timeout value may be suffered before packet sending resumes.

Packet losses due to mobility and transmission errors cause TCP to perform poorly in wireless environments [5,6]. 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.

In a last-hop environments, mobility causes packets to be lost due to hand-offs as a mobile node moves out of range of a base station and into the range of another [11,33]; packets lost during such transitions also initiate TCP's congestion avoidance. Several researchers have addressed optimizing TCP in wireless last-hop environments [5,11,33,35]. In ad hoc networks, if any of the hops constituting the path fails due to node mobility, TCP packets will be dropped and congestion avoidance will be initiated. 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. Other work focused on reducing the false congestion avoidance effect in ad hoc networks by using techniques such as explicit loss notification and randomized congestion avoidance [8,16,18]. Since preemptive maintenance eliminates most disconnections, there reason to believe that it will achieve a similar effect to these optimizations on TCP performance. This improvement is beyond that obtained from better routing and does not require changes to TCP.

5 Preemptive Route Maintenance Case Studies

As was noted previously, preemptive maintenance can be added to any Ad hoc routing protocols (we have only investigated on-demand ones). In order to evaluate preemptive route maintenance, the Dynamic Source Routing (DSR) protocol and AODV protocols were modified to incorporate preemptive maintenance. We call the modified versions Preemptive Dynamic Source Routing (PDSR) and Preemptive AODV (PAODV), respectively. We focus on DSR for the sake of brevity and to allow detailed analysis. AODV is used to illustrate that the results are not specific to DSR. In this section we describe the modifications made to DSR. The channel fading model and the algorithm for stable estimate of signal power are also described.

5.1 Preemptive Warning Generation

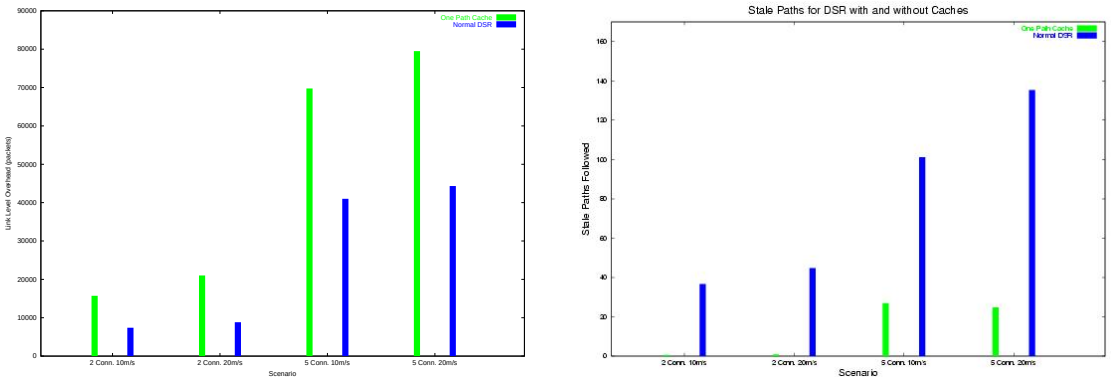
If the received signal strength is below $\delta \cdot P_{range}$ the node which received the packet with sub-threshold signal strength starts pinging the adjacent node (which transmitted the packet that was received with below-threshold power). Upon receiving the ping, a node immediately responds with a pong (which, like a ping is also a minimum sized packet used to "probe" the link state). Upon receiving the response, the original node (which received the packet with low power) pings the adjacent node again and receives a pong again. n such ping-pong responses are monitored for signal strength. During this monitoring period if the total number of bad packets received is above a certain threshold value k then a route warning is sent back to the source. If there is no response to a ping within a timeout-period $T_{\text{ping-timeout}}$ then a route warning is sent back. Thus, the total length of time window in which the link state is monitored can be as large as $n \times T_{\text{ping-timeout}}$. The original NS2 code has been modified to incorporate $n, k, T_{\text{ping-timeout}}$ and a few other parameters as inputs.

Upon receiving a route warning, the source initiates a route discovery in order to find a higher quality path to switch to. In experiments, it was observed that multiple packets caused repeated "route-warning" messages from the same link. To prevent this behavior, a field was added to the DSR header which was designated as "signal-strength-threshold." If any node receives a packet with signal strength below this value, it initiates link-monitoring (via the ping-pong probes) and if necessary, sends a route warning back to the source. Initially the source sets signal-strength-threshold to $(\delta \cdot P_{range})$. Upon receiving the first route warning, the subsequent packets are transmitted with a signal-strength-threshold of 0 to prevent repeated route warnings. Note that this mechanism does not require intermediate nodes to store any additional path information. It is significantly more flexible than having static or locally generated threshold and there is little additional overhead.

5.2 Route Cache Behavior

In order to minimize the discovery time, routing algorithms provide route caches that keep discovered/overhead paths for future use. For example, the current DSR implementation provides two caches: the primary cache is intended to store routes that were learned first hand, while the secondary cache stores routes that are "overheard" by snooping. The current implementation of DSR does not discriminate between these caches, it simply searches both caches when looking for a route. Moreover, paths that reside in caches are not "aged"; thus, a path in the cache may become invalid by the time it is called upon. In addition, nodes may reply from their caches when a path request is received, propagating these stale paths.

To illustrate the effect of stale paths, we conducted a simple experiment using the NS-2 simulator. The experiment consisted of 35 nodes in a 700x700 area using CBR communication. Each node randomly picks a point in the simulation area and starts moving towards it; when the point is reached, another is picked with no pause time. Scenarios with two different speeds and two different numbers of communicating node pairs were studied. We studied two configurations of DSR: conventional DSR and DSR with the route cache size reduced to 1 entry per path. Figure 3(a)



(a) Overhead (b) Stale Paths

Fig. 3. Effect of DSR Cache

shows the overhead for the two cache configurations. Obviously, with a single cache entry per path, a route flood is required after every path break. However, since each path is obtained in response to a route search, very few stale paths are encountered. A stale path is defined as a path that breaks without successfully delivering even a single data packet. Such a path is due to stale information in the cache. Figure 3(b) shows the number of stale paths for the same scenarios. Note that for regular DSR, the number of stale paths is quite high in a short simulation time (400 sec). These conclusions are consistent with those of other studies [18,25].

Due to the high ratio of stale cache paths, we disabled the DSR caches (both local and cache replies). Empirically, we saw that the cost of a path discovery is significantly lower than the time to recover from a path disconnect. Only one path is kept for each active destination. In addition, in our algorithm, we bypass the ring 0 search (a localized query of immediate neighbors) [10,23] since the likelihood of finding a path is small with cache replies disabled, allowing us to save the timeout cost on this phase. We note that this aspect of the implementation can be significantly optimized (reducing both the number of discoveries and the cost per discovery) by using effective caching with a higher success rate. For example Hu and Johnson analyzed the effect of several cache parameters [19] on cache operation. Other work explored localized query floods based on previous path [12] or location information [24] to reduce query costs. In other work, we suggested active cache optimizations to raise the quality of the paths in the cache [27]. To demonstrate that the results are not dependent on turning the caching off, we show the results for AODV [30] with no modification to its cache operation because its cache management scheme results in high success rates on recovered paths.

5.3 Path Query Implementation

When a query flood is generated in response to a warning on a given path, the first path received is immediately used. Any successive paths found by the query flood are compared against the path in use and the shorter path is picked. Improvements in this aspect of the algorithm that discriminate

based on other factors (for example, to pick the less congested path) are also possible.

The flood is generated while the original "bad" path is still active; we might receive the original path (or another path) which is about to be disconnected soon. In order to discover only good quality paths, the query is tagged with a minimum desired threshold on each link of the path. This threshold is currently set as the preemptive threshold, but can be different (ideally, it would account for the time of the flood + the time for another discovery if it is about to go into the preemptive region itself). Intermediate nodes that receive a path request packet with a signal power below the threshold, act as if they did not receive that packet. This has the effect of "limiting the range" of the nodes, so if all the available paths are below the desired threshold, we will discover no alternative paths. A more efficient implementation would have the route reply phase of the query keep the minimum power on the reply route. The source would then select the best path available, in case none are above the desired threshold.

6 Experimental Study

An extended version of UCB/LBNL network simulator (NS-2 [2]) was used for the experimental study. NS-2 is a discrete event simulator that was developed as part of the VINT project at the Lawrence Berkeley National Laboratory. The extensions implemented by the CMU Monarch project [3] enable it to simulate mobile nodes connected by wireless network interfaces. The NS-2 DSR protocol implementation was extended with preemptive maintenance as per the description in Section 5.

To simulate the effects of fading and other transients, we modified the channel propagation model provided in NS-2. Instead of marking the packet as "bad" (i.e., as having an error) as is done in NS-2, we decrease the power level to model the effects of fading according to a statistical distribution that approximates Rayleigh fading. The error model assigns one of two states to each link: good or bad. If the link is in the good state, the received power (calculated by the original NS-2) is left unchanged. If the link is in the bad state then the received power is decreased by a multiplicative factor which is a uniformly distributed random number (real) between 2 and 100. This model approximates a typical fading scenario such as the one illustrated in [32, page 71] and accounts for **deep fades** (up to 20dB, i.e., a signal strength reduction by a factor of 100).

The length of time the link remains in each state is determined by an exponentially distributed random variable. The mean length of stay in the good state (mean of the exponentially distributed random variable governing the good state) was set to 20,000 packets. Likewise, the mean (average) stay in the bad state was set at 2 packets. This corresponds to a mean packet error-ratio (BER) is 10^{-4}. We believe this is a good approximation of a moderate quality wireless channel (i.e., BER 10^{-6} and bursty errors). Note that not every fade results in an actual error (dropped packet) because if the initial strength is high, reduction by a factor of up to 100 may still leave it above the detection threshold.

For each packet received with a signal strength below the preemptive threshold, we start pinging the

previous-hop node as described above. The timeout period associated with each ping is $T_{\text{ping-timeout}}$ = 0.04 seconds. If no response is received within this time after sending a ping (any ping) a route-warning is initiated. Upto 3 pings are sent (i.e., $n = 3$). Thus, in effect, a window of upto 3×0.04 = 0.12 seconds is monitored after transmitting the first ping. If 3 packets (pong responses, data packets, or routing packets) with a strength below the preemptive threshold are received within this window, then a route warning is sent out (there can be intervening packets which are received with a signal strength above the threshold, we count 3 sub-threshold packets within the window to trigger a route warning). These values were chosen arbitrarily and can benefit from empirical tuning Note that since the average duration of the bad state due to temporary fadings is 2 packets, we require 3 low-strength packets (a number bigger than the average fade interval of 2 packets) to indicate that the link is bad with sufficient consistency to warrant a route warning. In general, depending upon the congestion, traffic rate (CBR), observed link state and other variables, it is possible to dynamically adjust the number *n* of pings sent, the timeout period, the critical (threshold) number of packets with sub-threshold power after which a route warning is generated, etc. In the set of experiments described in this paper, however, these parameters were inputted at the start of each run (and were not dynamically adjusted). Such dynamic adaptation is a topic for future work.

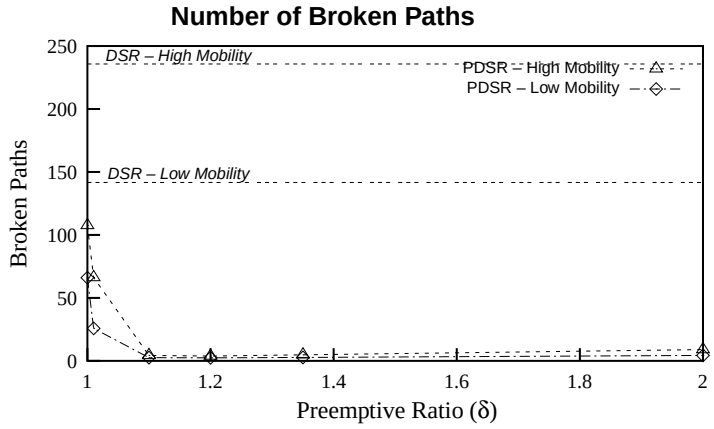


Fig. 4. DSR Broken Paths

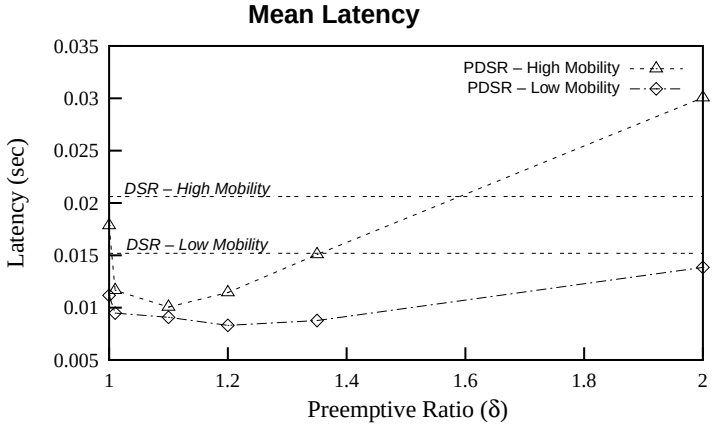


Fig. 5. DSR Packet Latency

For an unbiased comparison, scenarios similar to those previously studied [10,25] were selected and simulated with and without proactivity. 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. There were 10 source nodes transmitting to 10 destination nodes at a Constant Bit Rate (CBR) with 5 packets/sec. In addition, two mobility scenarios were considered: (i) low mobility (max. node speed 10 m/s); and (ii) high mobility (max. node speed 20 m/s). Note that both the selected CBR values represent significant load on the network given the large number of nodes sharing a relatively small area – the immediate range of a node ($\pi \cdot range^2$) represents nearly 40% of the whole area.

6.1 Preemptive Routing Analysis

The direct effect of preemptive routing can be seen by examining the number of broken paths (Figure 4). The horizontal lines on each figure correspond to baseline DSR (with no modifications whatsoever) under high mobility and low mobility. The number of broken paths is shown as the preemptive ratio (δ) is increased. **Note that he case with $\delta = 1$ corresponds to *non-preemptive PDSR* which is equal to DSR with the modified cache behavior described in Section 5**. Thus, non-preemptive PDSR results isolate the effects of the cache modifications from those due to proactivity. At the knee of the curve, the proactivity threshold provides sufficient time to initiate rediscovery; lower values cannot avoid all path breaks while higher values restrict the effective range and increase the overhead unnecessarily. We note that the optimal preemptive threshold increases with mobility and CBR rate, to allow more time to recover given the higher speed of the nodes and longer latency respectively.

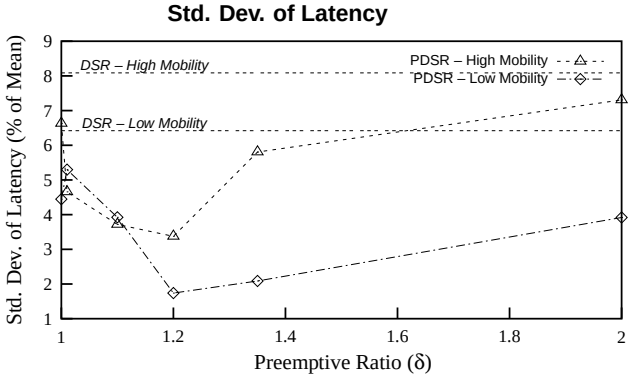


Fig. 6. DSR Jitter

It is evident that preemptive routing drastically reduces the number of broken paths, eliminating most of them under the low CBR case. Under high CBR, collisions are more frequent and large variability in latencies can be experienced due to the exponential back-off initiated when a collision occurs [15]. Under these conditions, the preemptive warning may not provide sufficient time for path discovery. Accordingly, while proactivity significantly reduces the number of broken paths, the number remains higher than the low CBR case. Proactivity is even more successful for the high mobility scenarios where the path break frequency is higher. Very high proactivity thresholds are inefficient (a proactivity threshold of 4 corresponds to limiting the effective range by 28% and coverage area by 48%!). In practice, the useful range of proactivity should be restricted well below

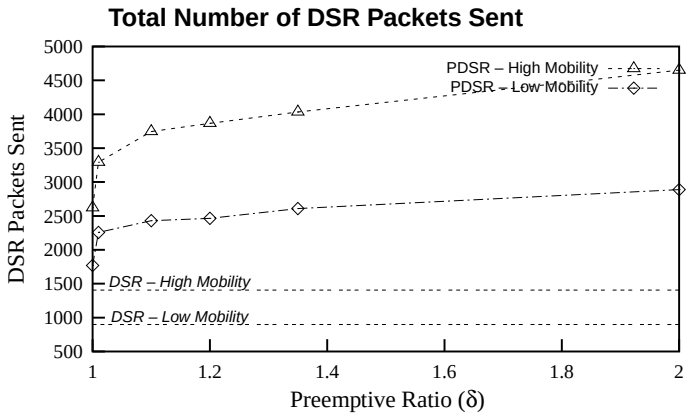


Fig. 7. DSR Overhead

$\delta = 2$. An alternative measure of the effectiveness of proactivity is the total recovery time experienced by broken paths; we observed this value going down drastically as well under preemptive routing implementations.

Figure 5 shows the mean overall packet latency. Significant reductions in latency can be observed in the best case (e.g., at $\delta \approx 1.2$ which is close to the optimal value of preemptive threshold). The reduction in latency is largely due to avoiding the delays associated with path breaks. In fact, latencies on established paths can be expected to increase slightly because proactivity limits the effective range slightly – an "optimal" path with a link below the proactivity threshold is rejected in favor of a longer path with higher quality links. Figure 6 plots the standard deviation of the latency (jitter) as a percentage of the mean latency. Please note that these jitter values must be taken with a grain of salt since variance in the delays is expected due to variations in path length and due to congestion. Ideally, the jitter would be measured on a per-connection basis. However, by eliminating the very high delays for disconnected paths, the overall jitter values are improved.

Figure 7 shows the overhead of PDSR compared to DSR. While the overhead is higher, we note that most of the overhead was experienced also by the non-preemptive version of PDSR ($\delta = 1$, corresponding to DSR with caching disabled) and increased only slightly for proactivities in the practical range. This indicates that most of the overhead is due to the modified cache behavior (no path replies from cache) and not due to the addition of proactivity. There is reason to believe that the overhead will drop with an effective caching strategy.

To illustrate that the results are not specific to DSR, we incorporated preemptive maintenance into AODV, a distance vector based routing algorithm for ad hoc networks. The modifications we had to make for AODV were somewhat different than those incorporated for DSR. Specifically, data packets do not carry the full source in their header; rather, normal distance vector routing is implemented. Thus, the source is not available in the network header. Peeking into the transport header can be done, but is not a clean solution. So, we included the source address in the packet to allow the warning to occur. Finally, AODV cache behavior ensures that the caches are fresh with a high probability – we did not have to turn off the caches as we did with DSR.

Figure 8 and Figure 9 show the number of broken paths and the packet latency for the same CBR

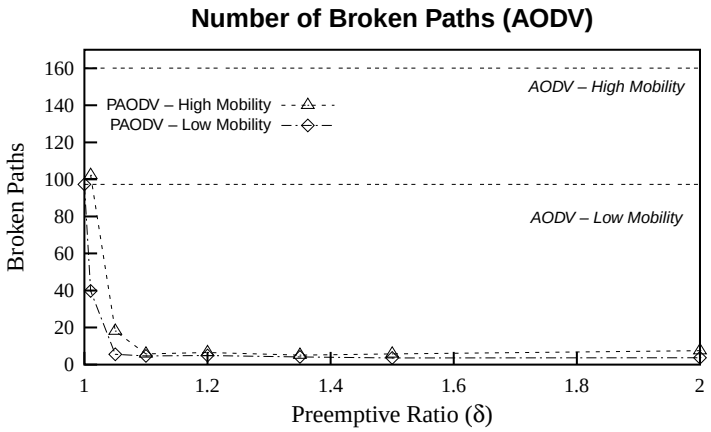


Fig. 8. AODV Broken Paths

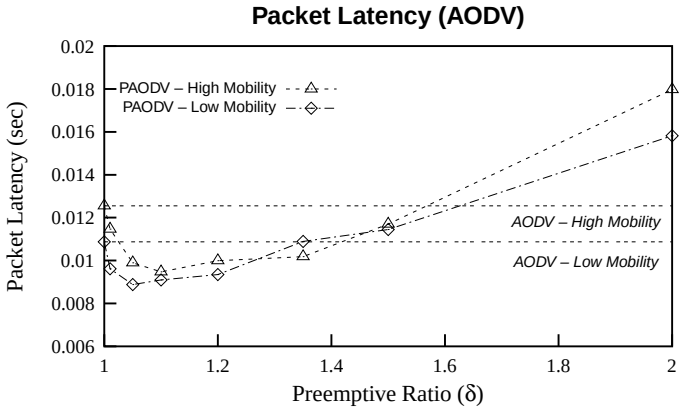


Fig. 9. AODV Packet Latency

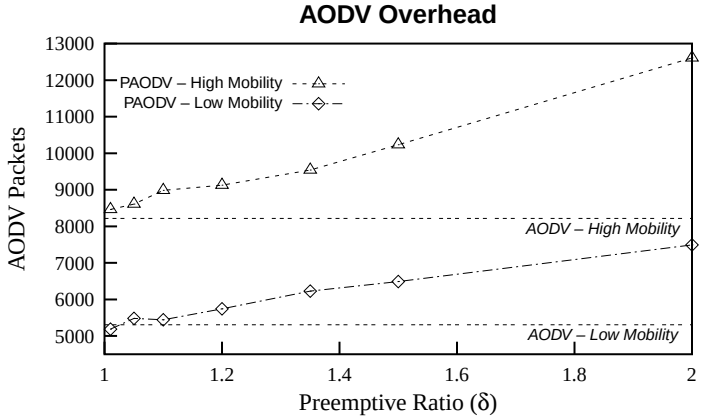


Fig. 10. AODV Overhead

traffic scenarios. Again, the number of broken paths is drastically reduced, and the latency is improved by up to 30% in the best case. We note that the number of broken paths for baseline AODV is less than that for baseline DSR (potentially due to the better caching scheme). Figure 10 shows number of AODV packets introduced. As can be expected, the overhead increases with preemptive routing (due to searches provocatively started that prove to be unnecessary). However, we note that

the increase is significantly lower than the increase in the DSR case. This strengthens the claim that the DSR increase was mainly due to turning off the caching.

6.2 TCP Analysis

We considered three TCP traffic scenarios: (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.

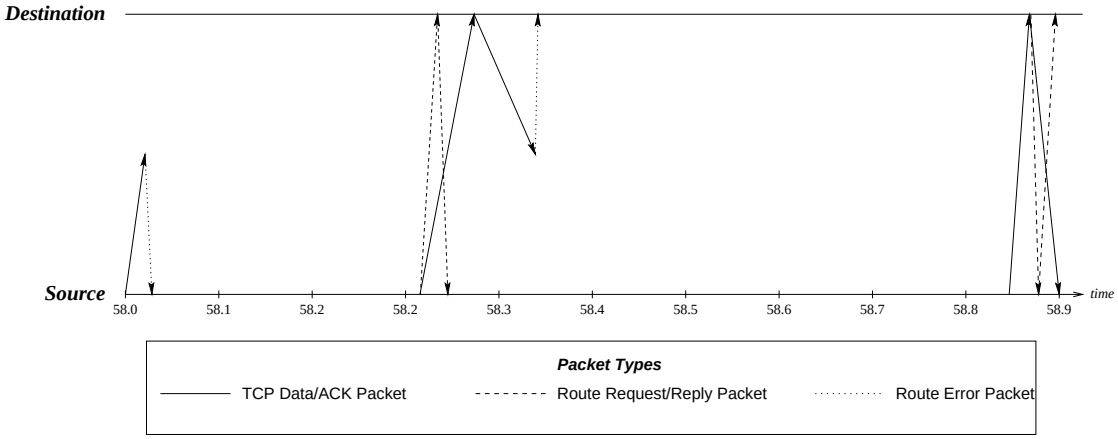


Fig. 11. Effect of TCP Back-off

Figure 11 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 re-sent 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 case, TCP back-off consumed about 0.6 seconds of the total delay (66%).

Figure 12 shows the congestion window size for PDSR with and without preemptive maintenance. 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. As can be seen in the diagram the congestion window size is, on average, higher for the preemptive case.

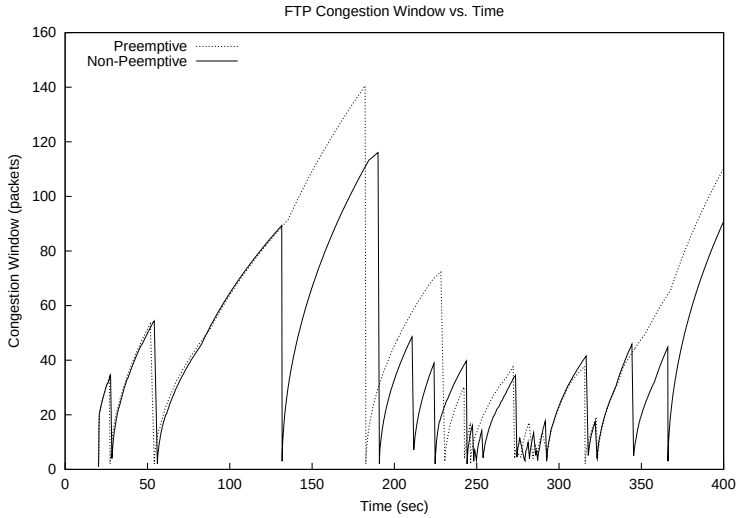


Fig. 12. TCP Congestion Window

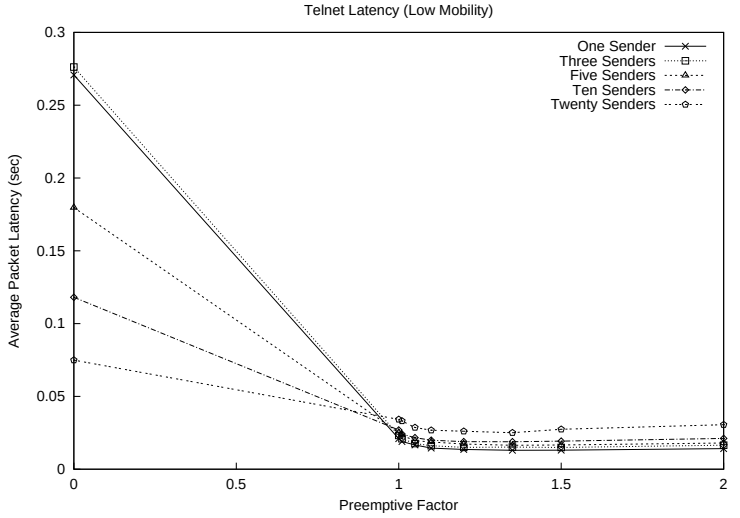


Fig. 13. Telnet Latency – Low Mobility

Figures 13 and 14 show the average packet latency in the telnet scenarios. Baseline DSR is the the left most point on each plot. Again, the performance of baseline DSR is significantly lower than PDSR with no preemptive maintenance due to the bad caching behavior observed in DSR. 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 traffic/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 15 and 16. 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

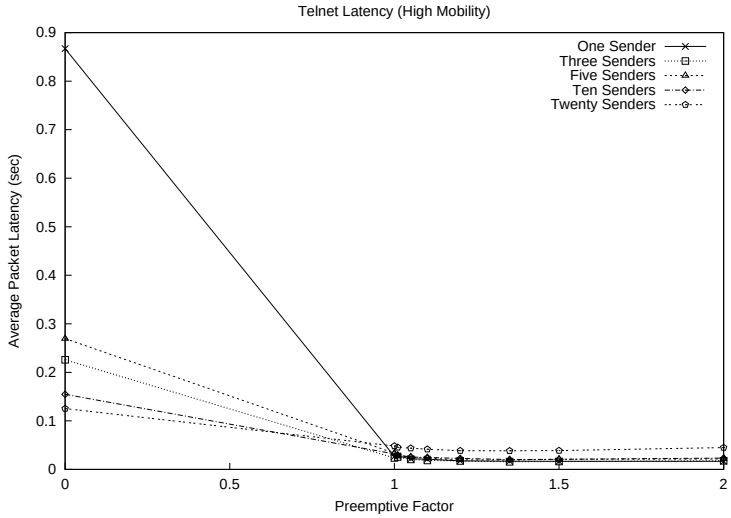


Fig. 14. Telnet Latency – High Mobility

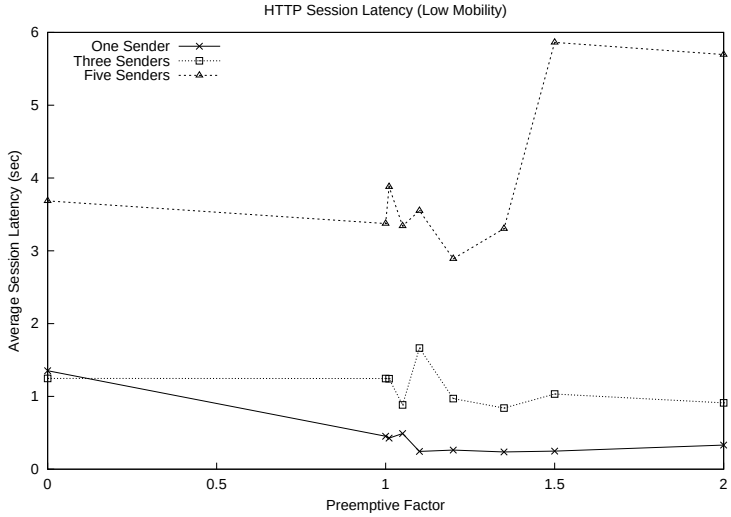


Fig. 15. HTTP Latency – Low Mobility

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). It bears repeating 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 17. The latency is marginally improved in the one sender case with preemptive maintenance. The small improvement relative to the

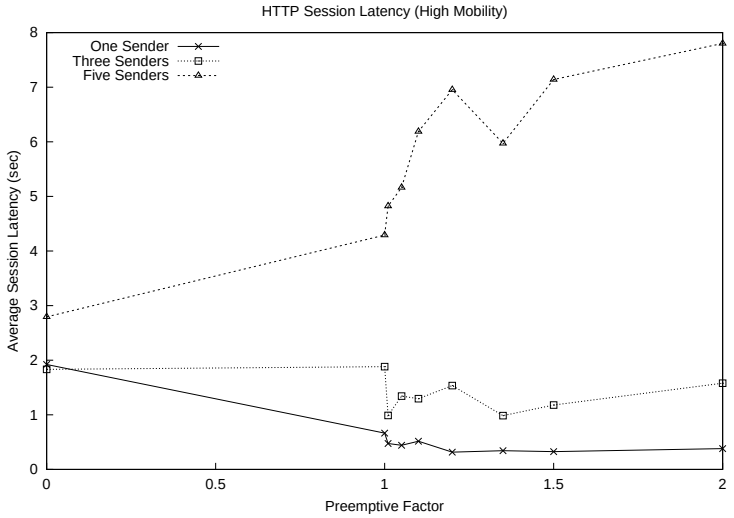


Fig. 16. HTTP Latency – High Mobility

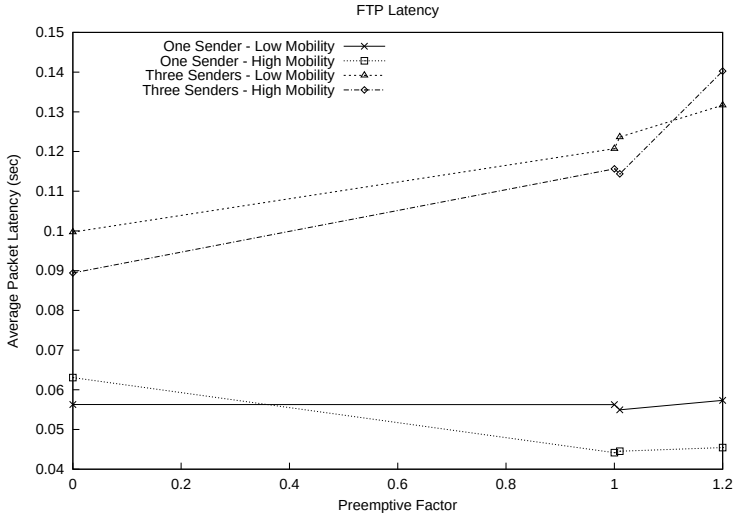


Fig. 17. FTP Latency

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 [34]. 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 18 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. In [27], we noticed that TCP performance with a single cache entry is very good because it periodically finds a new fresh path, while old paths in the cache become stale. The problem is especially bad for TCP since having a number of stale

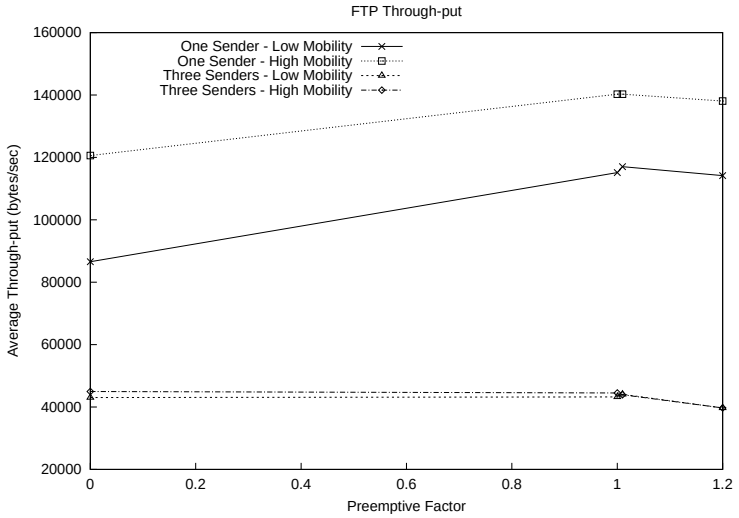


Fig. 18. FTP Throughput

paths in series can significantly raise the backoff timer. Moreover, often the same stale paths have to be expired by the destination (also in series) once a valid path is found by the source and the TCP packet is received in order to find a return path for the ACK packet. Several active cache optimizations were investigated resulting in over 50% improvement in TCP performance. We are currently investigating the combined effect of active route cache optimization with preemptive maintenance.

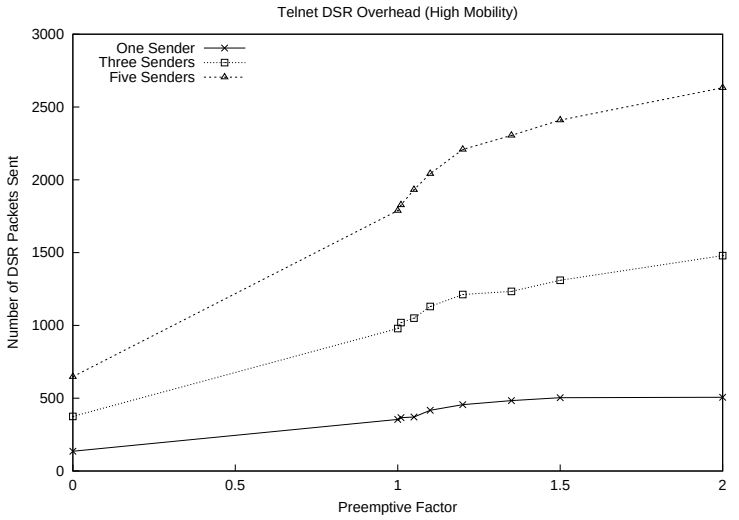


Fig. 19. Routing Overhead – High Mobility

The overhead results (Figure 19) were similar to those observed in the CBR scenarios. Figure 20 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 21 shows the sequence numbers of TCP packets as a function of time for an FTP sce-

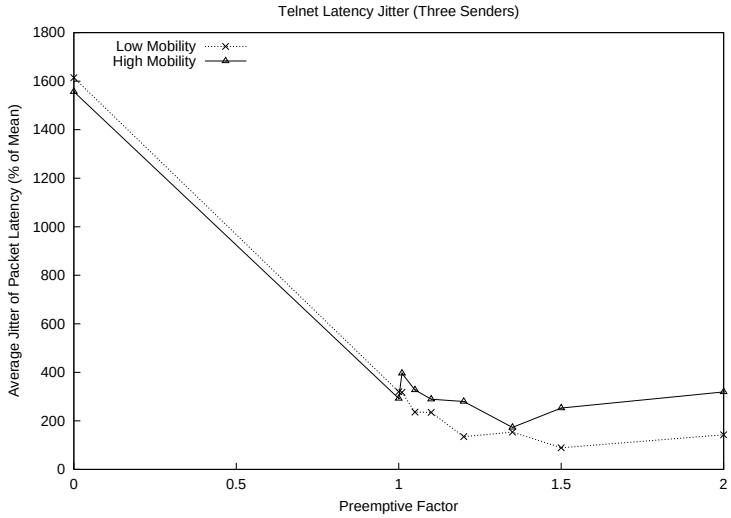


Fig. 20. Packet Jitter

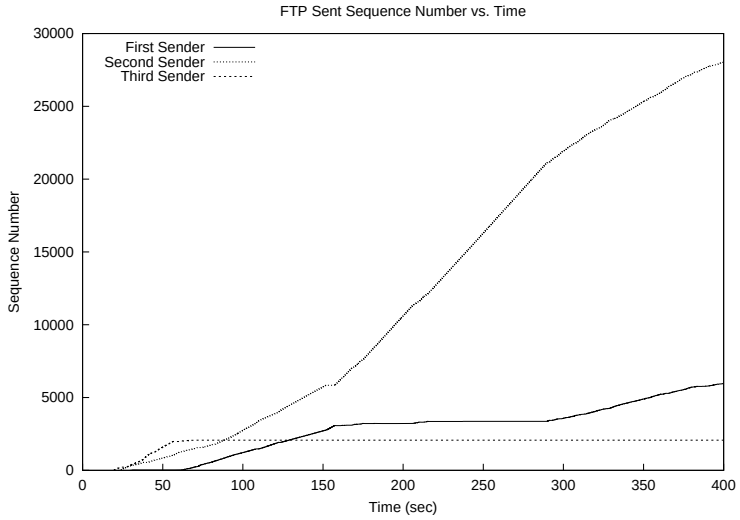


Fig. 21. TCP Sequence Numbers

nario 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 21) with at least one flow dominating the bandwidth. While TCP suffers from known fairness problems (favoring short RTT paths [20]), 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 22 which shows the events occurring during a typical inactive period. 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. 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 back-off 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

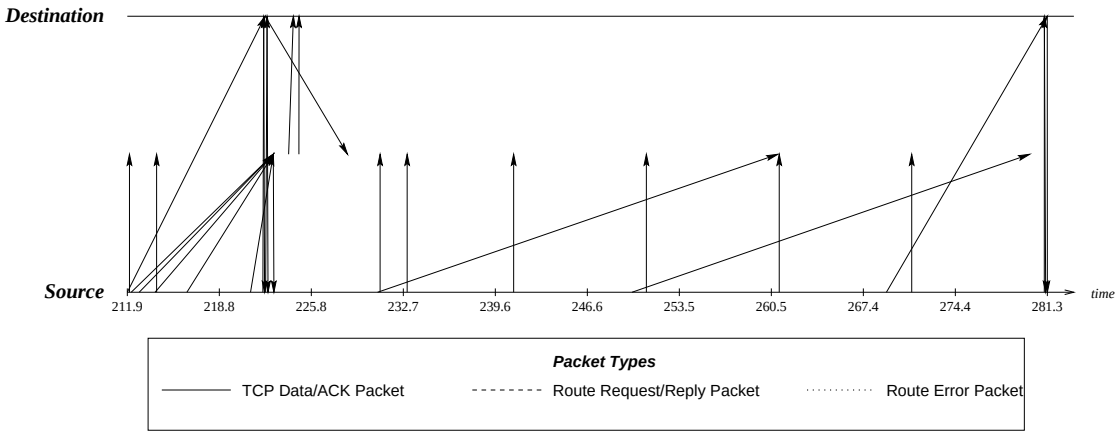


Fig. 22. Behavior in Inactive Period

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 [16]. 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.

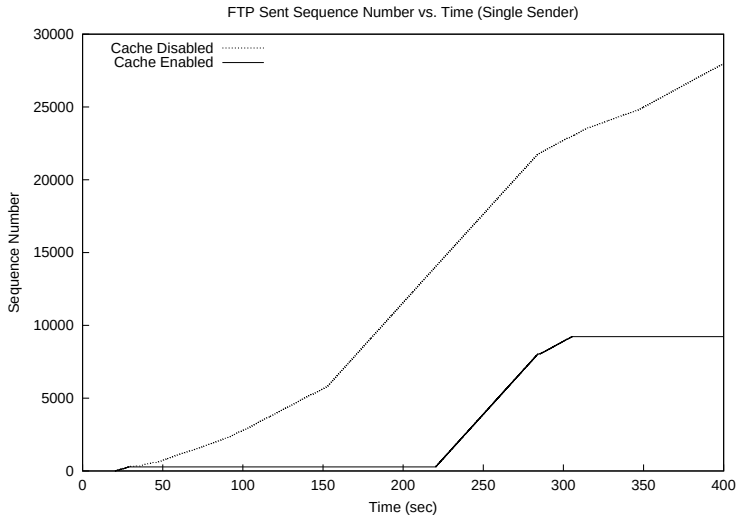


Fig. 23. Single Sender Inactive Period

Consistent with observations in [18], the inactivity periods were observed even in the case of a single sender/receiver pair for the baseline DSR (Figure 23). 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 timer to back-off exponentially, further exacerbating the problem. As can be observed in Figure 23, 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.

7 Concluding Remarks

In traditional mobile and wired-network routing algorithms, a change of path occurs in one of two cases: (i) a link along the path fails; or (ii) a shorter path is found. A link failure is costly since: (i) multiple retransmissions/timeouts are required to detect the failure; (ii) a new path has to be found and used (in on-demand routing). Since paths fail so infrequently 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. In this paper, we presented a class of algorithms that initiates proactive path switches when the quality of a path in use becomes suspect. We showed that this proactivity avoids using a path that is about to fail and eliminates the associated costs of detecting the failure and recovering from it, significantly improving the performance of the network.

We focused on signal power along each hop of the path as a measure of the quality of the path (a more robust definition of quality could include more factors such as the age of the path, number of hops, congestion). More specifically, using an estimate of the time needed to complete a path query, and relating that time to the motion patterns of the nodes, we derived a threshold on the signal power that will allow the nodes enough time to recover before the path gets disconnected. When a packet is received with a signal power below this threshold by a packet along a path, it generates a warning packet destined to the source of the path. The source then initiates a search for a higher quality path (a path where all the links are above the threshold) and immediately switches to it, avoiding a path break altogether.

As a case study, DSR and AODV were extended for preemptive maintenance (we call the modified algorithms PDSR and PAODV respectively). In DSR, route cache use was disabled to minimize false cache replies. Despite not using any caching, PDSR demonstrated significant improvements over non-proactive DSR: the number of broken paths was significantly reduced, and the latency and jitter of all packets were also improved. As expected, the overhead also increased since some path discoveries are being carried out proactively; however, much of the increase was due to disabling caching.

The effect of preemptive routing on TCP performance was also investigated for a number of traffic scenarios. Preemptive routing reduces the number of disconnections thereby avoiding unnecessary TCP backoff resulting in significant improvement in the performance of TCP. This improvement is obtained without requiring changes to TCP. The effect of stale cache paths in DSR causes significant reduction in performance of TCP. Elsewhere we studied active route cache optimization by validating cached paths infrequently, pruning low quality paths and doing scoped opportunistic searches for shorter paths. These optimizations resulted in dramatic improvement in TCP performance [27]. We are investigating combining the optimized cache with preemptive routing. We also observed unfairness issues with multiple TCP connections.

We are currently working on providing a more comprehensive evaluation of preemptive routing. More specifically, we are: (i) conducting overhead and hand-off delay studies vs. table-driven algorithms; (ii) studying other scenarios (larger networks and less dense node population). Finally,

in preemptive route maintenance, the first measure of a quality of a path was that the link integrity (being above the acceptable threshold). The length of the path was a secondary measure. We are currently studying the consolidation of other quality measures (such as path congestion and age) into a more comprehensive model for choosing the optimal path.

References

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

[2] UCB/LBNL/VINT Network Simulator, web-site http://www-mash.CS.Berkeley.EDU/ns.

[3] NS-2 with Wireless and Mobility Extensions, available via web-site http://www.monarch.cs.cmu.edu.

[4] Jrgen Bach Andersen, Theodore S. Rappaport, and Susumu Yoshida. Propagation measurements and models for wireless communications channels. *IEEE Communication Magazine*, 33(1):42–49, January 1995.

[5] 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.

[6] 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.

[7] A. Boukerche. Simulation based comparative study of ad hoc routing protocols. In *Proceedings of the 34th Annual Simulation Symposium*, pages 85–92, April 2001.

[8] A. Boukerche, S. Das, and A. Fabbri. Analysis of randomized congestion control with DSDV routing in ad hoc wireless networks. *Journal of Parallel and Distributed Computing*, pages 967–995, 2001.

[9] A. Boukerche and S. Rogers. GPS query optimization in mobile and wireless ad hoc networking. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, pages 198–203, July 2001.

[10] J. Broch, D. Maltz, D. Johnson, Y.-C. 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)*, October 1998.

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

[12] R. Castaneda and S.R. Das. Query localization techniques for on-demand routing protocols in ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'99)*, August 1999.

[13] C-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, April 1997.

[14] Federal Communications Commission (FCC). Enhanced 911 (e911) mandate, 2001. `http://www.fcc.gov/e911/`.

[15] J. Geier. *Wireless LANs: Implementing Interoperable Networks*. McMillan Technical Publishing, 1999.

[16] 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.

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

[18] 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)*, August 1999.

[19] Y.-C. Hu and D. Johnson. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networks (MobiCom'00)*, pages 231–242, August 2000.

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

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

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

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

[24] Y. Ko and N. H. Vaidya. Location-Aided Routing (LAR) Mobile Ad Hoc Networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom'98)*, October 1998.

[25] 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, special issue on mobile and wireless networks*, Aug. 1999.

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

[27] N. Panchal. Optimizing TCP and DSR caching behavior for ad hoc networks. Master's thesis, State University of New York at Binghamton, June 2001.

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

[29] 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.

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

[31] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM Computer Communications Review*, 24(4):234–244, October 1994. SIGCOMM '94 Symposium.

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

[33] T. Goff and J. Moronski and D. S. Phatak and V. Gupta. Freeze-TCP: a true end-to-end pro-active TCP enhancement mechanism for mobile environments. In *Proceedings of IEEE INFOCOM'2000, Israel, (to appear)*, Mar. 2000.

[34] 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.

[35] N. Vaidya. Tutorial on TCP for wireless and mobile hosts available at http://www.cs.tamu.edu/faculty/vaidya/seminars/tcp-tutorial-aug99.ppt.

Author Biographies

Tom Goff received the B.S. degree in electrical engineering and the M.S. degree in computer science from the State University of New York at Binghamton, in 1997 and 2000 respectively. He is currently pursuing a Ph.D. degree in computer science at the University of Maryland, Baltimore County. His research interests include wireless and mobile networking, distributed computing, and computer arithmetic algorithms and their implementations.

Nael Abu-Ghazaleh is an Assistant Professor at the Computer Science Department at the State University of New York, Binghamton. He received his B.Sc. Degree in Electrical Engineering from the University of Jordan, and the MS and PhD degrees in Computer Engineering from the University of Cincinnati in 1994 and 1997 respectively. His research interests are in mobile computing and networking, computer architecture, and parallel processing.

Dhananjay Phatak received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Bombay, in 1985; M.S. and Ph.D. from the Electrical and Computer Engineering Department, University of Massachusetts in 1990 and 1993 respectively. He is currently an Associate Professor in the Computer Science and Electrical Engineering Department at the University of Maryland Baltimore County. His research interests are in mobile networking, computer arithematics, and nueral networks. He holds a patent for a high speed CORDIC algorithm and has filed two other patents. He is a recipient of NSF's CAREER award and is an Associate Editor for IEEE Transactions on Computers.

Ridvan Kahvecioglu graduated from the Computer Science Department at Bogazici University in Turkey in 2001. From fall 1999 to fall 2000 he was an exchange student at the State Univer-

sity of New York Binghamton. He currently works as a Software Engineer for HP Turkey in the development and support of the HPUX Operating System.