Preemptive Routing in Ad Hoc Networks[*]

Tom Goff[†], Nael B. Abu-Ghazaleh[‡], Dhananjay S. Phatak[†] and Ridvan Kahvecioglu[‡]
[†]Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County (UMBC)
Baltimore, MD 21250
{tgoff1,phatak}@umbc.edu

[‡]Computer System Research Laboratory
Computer Science Dept.
Binghamton University
Binghamton, NY 13902–6000
nael@cs.binghamton.edu

ABSTRACT

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. We also show some experimental results obtained by running TCP on top of the proactive routing schemes proposed. Several improvements and extensions are also discussed. Pro-active route selection and maintenance is general and can be used with other routing algorithms and optimizations to them.

1. INTRODUCTION

Routing protocols for ad hoc networks can be classified into two categories: (1) Table-driven; and (2) Source initiated on-demand. Table-driven protocols attempt to maintain consistent, up-to-date routing information among all nodes in the network. Table-driven algorithms require periodic route-update messages to propagate throughout the network. This can cause substantial overhead (due to the "route information" traffic) affecting bandwidth utilization, throughput as well as power usage. The advantage 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 traffic overhead is less than Table-driven algorithms where many of the updates are for unused paths. Thus, there is a tradeoff between the overhead for maintaining paths and the time for establishing and mending paths.

In both types of algorithms, an alternative path is sought only *after* an active path fails. The cost of detecting a failure is high compared to typical packet latencies (several retries have to timeout 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 maintainance and selection to ad hoc routing protocols by finding alternative paths when a link is in danger of breaking (but before the disconnection occurs). 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 early and switch to a more stable path potentially avoiding the path break altogether. Moreover, when a path break is not avoided, the path discovery latency is reduced.

Preemptive routing maintenance algorithms attempt to combine the best of on-demand and table-driven: 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. While on-demand algorithms minimize the overhead by initiating route discovery only when needed, they do so *reactively*. Accordingly, 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. This increases both the average and variance (jitter) of packet latency. 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.

The effect of the suggested method is studied by extending Dynamic Source Routing (DSR) [12]; 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 [19]. In addition, the method does not affect other ad hoc routing optimizations such as location awareness [13] and query localization [6]. The signal strength is used as the preemptive 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.

[*]This work was supported by NSF Grants number ECS–9875705, CDA–800828, and EIA-9911099

The remainder of this paper is organized as follows. Section 2 overviews ad hoc routing protocols in more detail. Section 3 introduces the preemptive algorithm and discusses some possible optimizations to it. Section 4 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 maintainance. Section 6 presents an experimental evaluation of the proposed mechanism. Finally, Section 7 presents concluding remarks.

2. MOBILE AD HOC ROUTING ALGORITHMS

Ad hoc routing protocols can be broadly categorized as table-driven and source-initiated on-demand routing protocols. In table-driven protocols, each node maintains tables that store routing information. Changes in network topology trigger propagating updates throughout the network in order to maintain a consistent network view. The protocols in this area differ in the number of tables maintained, the information the tables contain as well as the details of how they are updated. For example, nodes in the Destination-Sequenced Distance Vector (DSDV) algorithm [20] is based on the Bellman-Ford algorithm [9] with every node maintaining route information to every other node in the network. As the network status changes, full updates (dumps) or incremental updates are exchanged among all nodes. The Wireless Routing Protocol (WRP) [16] localizes the updates to immediate neighbors. When a new node A moves into range of a node B and a hello message is received from it, A is added to B's routing table, and sent a full copy of the table. When a link fails, a node sends updates to its neighbors. The Cluster Gateway Switch Routing (CGSR) protocol [7] reduces the size of the tables and amount of information propagation by having each cluster of nodes elect a cluster head. Network-wide information is only exchanged among the cluster heads. While the amount of information propagation is reduced, this results in inefficient routes. The Fisheye State Routing protocol has been recently suggested. This is a link state table driven protocol that differs in that the update frequency is inversely related to the distance between any two nodes [18].

On-demand routing protocols are characterized by a path discovery mechanism which is initiated when a source needs to communicate with a destination that it does not know how to reach. Routing information is obtained via a "route discovery" process usually in the form of (an optimized) network-wide query flood. Once a route has been established, it is maintained (for example, by re-initiating route discovery when any link fails) until it is no longer needed. Generally, on-demand routing requires less overhead than table-driven routing [5, 11, 14, 15], but it incurs a path discovery delay whenever a new path is needed.

The differences between on-demand protocols are in the implementation of the path discovery mechanism and optimizations to it. Dynamic Source Routing (DSR) [12] uses source routing, with every packet carrying the full path information with it. A route discovery propagates through the network until it reaches the destination (which, assuming bidirectional links, then reverses the path and sends it back to the destination). Similarly, Ad hoc On-Demand Distance Vector Routing (AODV) [19] is an on-demand version of DSDV where the path discovery results in exchange of the portions of the routing tables necessary for establishing the route. Other on-demand algorithms include Temporally Ordered Routing Algorithm (TORA) [17] which discovers multiple paths from a source to destination and re-initiates discovery only when all of them have failed. Associativity-based routing (ABR) incorporates route quality by preferring hops that have been static for a long period [23]. Similarly, Signal Stability Routing (SSR) prefer routes with strong received signal power [8] – unlike our method they use signal power only after the path break is detected. To our knowledge, all existing algorithms change path only in reaction to a broken link (causing packet delay while the path break is detected, even if another path is available).

3. PREEMPTIVE ROUTE MAINTENANCE

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 networks follow this model despite the significantly higher frequency of path disconnections that occur in this environment; for each path break (in IEEE 802.11 standard) 3 MAC layer retransmits (a total of 4 time-outs including the original transmit) are tried before a link is considered broken.

A preemptive route maintenance algorithm initiates recovery action early by detecting that a link is likely to break soon and finds and uses an alternative path before the cost of a link failure is incured. This technique is similiar to soft-handoff techniques used in cellular phone networks as mobiles move across cells [21]. Thus, the algorithm maintains connectivity by preemptively switching to a "higher quality" path when the quality of a path in use becomes suspect. More specifically, the 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 on-demand protocols: we replace path failure, with the likelihood of failure as the trigger mechanism for route discovery. Although continuous update protocols could benefit from preemptive maintainance, their overhead is already too high and will only be increased from it.

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 ad hoc scenario, 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 similar temporary disturbances do not generate erroneous preemptive warnings. The next section examines these issues in more detail describing our approach to mitigating the effects of random signal fades (these are incorporated in our simulation experiments).

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.

4. 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. However, if the value is too high, the warning is generated too early with three negative side-effects: (i) unnecessary discoveries: the full life of the path currently in use is not exploited. Likewise, the moving nodes may change direction and the current path never breaks, rendering the preemptive action an unnecessary overhead; and (ii) 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 disconnects can 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.

4.1 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.

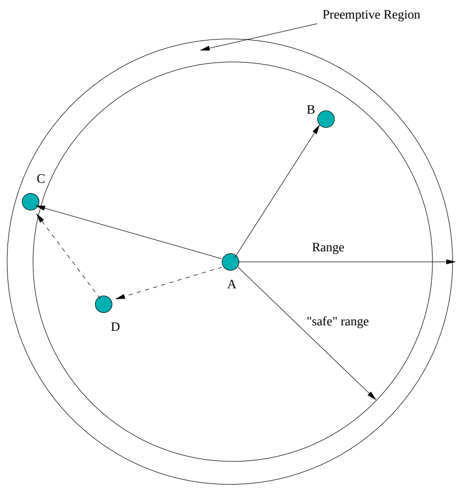


Figure 1: Preemptive Region

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

Given 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 is derived empirically, and in the protocol would be based on a running history estimate). 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. Of course, 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.

4.2 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. 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) [21]. In open environment, the main secondary component is the strong reflection of the transmitted signal from the ground. Equation 1 represents an approximate (and idealized) model for the channel with $n = 2$ near the source until a certain point where n becomes 4. Such an equation cannot account for channel fading in general (which can cause sharp and suddent fluctuations in signal power) because it is highly dependent on the specific surrounding terrain; we shall consider stable 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.

Figure 2(a) shows the received packet histogram from one of our simulations (non-preemptive). Figure 2(b) shows the histogram for the same scenario with a preemptive ratio of (1.1). The packets to the left of the preemptive threshold are communicated among nodes that are within the preemptive region of each other. In the preemptive histogram, the first packet sent when the nodes enter that region generates a warning. After an alternative path is found, this hop is no longer used; note that the number of packets to the left of the threshold is reduced in the preemptive case. In an ideal world, the number of packets in the preemptive region would be almost identical, as we preemptively switch just in time to avoid the path break.

4.3 Mitigation of Channel Fading and other transient interferences

In practice, the received signal power may experience sudden and substantial fluctuations due to channel fading and multipath effects. There is a concern that channel fading will trigger a preemptive route warning, causing unecessary route request floods. The unnecessary route request floods can have adverse effect on the performance as the network is saturated, and route switches to lower quality routes are initiated. Fortunately, there are established mechanisms to solve this problem in the cellular telephony field. 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. This is the approach we have used in our experiments. The details about the number of pings sent, the timeout periods, etc. are described 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. The source can also apply a dead reckoning calculation (using its own location and velocity information) to determine exactly when the path will be broken and when the optimal time to start corrective action is.

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.

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

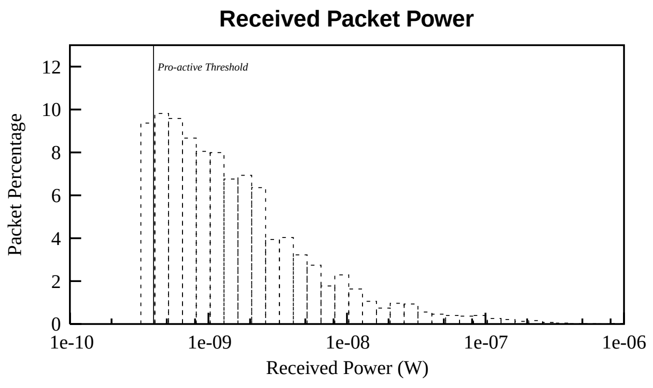
$$T_{\text{pkt}} \approx \frac{1}{\text{CBR}}. \tag{5}$$

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:

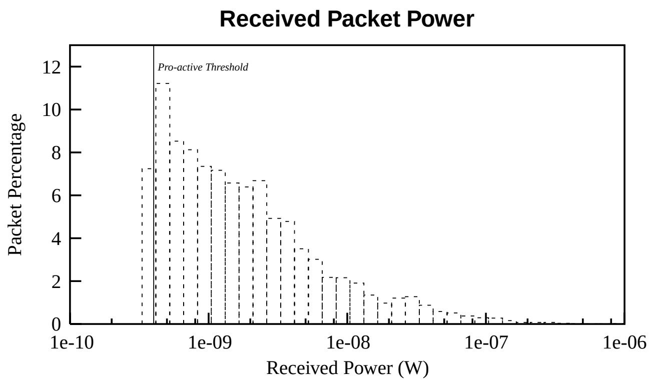
$$\begin{aligned} T_{\text{pkt}} &= \frac{1}{\text{CBR}} < \text{time to traverse the preemptive region} \\ &= \frac{w}{\text{average relative speed}} = T_{recover} \\ \text{or} \quad & \text{CBR} > \frac{1}{T_{recover}} \end{aligned} \tag{6}$$

5. PREEMPTIVE ROUTE MAINTENANCE CASE STUDIES

As was noted previously, preemptive maintainance can be added to all ad-hoc routing protocols (especially 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. These algorithms were chosen because they are widely used and have implementations available that can be modified; preemptive route maintenance strategy can be adopted with other on-demand routing algorithms. We focus on DSR for the sake of brevity and to allow detailed analysis. AODV is used to illustrate that ther results are not specific to DSR. In this section we describe the modifications made to DSR. The channel fading model



(a) On-Demand



(b) Preemptive (1.1)

Figure 2: Received Power

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 necesessary, 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. It was shown empirically that the paths discovered by DSR when cache replies are enabled are valid only 59% of the time [15].

Since the cost of a path discovery is significantly lower than the time to recover from a path disconnect in all but very large networks, we disabled the DSR caches (both local and cache replies). 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) [5, 12] 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, the scheme used in AODV [19]), and query localization [6, 13].

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 to 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 utilized a slightly modified form of the packet corruption model provided in the NS2. Instead of marking the packet as "bad" (i.e., as having an error) as is done in NS2, we decrease the power level to model the effects of 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 [21, 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 (PER) 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 stength 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 $3 \times 0.04 = 0.12$ 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 subthreshold packets within the window to trigger a route warning). These values were chosen arbitrarily and can benefit from emperical 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 exeriments 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.

For an unbiased comparison, scenarios similar to those previously studied [5, 15] 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.

The direct effect of preemptive routing can be seen by examining the number of broken paths (Figure 3). 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.

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 backoff initiated when a collision occurs [10]. 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 $\delta = 2$. 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 4 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

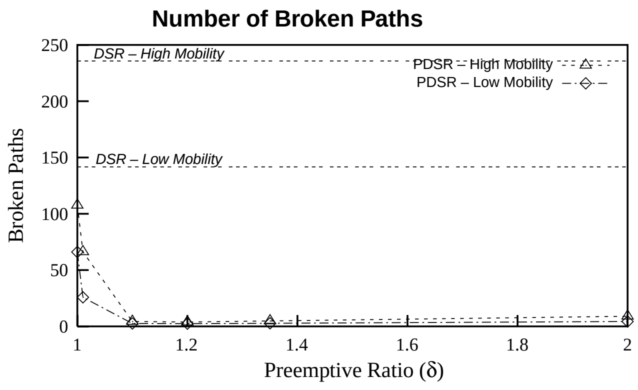


Figure 3: DSR Broken Paths

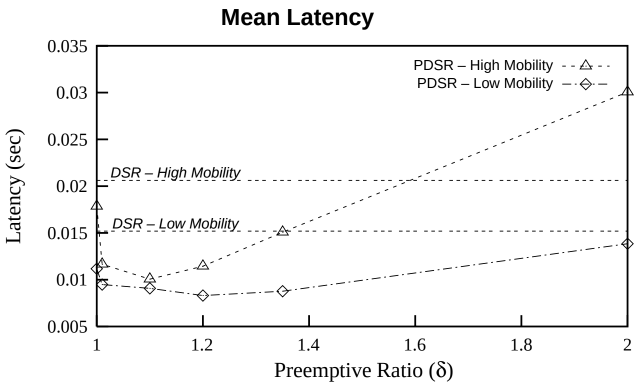


Figure 4: DSR Packet Latency

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 5 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 6 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 7 and Figure 8 show the number of broken paths and the packet latency for the same CBR 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 9 shows number of AODV packets introduced. As can be expected, the overhead increases with preemptive routing (due to searches prooactively 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.

Finally, Figure 10 demonstrates the effect of proactive routing on TCP traffic. The same mobility scenario was used with a number of TCP connections opened (instead of the CBR traffic). The traffic sources used were telnet-like (using the telnet agents in NS-2) – Tang and Baker showed that much of the traffic on a local area wireless network was of this type [22]. The improvement in packet latency was again significant. We conjecture that preventing broken paths has favorable effects on TCP's congestion avoidance behavior. Detailed analysis of the interaction between TCP and preemptive routing is a subject of future work.

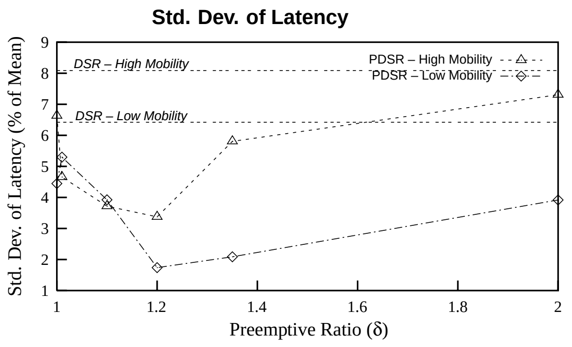


Figure 5: DSR Jitter

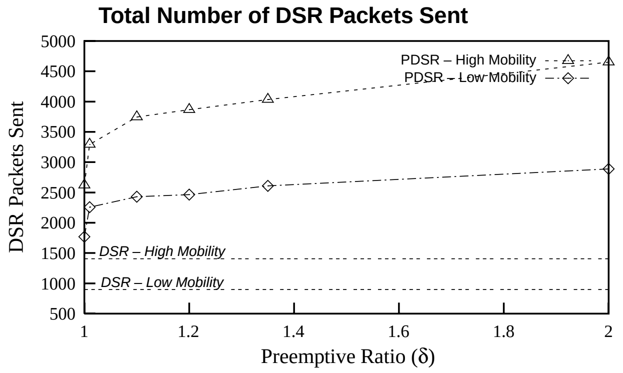


Figure 6: DSR Overhead

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 proactivity (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.

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

8. 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.

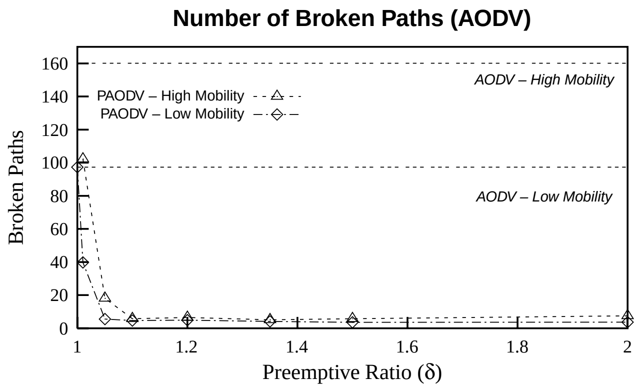


Figure 7: AODV Broken Paths

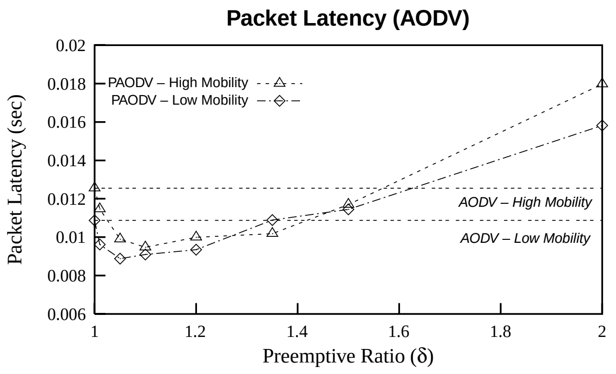


Figure 8: AODV Packet Latency

[4] J. B. Andersen, T. S. Rappaport, and S. Yoshida. Propagation measurements and models for wireless communications channels. *IEEE Communication Magazine*, 33(1):42–49, Jan. 1995.

[5] 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.

[6] R. Castaneda and S. 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)*, Aug. 1999.

[7] 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.

[8] R. Dube, C. Rais, K. Wang, and S. Tripathi. Signal Stability based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks. *IEEE Personal Communication*, pages 36–45, Feb. 1997.

[9] L. Ford Jr. and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

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

[11] P. Johansson, T. Larsson, N. Hedman, B. Milczarek, and M. Degermark. Routing protocols for mobile ad-hoc networks - a comparative performance analysis. In *Proceedings of the 1999 Mobile Computing Conference (MobiComm 1999)*, 1999.

[12] 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.

[13] 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)*, Oct. 1998.

[14] S.-J. Lee, M. Gerla, and C.-K. Toh. 'A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, Jul. 1999.

[15] 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.

[16] 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.

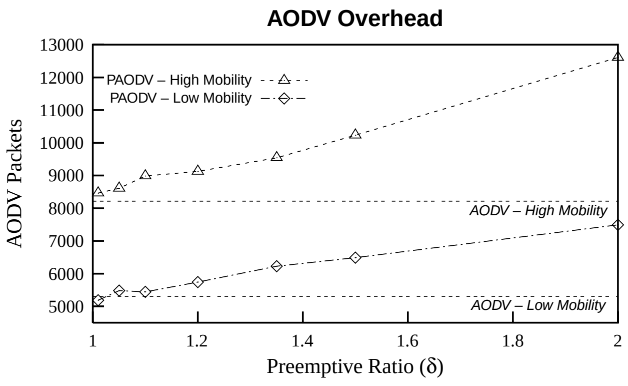


Figure 9: AODV Overhead

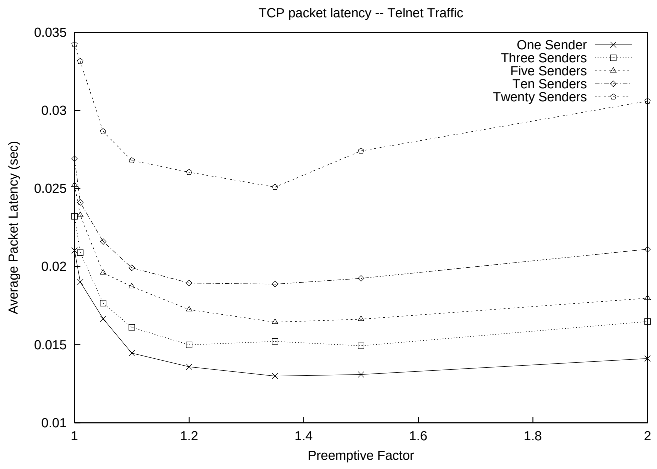


Figure 10: TCP Packet Latency; Telnet Traffic using DSR

[17] 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.

[18] 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.

[19] 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.

[20] 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, Oct. 1994. SIGCOMM '94 Symposium.

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

[22] 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.

[23] C. Toh. Associativity-Based Rotuing for Ad-Hoc Mobile Wireless Networks. *Wireless Personal Communications*, 4(2):1–36, Mar. 1997.