To appear in the *IEEE Journal on Selected Areas in Communications* special issue on mobile and wireless networks.

The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks

David A. Maltz   Josh Broch   Jorjeta Jetcheva   David B. Johnson

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A number of different routing protocols proposed for use in multi-hop wireless ad hoc networks are based in whole or in part on what can be described as *on-demand* behavior. By *on-demand* behavior, we mean approaches based only on reaction to the offered traffic being handled by the routing protocol. In this paper, we analyze the use of on-demand behavior in such protocols, focusing on its effect on the routing protocol's forwarding latency, overhead cost, and route caching correctness, drawing examples from detailed simulation of the *Dynamic Source Routing* (DSR) protocol. We study the protocol's behavior and the changes introduced by variations on some of the mechanisms that make up the protocol, examining which mechanisms have the greatest impact and exploring the trade-offs that exist between them.

1 Introduction

In a *wireless ad hoc network*, individual mobile nodes forward packets for other communicating mobile nodes that are out of wireless transmission range of each other. The network is dynamically self-organizing and self-configuring, with nodes establishing the necessary routing between each other without requirement for any existing infrastructure or administration. Since communicating mobile nodes may be some distance apart, multiple network "hops" through intermediate mobile nodes may be required to extend the communication range between them. As communicating nodes or those between them forwarding packets move about, the routing protocol in use in the network must adapt its routing decisions to enable continued communication between the nodes. The rate of topology change, and thus the rate of routing protocol reaction, may be quite dramatic in some ad hoc networks.

Many different routing protocols for use in such networks have been proposed, utilizing a wide variety of different routing algorithms and approaches. A number of these proposed protocols are based in whole or in part on what can be described as *on-demand* behavior [1, 3, 5, 6, 7, 9, 10, 12, 13, 14, 15], allowing them to dynamically adapt the level of routing protocol activity required to correctly handle the offered traffic. By *on-demand* behavior, we mean approaches based only on reaction to the presence of data packets. The use of strictly periodic or timer-based activities such as typical router advertisements, link or neighbor status sensing messages, or the deletion of routing table or cache entries after some expiration time, are not considered here. Although these other approaches may be quite useful as part of protocols for routing in wireless ad hoc networks, our aim here is to analyze the use of on-demand behavior as found in a number of proposed protocols.

In this paper, we focus on three questions that apply in general to any wireless ad hoc network routing protocol using on-demand behavior:

- *What effect does on-demand routing have on packet latency?* An on-demand routing protocol attempts to discover a route to a destination only when it is presented with a packet for forwarding to that destination. This discovery must be completed before the packet can be sent, which adds to the latency of delivering the packet.

Indeed, some mechanisms to reduce the overhead cost of discovering a new route, may result in an increase in latency for some Route Discovery attempts.

- *What is the overhead cost of on-demand routing behavior?* Without additional information, a protocol using on-demand routing must search the entire network for a node to which it must send packets, but does not know how to reach. Optimizations to the protocol may reduce the cost of initiating communication, but discovering a new route is likely to remain a costly operation.

- *When caching the results of on-demand routing decisions, what is the level and effect of caching and cache correctness on the routing protocol?* Any on-demand routing protocol must utilize some type of routing cache in order to avoid the need to re-discover each routing decision for each individual packet. However, the cache itself may contain out-of-date information indicating that links exist between nodes that are no longer within wireless transmission range of each other. This stale data represents a liability that may degrade performance rather than improve it.

To address these questions, we utilize examples drawn from detailed simulation of the *Dynamic Source Routing* (DSR) protocol [1, 9, 10], studying its behavior and the changes introduced by variations on some of the mechanisms that comprise it. The DSR protocol provides a good source of examples for this study, since it is based *entirely* on on-demand behavior, although we believe that our results here generalize to other protocols using related approaches.

Previous simulation efforts have evaluated the performance of ad hoc routing protocols with respect to varying environmental conditions, such as the number of nodes, the nodes' movement pattern, or the traffic load. Evaluations of this form tend to examine only a few summary metrics of interest to network users (e.g., packet delivery ratio or routing overhead). In contrast, our goal is to dissect the protocol into its component mechanisms. We seek to determine both the effectiveness of the individual mechanisms, and the manner in which they interact to contribute to the overall performance measured by summary metrics.

In Section 2 of this paper, we give an overview of the operation of the DSR protocol, and in Section 3, we discuss our simulation framework and methodology for this study. In Section 4, we give a summary of the baseline performance of DSR, including all optimizations to the protocol. The next three sections then present our analysis of each of the general questions posed above: Section 5 covers latency issues, Section 6 covers overhead cost, and Section 7 covers cache consistency. Finally, in Section 8, we present conclusions and discuss future work.

2 Overview of the DSR Protocol

The DSR protocol [9, 10, 1] is composed of two mechanisms: *Route Discovery* and *Route Maintenance*, each of which operates using entirely *on-demand* behavior. When a node in the ad hoc network attempts to send a packet to some destination, if it does not already know a route to that destination, it uses Route Discovery to dynamically discover one. The route is cached and used as needed for sending subsequent packets, each of which utilizes the Route Maintenance mechanism to detect if the route has broken, for example due to two nodes along the route moving out of wireless transmission range of each other. Route Discovery is only invoked when needed, and Route Maintenance operates only when actively using the route to send individual packets.

The routes that DSR discovers and uses are *source routes.* That is, the sender learns the complete, ordered sequence of network hops necessary to reach the destination, and each packet to be routed carries this list of hops in its header. The key advantage of a source routing design is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets that they forward, since the packets themselves already contain all the routing decisions. This fact, coupled with the on-demand nature of the protocol, eliminates the need for the periodic route advertisement and neighbor detection packets present in other protocols.

Route Discovery works by flooding a request through the network in a controlled manner, seeking a route to some target destination. In its simplest form, a source node **S** attempting to discover a route to a destination node **D** broadcasts a ROUTE REQUEST packet that is re-broadcast by intermediate nodes until it reaches **D**, which then answers by returning a ROUTE REPLY packet to **S**. Many optimizations to this basic mechanism are used to attempt to limit the frequency and spread of Route Discovery attempts.

When sending or forwarding a packet to some destination **D**, Route Maintenance is used to detect if the network topology has changed such that the route used by this packet has broken. Each node along the route, when transmitting the packet to the next hop, is responsible for detecting if its link to the next hop has broken. In many wireless MAC protocols, such as IEEE 802.11 [8], the MAC protocol retransmits each packet until a link-layer acknowledgment is received, or until a maximum number of transmission attempts have been made. Alternatively, DSR may make use of a *passive acknowledgment* [11] or may request an explicit network-layer acknowledgment. When the retransmission and acknowledgment mechanism detects that the link is broken, the detecting node returns a ROUTE ERROR packet to the original sender **S** of the packet. The sender **S** can then attempt to use any other route to **D** that is already in its route cache, or can invoke Route Discovery again to find a new route.

2.1 Optimizations to Route Discovery

Nonpropagating ROUTE REQUESTS. When performing Route Discovery, nodes first send a ROUTE REQUEST with the maximum propagation limit (hop limit) set to zero, prohibiting their neighbors from rebroadcasting it. At the cost of a single broadcast packet, this mechanism allows a node to query the route caches of all its neighbors for a route and optimizes the case in which the destination node is adjacent to the source. If the nonpropagating ROUTE REQUEST fails to elicit a reply within a 30 ms time limit, a propagating ROUTE REQUEST with a hop limit set to the maximum value is sent. The 30 ms timeout was chosen based on the distribution of REPLY latencies shown in Figure 4.

Replying from cache. If a node receives a ROUTE REQUEST for a destination **D** to which it has a route, the node may generate a ROUTE REPLY based on its cached information instead of rebroadcasting the ROUTE REQUEST. This optimization is intended to both reduce the latency of ROUTE REPLIES and prevent ROUTE REQUESTS from flooding through the entire network.

Gratuitous ROUTE REPLIES. When a node overhears a packet not addressed to itself, the node checks if the packet's header contains its address in the unprocessed portion of the source route. If so, the node knows that packet could bypass the unprocessed hops preceding it in the source route. The node then sends a gratuitous ROUTE REPLY message to the packet's source, giving it the shorter route without these hops. Upon receiving the ROUTE REPLY, the originator will insert the shorter route into its route cache. The route cache can store more than one route to a destination, so the shorter route will not necessarily overwrite the longer route already in the cache. If the shorter route is found not to work, the originator can immediately revert to the longer route.

2.2 Optimizations to Route Maintenance

Salvaging. When an intermediate node forwarding a packet discovers that the next hop in the source route for the packet is unreachable, it examines its route cache for another route to the same destination. If a route exists, the node replaces the broken source route on the packet's header with the route from its cache and retransmits the packet. If a route does not exist in its cache, the node drops the packet — it does not send a ROUTE REQUEST. In either case, the node attempting to perform salvaging returns a ROUTE ERROR to the source of the data packet.

Gratuitous ROUTE ERRORS. When a source **S** receives a ROUTE ERROR for a packet that it originated, **S** propagates this ROUTE ERROR to its neighbors by piggybacking it on its next ROUTE REQUEST. In this way, stale information in the caches of nodes around **S** will not generate ROUTE REPLIES that contain the same invalid link for which **S** received a ROUTE ERROR.

2.3 Optimizations to Caching Strategies

Snooping. When a node forwards a data packet, it "snoops" on the unprocessed portion of the source route and adds to its cache the route from itself to the final destination listed in the source route.

Tapping. Nodes operate their network interfaces in *promiscuous* mode, disabling the interface's address filtering and causing the network protocol to receive all packets that the interface overhears. These packets are scanned for useful source routes or ROUTE ERROR messages and then discarded. This optimization allows a node to prime its route cache with potentially useful information, while causing no additional use of the limited network bandwidth.

3 Methodology

We analyzed the effect of the mechanisms that comprise DSR by simulating different variations of the DSR protocol on an identical network with an identical workload. Simulation enabled us to study a large number of points in the DSR design space and to directly compare the results of the simulations, since we were able to hold constant factors such as the communication and movement pattern between runs of the simulator. We conducted the experiments using the *ns-2* network simulator [4] extended with our support for realistic modeling of mobility and wireless communication [2]. The simulator allows the specification of arbitrary movement patterns for the nodes, and correctly models the effects of contention for the media and the distance between nodes in determining whether a transmitted packet will be successfully received. Each simulation used 50 nodes and simulated 900 seconds of real time.

Each node in the simulation communicated via a radio with the characteristics of Lucent Technologies WaveLAN [16]. WaveLAN is a shared-media radio with a raw capacity of 2 Mb/s, and a 250m nominal range. The exact range of a transmission varies with the number of simultaneous transmissions, since our simulator models the attenuation of transmitted signals with distance and capture effects. At the link layer, we simulate the complete Distributed Coordination Function (DCF) medium access control (MAC) protocol of the IEEE 802.11 Wireless LAN standard [8].

3.1 Node Movement and Communication Pattern

In preparing this paper we analyzed the effect of two different movement models: constant node motion and no node motion. All tables and graphs in this paper are from the constant motion case, since without motion DSR does a single Route Discovery for each destination, and then aside from congestion-caused packet drops leading to Route Errors, there is no more protocol activity for the duration of the run. For both constant motion and no motion models, the nodes start at a uniformly distributed location. In the constant motion case, all nodes then move according to the *random waypoint* algorithm [10, 2] wherein each node picks a new random location in the simulated area and proceeds there at a speed chosen uniformly from 0 to 20 m/s. When the node reaches this waypoint, it picks another location to move to and repeats the cycle.

We chose to model node communication using a uniform node-to-node communication pattern with constant bit rate (CBR) traffic sources sending data in 512-byte packets at a rate of 4 packets per second. A total of 20 CBR connections were modeled in each simulation run, with each node being the source of 0, 1, or 2 connections. In all runs, the 20 connections were spread in this way over a total of 14 different originating nodes (which we label nodes 1 through 14). All CBR connections were started at times uniformly distributed during the first 180 seconds of simulated time and then remained active through the entire simulation. We chose the parameters of the communication pattern to stress the ability of the routing protocol to discover and maintain routes during the experiments, but to avoid causing extreme congestion.

In order to average out the effects of particular motion or communication patterns, we generated 10 different scenarios — 5 with constant node motion, and 5 with no node motion. We ran all experiments over the same set of scenarios.

3.2 Simulated Sites

In an attempt to generate results that would be representative of some potential, real-world scenarios that DSR might encounter, we ran our simulations over two different types of simulated sites:

- A flat site with length much greater than its width. In this paper, we used dimensions of 1500m $\times$ 300m, and we call this the *rectangular site*.
- A flat site with area approximately equal to that of the rectangular site, but in which the length and width are equal. In this paper, we used dimensions of 670m $\times$ 670m, and we call this the *square site*.

Altering the shape of the simulated site in this way creates networks with qualitatively different topologies and throughput bottlenecks, thereby exercising DSR in different ways.

The rectangular site causes a roughly linear arrangement of the nodes. The results seen here could be applied to situations where the physical paths between nodes are very constrained, such as when the mobile nodes are vehicles driving along a road. The lengths of the routes taken by packets in this site are typically longer than the corresponding route lengths in the square site. Since the site is also narrow, a packet being transmitted down the site is typically overheard by all the nodes spatially located between the sender and destination. This makes networks in the rectangular site more prone to congestion bottlenecks since there is little spatial diversity in the narrow dimension.

The square site models situations in which nodes can move freely around each other, and where there is a reasonable amount of path and spatial diversity available for the routing protocol to discover and use. Since its total area is approximately the same as that of the rectangular site, the average node density between the two sites is constant, but the average route length used in the square site is less than in the rectangular site.

4 Baseline Evaluation of DSR

In order to characterize the performance of DSR in the scenarios used for this paper, we evaluated the performance of DSR based on the following two metrics:

- *Packet delivery ratio*: The ratio between the number of packets originated by the "application layer" CBR sources and the number of packets received by the CBR sinks at the final destinations.
- *Routing overhead*: The total number of routing packets transmitted during the simulation. For packets sent over multiple hops, *each* transmission of the packet (each hop) counts as one transmission.

Packet delivery ratio is important as it describes the loss rate that will be seen by the transport protocols, which in turn determines the maximum throughput that the network can support. This metric characterizes both the completeness and correctness of the routing protocol.

The routing overhead metric is important because it measures the scalability of a protocol, the degree to which it will function in congested or low-bandwidth environments, and its efficiency in terms of consuming node battery power. A protocol that sends large numbers of routing packets can also increase the probability of packet collisions and may delay data packets in network interface transmission queues.

Figure 1 shows the packet delivery ratio (the top curve on the graph) and routing packet overhead (the bottom curve) for the All-Opt DSR version of the protocol, which utilizes all of the optimizations described in Section 2. Both metrics are plotted as a function of the node mobility rate, measured in seconds of *pause time* [2], with 0 seconds meaning constant node motion and 900 seconds meaning no node motion. All-Opt DSR is able to deliver 90% or more of the data packets originated by the CBR traffic sources, and the routing overhead scales well with pause time, remaining reasonable even at constant node mobility. The offered traffic load and movement pattern in these scenarios was the same as that described in Section 3.

The figure confirms our prior experience comparing four ad hoc network routing protocols [2], which showed that DSR has great potential as an efficient protocol in a multi-hop ad hoc network environment. However, DSR is only one example of the larger class of on-demand routing protocols. The remainder of this paper is a detailed analysis of several of DSR's mechanisms aimed at identifying their strengths and weaknesses, in order to guide the design of future on-demand protocols.

5 Effects on Latency

The use of on-demand behavior in routing protocols for multi-hop wireless ad hoc networks can result in increased packet latency due to the need to delay sending a packet if the packet requires a route to a previously unknown destination. When some node **A** wants to send a packet to a node **B**, and it does not currently have a route to **B**, it must first perform a Route Discovery for node **B** before sending the packet. Because **A** must buffer its packet until it has a route to **B**, the entire time that Route Discovery takes to obtain a route to **B** adds directly to the time it will take for the packet to be delivered. This makes the latency of Route Discovery critical in any environment in which packets must be delivered in a timely fashion.

In the DSR protocol, each Route Discovery consists of two phases:

- **Nonpropagating ROUTE REQUEST**. Node **A** transmits a ROUTE REQUEST with a maximum propagation limit of one hop. Any node receiving the ROUTE REQUEST that does not have a route to **B** simply discards the request.
- **Propagating ROUTE REQUEST**. If after 30 ms the nonpropagating ROUTE REQUEST has failed to return a route to **B**, node **A** will then transmit a propagating ROUTE REQUEST. Each node other than **B** that receives this request will either return a ROUTE REPLY based on information in its route cache, or will rebroadcast the ROUTE REQUEST, propagating it further through the network. Should **B** itself receive the request, it will return a ROUTE REPLY consisting of the source route collected in the ROUTE REQUEST.

We consider two aspects of latency: the amount of time it takes a node to acquire a route to a destination, and the amount of time it takes a sender to "recover" (find a new source route) when a route that it is using breaks. In order to calculate the latency of a Route Discovery, we measure the time from when a node sends a ROUTE REQUEST until it receives the *first* ROUTE REPLY that answers the request. We use the first reply to calculate latency since as soon as it arrives, the node can begin transmitting data packets — it need not wait for multiple replies to be returned before sending.

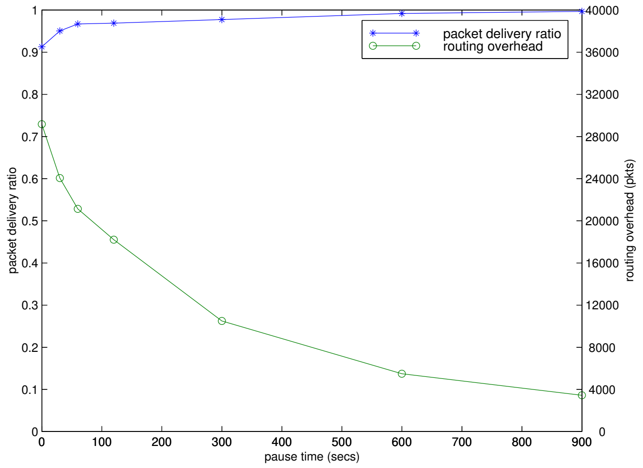

Figure 1 Baseline DSR packet delivery ratio and routing packet overhead (All-Opt DSR, rectangular site)

The individual per-hop forwarding latency of all packets is dictated by the length of the interface queue and by the media access time, since the packets at each node must wait their turn for transmission. The latency of a single ROUTE REQUEST propagating outwards or a ROUTE REPLY propagating inwards is dominated by effects due to path length: the greater the number of hops over which the ROUTE REQUEST or ROUTE REPLY packet must travel, the longer the time the packet takes to reach its destination. However, the greatest influence on the latency of a Route Discovery is the distribution of information in nodes' caches, since it is the path length to the closest node that can generate a ROUTE REPLY that determines the latency of Route Discovery.

The latency of the slowest Route Discoveries seen during an experiment are dominated by the effects of congestion. The maximum latencies reported in this section are frequently in the tens of seconds — we have verified in each case that the REQUEST or REPLY was held up in a series of long interface queues as the packet propagated, each queue draining very slowly due to media contention around the node.

5.1 Latency Related to Path Length

In Figure 2, we show the distribution of times required for Route Discovery, broken out by the number of hops in the discovered route. Figure 3 is based on the same data, showing the latency of Route Discovery as a scatter plot versus the number of hops over which the ROUTE REQUEST and matching ROUTE REPLY had to travel. The scatter plot makes visible the distribution of individual samples, and the abscissas have been uniformly jittered to make the density of the points visible. The least squares fit line shown on the graph has a slope of 14.5 ms/hop.

The spread of latencies recorded reflect the realistic nature of the simulator used. The minimum time required to forward a small packet, such as a ROUTE REQUEST or ROUTE REPLY, over a single hop is approximately 600 μs. This assumes the packet immediately acquires the media when offered to the network. The minimum per-hop forwarding time for a 512 byte data packet is approximately 3 ms. However, since the effective carrier-sense range on the simulated radios is 550 m, on average each packet sent must defer to other packets being transmitted elsewhere in the network.

5.2 Latency of Route Discovery

The ROUTE REPLY packets received by a node in response to sending a ROUTE REQUEST may be categorized as follows:

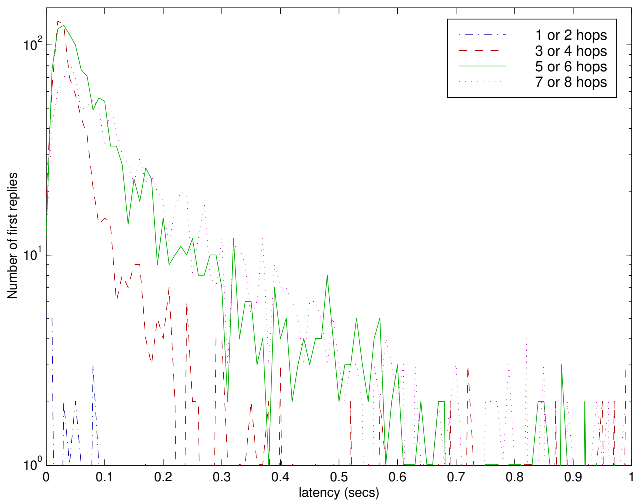

Figure 2 Distribution of latencies of ROUTE REPLY packets by number of hops (No cache replies, rectangular site, constant motion)

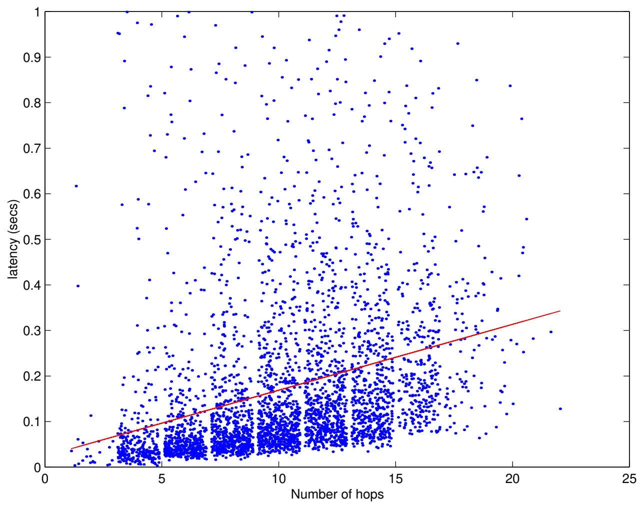

Figure 3 Scatter plot of forwards vs. latency. Forwards includes the number of hops from the originator of the Route Discovery to the target and from the target back to the originator. (No cache replies, rectangular site, constant motion)

- **Cache Replies**. Replies constructed from cached routing information by a node other than the target of a Route Discovery.
- **Target replies**. Replies originated by a node in response to receipt of a ROUTE REQUEST targeting it. We say that target replies are based on *fresh* routing information because the route from requester to target has just been traversed by the corresponding ROUTE REQUEST.
- **Neighbor replies**. A ROUTE REPLY returned in response to a ROUTE REQUEST with a maximum propagation limit (TTL) of 1. Neighbor replies must originate from a direct neighbor of the requester (since the request cannot propagate), but may be either cache replies or target replies depending on whether or not the respondent is the target of the Route Discovery.

Table I summarizes the latency required for the initiator of a Route Discovery to receive the first ROUTE REPLY in response to its ROUTE REQUEST. This data is broken down into the mean, minimum, and maximum latencies for each of the three different types of ROUTE REPLIES listed above. In this section, we analyze the latency for each type of ROUTE REPLY and examine the effect of changing the shape of the site on the latency.

5.2.1 The Latency of Neighbor Replies

Figure 4 shows a detailed view of the latency of the first ROUTE REPLY packet received for nonpropagating ROUTE REQUESTS. The graph shows only those replies that were returned within 100 ms. After 30 ms, a nonpropagating request is considered unsuccessful and a propagating ROUTE REQUEST is transmitted and so only replies returned

Table I Latency of first ROUTE REPLY by type (All-Opt DSR, rectangular site, constant motion)

		Latency			
	# Replies	Mean	Min	99 percentile	Max
Neighbor Replies	3136	7.1ms	1.3ms	457ms	17.8s
Cache Replies	1524	45.9ms	1.3ms	752ms	2.4s
Target Replies	12	87.6ms	23.6ms	458ms	458.3ms

within the 30 ms timeout are useful in terms of preventing a propagating Route Discovery. Since neighbor replies with latency greater than 30 ms may be interpreted as the first reply to the propagating ROUTE REQUEST, we have included replies with a latency of up to 100 ms in our analysis. The mean of this distribution (Table I) is 7.1 ms, and was computed by discarding as outliers all replies that arrived after 100 ms.

5.2.2 The Latency of Cache Replies

When a nonpropagating request fails to obtain a route, the node performing Route Discovery sends a propagating ROUTE REQUEST. Each node that hears this request will either transmit a ROUTE REPLY from its cache or forward the request as described above. Figure 5 shows a detailed view of the latency of ROUTE REPLY packets sent from cached route information received in response to a propagating ROUTE REQUEST. Again discarding outliers, the mean of this distribution is 45.9 ms (Table I), nearly seven times larger than the mean latency for neighbor replies (7.1 ms). This difference between mean neighbor reply latency and the latencies of cache replies indicates that successful nonpropagating requests are very effective in reducing latency. However, the difference does not show exactly how effective they are since the cache replies in Table I are elicited by propagating requests which are only sent after a nonpropagating request has already failed.

In order to factor out the effect of nonpropagating ROUTE REQUESTS on the latency of cache replies, we removed the mechanism that sends nonpropagating requests and ran this modified version of DSR on the same set of scenarios. The results of this experiment are reported in Table II and Figure 6. When only propagating ROUTE REQUESTS are sent, the mean latency for cache replies dropped to 21 ms, less than half of the latency measured when nonpropagating requests were enabled. From this, we conclude that nonpropagating ROUTE REQUESTS offer an average savings of about 14 ms in latency (a factor of three decrease).

5.2.3 The Latency of Target Replies

The third row of Table I in Section 5.2 summarizes the latency of replies from the target of a Route Discovery. The mean of 87.6 ms is an increase of approximately 40 ms over the time for cache replies. This increased latency is in general due to the increased number of hops over which the ROUTE REQUEST and matching ROUTE REPLY must propagate to reach the target, relative to what is needed to reach the first node with a route cache entry that can return a reply from its cache.

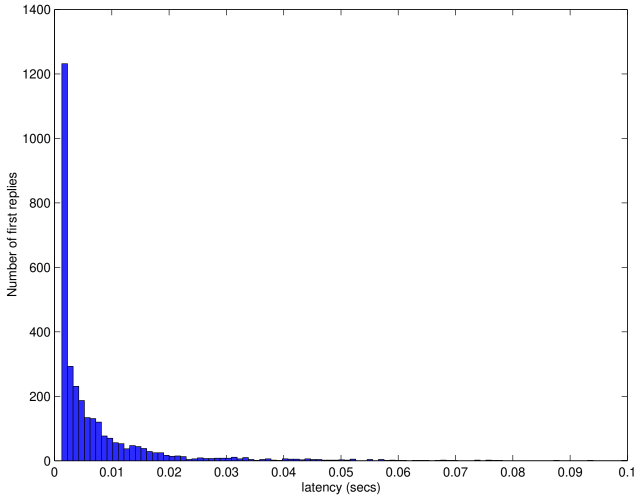

Figure 4 Latency of neighbor replies (All-Opt DSR, rectangular site, constant motion)

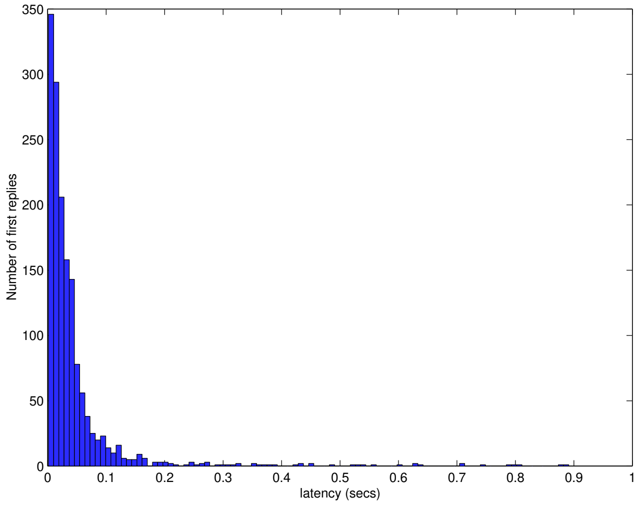

Figure 5 Latency of Cached ROUTE REPLY packets (All-Opt DSR, rectangular site, constant motion)

Table II Latency of first ROUTE REPLY (No nonpropagating ROUTE REQUESTS, rectangular site, constant motion)

	Mean Latency	Min Latency	Max Latency
Cache Replies	21.0ms	1.3ms	8.0s
Target Replies	33.3ms	3.8ms	63.0ms

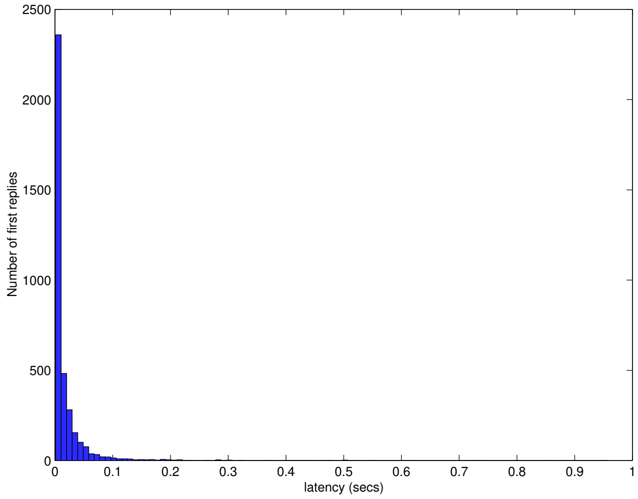

Figure 6 Latency of cached ROUTE REPLY packets (No nonpropagating ROUTE REQUESTS, rectangular site, constant motion)

Table III Latency of first ROUTE REPLY (No nonpropagating ROUTE REQUESTS and no cache replies, rectangular site, constant motion)

	Mean Latency	Min Latency	Max Latency
Target Replies	403.1ms	3.5ms	42.1s

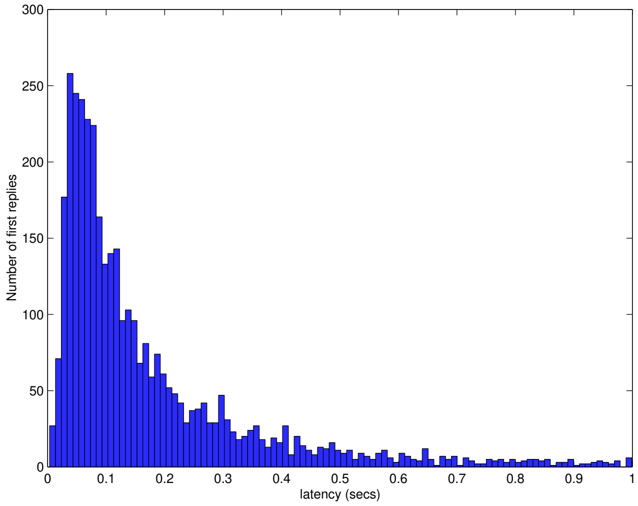

Figure 7 Latency of ROUTE REPLY packets (No nonpropagating ROUTE REQUESTS and no cache replies, rectangular site, constant motion)

In almost all cases, the first ROUTE REPLY received in response to a ROUTE REQUEST is not a target reply, but rather is a cache reply. In our simulations, the first ROUTE REPLY came from the target node only a total of 12 times over all All-Opt DSR scenarios studied. This small number of target replies indicates that DSR's caching mechanism and its ability to send replies from the cache are very effective in reducing the latency of Route Discovery, but this limits our ability to directly evaluate the latency of target replies, and the latency for target replies shown in Table I represents only these 12 samples.

As another way of measuring the latency of target replies, we disabled both nonpropagating ROUTE REQUESTS and replying from cache in DSR and ran this modified version of the protocol over the same scenarios as above. Table III summarizes the target reply latencies from this experiment, and Figure 7 shows a detailed view of this latency. The mean latency for target ROUTE REPLIES in this experiment (the only type enabled) was 403.1 ms. This indicates that the optimizations added to DSR (replies from cache and non-propagating route requests) significantly decrease the latency of Route Discovery. The reason latency is higher with the optimizations turned off is the decreased containment (Section 6) of ROUTE REQUESTS and the longer paths that ROUTE REPLIES and REQUESTS must take.

5.2.4 The Effect of a Different Topology

In the square site, we expect two factors to work together to decrease the latency of Route Discovery as compared to the rectangular site. The mean path length is less, so forwarding latency related to path length should be less, and the average degree of each node (the number of direct, 1-hop neighbors of each node) is higher, which should result in cache information being better distributed throughout the network, causing the first ROUTE REPLY received to come from nodes closer to the initiator of the Route Discovery.

The average degree of a node in the square site is 16.5 as compared to 11.5 in the rectangular site. As shown in Table IV, the mean first ROUTE REPLY latency for all three types of REPLIES decreases relative to the latencies in the rectangular site (Table I), supporting the hypothesis that there are more nodes "close" to the originator of a Route Discovery and that routing information is more spread out through the network.

5.3 Latency Incurred by Broken Routes

As described in Section 2, a node originating packets will detect that a route it is using has broken only after it tries to send a packet along the broken route, and receives a ROUTE ERROR in response. The ROUTE ERROR is sent by the

Table IV Latency of first ROUTE REPLY by type (All-Opt DSR, square site, constant motion)

	Mean Latency	Min Latency	Max Latency
Neighbor Replies	4.7ms	1.3ms	117.1ms
Cache Replies	29.1ms	1.3ms	1.3s
Target Replies	36.6ms	29.4ms	54.4ms

node that, upon attempting to forward the packet over a link in the source route, decides that the link is broken. The forwarding node declares the link broken after making several attempts to transmit the packet to the next hop, and failing to receive a passive or explicit acknowledgement of success.

An important aspect of latency in the protocol is the time that it takes a source node **S** to detect that such a link in its route to a destination **D** has broken. This latency is a function of the time required to send the original packet (the packet triggering the ROUTE ERROR) along the route as far as the node that detects the broken link, and the time required for that node to send the ROUTE ERROR back to the original sender.

For All-OptDSR in the rectangular site, Table V shows the time taken for a ROUTE ERROR to propagate from the node that detected a link breakage to the originator of the packet that contained the broken source route. The median latency per hop is 8.6 ms, while the mean is 26.6 ms. The difference between the two is due to the heavy tail caused by congestion. The average link breakage in this environment occurred 1.8 hops from the packet's originator.

6 Effects on Overhead Cost

Although on-demand routing protocols can reduce routing overhead by not disseminating routing information throughout the network on a periodic basis, the actual cost of performing on-demand Route Discovery can be significant. When a node **A** transmits a ROUTE REQUEST, the request flood-fills through the network, potentially disturbing each node in the network and consuming valuable bandwidth and battery power. Each node that receives the request must either transmit a ROUTE REPLY based on information in its route cache or forward the request further.

We define two metrics to evaluate the cost of on-demand Route Discovery, which we refer to as *containment* and *discovery cost*:

- Containment is defined as the percentage of nodes that do not receive a particular ROUTE REQUEST. For a nonpropagating ROUTE REQUEST (Section 2), containment is equivalent to measuring the percentage of nodes in the network which are not neighbors (within transmission range) of the node originating the request. For a propagating ROUTE REQUEST, containment measures how far out the request propagates before running into either the edge of the network or a band of nodes with cached information about the target that is wide enough to stop further propagation. Values of containment approaching 1 indicate that a ROUTE REQUEST was well contained and interrupted very few nodes, whereas containment values approaching 0 indicate that most of the nodes in the network had to process the request.

Table V Latency of ROUTE ERRORs (All-OptDSR, rectangular site, constant motion)

	number of hops ROUTE ERROR traveled							
	1	2	3	4	5	6	7	8
median (ms)	8.75	17.37	24.64	29.21	37.75	80.82	58.13	336.1
# ERRORS	8602	3985	1978	929	367	123	18	4

- The cost of a single Route Discovery is defined as $1 + \mathrm{FwReq} + \mathrm{OgRep} + \mathrm{FwRep}$, where 1 represents the transmission of the original request, *FwReq* is the number of ROUTE REQUEST forwards, *OgRep* is the number of ROUTE REPLY originations, and *FwRep* is the number of ROUTE REPLY forwards. For each Route Discovery, this metric measures the number of routing packets (requests and replies) that were transmitted to complete the discovery. The average discovery cost is calculated as $\frac{\mathrm{OgReq}+\sum\mathrm{FwReq}+\sum\mathrm{OgRep}+\sum\mathrm{FwRep}}{\mathrm{OgReq}}$, where *OgReq* is the number of ROUTE REQUEST originations, and *FwReq*, *OgRep*, and *FwRep* are summed over all Route Discoveries.

Our intuitive model of how Route Discovery works predicts that both the containment and discovery cost metrics of Route Discovery should be sensitive to the average *degree* of the nodes in the network. The degree of a node is the number of direct neighbors the node has, and measures how tightly interconnected the network is. As the degree of interconnectivity goes up, it is harder to contain a ROUTE REQUEST to one part of the network. In addition, the "branching factor" of a propagating ROUTE REQUEST increases which causes more nodes to receive and process it. Thus, we expect containment to decrease and discovery cost to increase in environments where the average node degree increases.

Using the containment and discovery cost metrics, we examine the overall cost of Route Discovery in DSR. We focus on how the use of route caches and nonpropagating requests effect Route Discovery among nodes at the rectangular site, and then compare those results with data collected at the square site.

6.1 Overall Route Discovery Cost in All-Opt DSR

We began by studying the behavior of Route Discovery in scenarios using the rectangular site, where the average node degree was measured to be 11.5 neighbors. Route Discovery behaved as described in Sections 2 and 5, with All-Opt DSR sending a nonpropagating ROUTE REQUEST before transmitting a propagating ROUTE REQUEST if the nonpropagating request failed to provide a route within 30 ms. Table VI summarizes the discovery costs of propagating and nonpropagating requests broken out by the number of times a routing packet of the given type was originated (Og) or forwarded (Fw). When propagating and nonpropagating requests are used together in All-Opt DSR, the total containment for all requests is 68%, and the average discovery cost metric is nearly 17. The two types of ROUTE REQUESTS have very different behavior and costs, however, and so it is informative to analyze them separately.

An average of 316 propagating requests were initiated during each of the simulations. Examining only these requests, on average each request was forwarded 20 times and caused 10 ROUTE REPLY packets to be returned, yielding a containment metric of 41%. This means that most propagating requests involve a little more than half of the nodes in the network. The average discovery cost of a propagating request was nearly 53 transmissions. The fact that the origination of a single propagating ROUTE REQUEST results in the transmission of more than 50 DSR packets, even when the containment metric is 40%, indicates that Route Discovery has the potential to be a very expensive operation when not well contained.

Table VI Summary of Route Discovery costs (All-Opt DSR, rectangular site, constant motion)

	Nonpropagating	Propagating	Total
Request Og	957	316	1,273
Request Fw	0	6,115	6,115
Reply Og	3,912	3,215	7,127
Reply Fw	0	7,002	7,002
Containment	77.0%	41.0%	68.0%
Cost	5.09	52.69	16.90

In comparison to the propagating ROUTE REQUESTS, the nonpropagating requests have a containment metric of 77% and the average discovery cost drops to only 5 transmissions. A propagating request costs nearly 10 times more than a nonpropagating request because the total cost of a nonpropagating request is the one transmission of the ROUTE REQUEST plus one transmission for each of the neighbors that generates a ROUTE REPLY.

6.2 Replying from Cache

As described in Section 2.1, DSR contains Route Discoveries by allowing a node to short-circuit the outward propagation of a ROUTE REQUEST packet when it has a route to the request's target in its route cache. In order to discern the effect of replying from cache on the cost of Route Discovery, we re-ran all the scenarios with a modified version of DSR where replying from cache was disabled.

As shown in Table VII, the overall containment metric decreases to 10% when replying from cache is turned off. The cost of each discovery likewise increases by more than a factor of five to 102 packet transmissions per Route Discovery. The dramatic decrease in containment and increase in overhead argue strongly that allowing nodes to reply from their route cache is vital to limiting the cost of discovery. Even though the discovery cost quintupled, the packet delivery ratio of DSR with cache replies disabled was the same as in All-Opt DSR. While this is an encouraging result for on-demand Route Discovery, this may not be significant, as a different offered traffic load or communication pattern could suffer more from the dramatic increase in overhead.

6.3 Nonpropagating ROUTE REQUESTS

A second mechanism that may be used to reduce the cost of Route Discovery is an expanding ring search. Specifically, we consider the algorithm used by All-Opt DSR where each propagating ROUTE REQUEST is preceded by a nonpropagating ROUTE REQUEST, i.e., with a maximum propagation limit (TTL) of 1. To evaluate whether the nonpropagating ROUTE REQUEST accomplishes its intended purpose of allowing the initiator of the Route Discovery to quickly and inexpensively query the route caches of each of its neighbors, we experimented with a version of DSR in which the sending of nonpropagating requests has been disabled.

When DSR is modified to send only propagating ROUTE REQUESTS (Table VIII), the overall containment metric decreases from 68% to 58%, and the cost of a single Route Discovery doubles from 16 to 34 packets. These data argue strongly in favor of a two-phase Route Discovery that uses both nonpropagating and propagating requests, even though the tradeoff is an increase in latency when the nonpropagating request fails (Section 5).

6.4 Effects of Node Degree on Route Discovery

As previously mentioned, nodes in the rectangular site have an average node degree of 11.5. To evaluate the effect of node degree on Route Discovery, we performed the same experiments using the square site where nodes have an

Table VII Summary of Route Discovery costs (No reply from cache and no nonpropagating ROUTE REQUESTS, rectangular site, constant motion)

	Propagating
Request Og	871
Request Fw	39,471
Reply Og	8,413
Reply Fw	39,554
Containment	10%
Cost	102.52

Table VIII Summary of Route Discovery costs (no nonpropagating ROUTE REQUESTS, rectangular site, constant motion)

	Propagating
Request Og	776
Request Fw	8,812
Reply Og	8,077
Reply Fw	8,894
Containment	58 %
Cost	34.19

average node degree of 16.5. This increase in the number of neighbors stems from the fact that the rectangular site is only 300m wide and hence, much of the area covered by a node's transmission range lies outside the space where nodes are allowed to move to.

Our intuitive model of Route Discovery predicted that changing to the square site where nodes have greater degree should both decrease containment and increase discovery cost. As Table IX shows, however, containment does decrease by 6% from 68% to 62%, but the discovery cost remains relatively *unchanged*. Analyzing the nonpropagating and propagating requests separately makes the reasons for the discrepancy more apparent.

The reason the overall metrics do not follow our expectations is that the increased node degrees in the square site not only increase the number of nodes that overhear each ROUTE REQUEST packet (thus increasing the cost of the request), but also increase the number of nodes that overhear source routes used by their neighbors. This causes routing information about each destination to be spread more widely throughout the network than at the rectangular site. The greater spatial distribution of routing information allows nonpropagating requests to succeed more often in the square site, and prevents a greater number of the expensive propagating route requests. In the rectangular site, 24% of the ROUTE REQUESTS were propagating requests, while in the square site, only 12% of the Route Discoveries required a propagating ROUTE REQUEST. This difference makes the cost and containment metrics of nonpropagating requests dominate the overall overhead for the square site, thereby giving the appearance that the discovery cost and containment metrics are roughly equivalent on the two sites. However, Route Discovery tends to be more expensive and less well-contained in the square site.

7 Effects on Cache Consistency

All routing protocols which use on-demand Route Discovery must include some kind of route caching system, since the originator of a packet cannot afford the cost of doing a Route Discovery operation for every packet it wishes to

Table IX Summary of Route Discovery costs (All-Opt DSR, square site, constant node motion)

	Nonpropagating	Propagating	Total
Request Og	217	27	244
Request Fw	0	798	798
Reply Og	1,931	387	2,318
Reply Fw	0	584	584
Containment	67.0%	18.0%	62.0%
Cost	9.87	66.05	16.11

send. Once the originator discovers a route through the network, it must remember the route in some kind of cache for use in sending future packets. DSR, in particular, makes even greater use of the route cache, using it not only to cache routes for the purpose of originating packets, but also for the purpose of allowing nodes to answer ROUTE REQUESTS targeted at other nodes, as explained in Section 2.

When a cache is added to the system, however, the issue arises of how stale cache data is handled. In the context of a route cache, we call a route stale if any link in the route is broken, since a packet sent using the route will encounter a forwarding error when it attempts to traverse the broken link. Removing stale data from the cache of a node originating packets is critical, since any packet the node sends with a stale route will result in a ROUTE ERROR being returned, with a reasonable chance of the packet being dropped.

Allowing nodes to use the data in their route caches to issue ROUTE REPLY packets carries even greater risk, as nodes with stale cache data may return stale, incorrect routes that will pollute both that particular Route Discovery and potentially other nodes' caches.

For the purposes of analyzing cache behavior, we label the nodes as either *originator nodes* or *forwarder nodes*. Originator nodes are those that are the sources of packets in a connection, while forwarder nodes do not themselves originate data packets, but only act as routers to carry others' communications. As described in Section 3, there are 14 originator nodes (which may also serve as forwarders for other connections) and 36 forwarder nodes (which are not the originator of any connections) in the communication pattern used in our simulations. The route cache of an originator node behaves very differently from that of a forwarder node, since the originator actively invokes Route Discovery to maintain good routes to the nodes to which it sends packets, and it is the target of ROUTE ERRORS whenever it sends a packet with a broken route. Forwarder nodes must learn and correct routes opportunistically.

7.1 Contents of the Route Cache

The most basic measure of how well a node's route cache performs is the percentage of cache lookups that actually find a *good* cached route in the node's cache. This measure is a function of the percentage of good links in a node's cache and the overall cache hit rate. Figure 8 shows the average number of links in each node's cache, broken out into number of good links and bad links, for our constant node motion scenarios. On average, 84% of the links in caches were found to be good while sampling once per second. As expected, all of the bad links in the caches are due to node motion, and there are 0% bad links in the caches of nodes in the scenarios with no node motion. Originating nodes have a slightly greater number of links in their caches, due to the fact that they receive ROUTE REPLY packets from nodes throughout the network when they perform Route Discovery, whereas forwarding nodes only hear ROUTE REPLIES that travel past them on the way to an originating node. That the difference is slight argues that Route Discovery is successfully priming the caches of the forwarding nodes.

To determine if the routes in the nodes' caches are useful, we evaluated the cache hit rate for each node in our simulations and show this data in Figure 9. As described in Section 2, nodes use their cache when originating packets, when deciding whether to return a cache reply in response to a received ROUTE REQUEST, and when salvaging. Originating nodes have a significantly higher hit rate than forwarding nodes, since the vast majority of the cache lookups that originating nodes perform are for packets to destinations for which they have performed Route Discovery. The hit rate of forwarding nodes is sensitive to both the geometry of the space and the communication pattern, since a node's hit rate will go up if the ROUTE REQUESTS that it overhears are for destinations in close spatial proximity, and down as it processes ROUTE REQUESTS for nodes far away from it. Forwarding nodes in the rectangular site have an average hit rate of only 55%, which, though low, is consistent with the philosophy of on-demand Route Discovery: there is no reason to have a route to a destination unless you are originating packets to it. In the square site, the average hit rate for forwarding nodes increases to 62%, since nodes are fewer hops apart and there is greater shared knowledge as nodes overhear a greater number of useful source routes that they can then add to their route caches (Section 2.3).

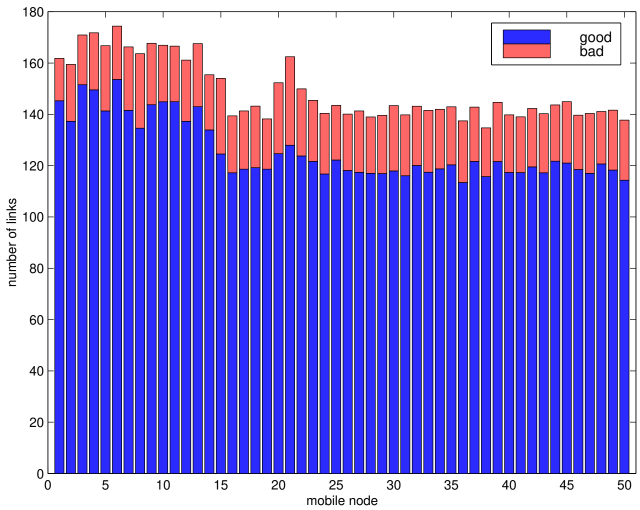

Figure 8 Average number of links in each node's cache over simulation runs for All-Opt DSR (rectangular site, constant motion). Nodes 1 through 14 are originating nodes.

7.2 The Quality of Route Replies

Nodes in the network use their caches to generate ROUTE REPLY packets in response to other nodes' Route Discoveries, in order to limit the propagation of ROUTE REQUESTS (Section 2.3). Given that there may be some stale data in nodes' caches, the question arises as to the extent of the cache pollution that the Route Discovery process might cause as bad replies from cache are returned to the initiator of a Route Discovery and overheard by forwarding nodes. We collected statistics for this over our scenarios using All-Opt DSR and report these results in Table X. Using the nominal 250 m transmission range of the radios modeled in our simulations to determine if the links are good, 40% of the ROUTE REPLIES received by the initiator of a Route Discovery contain routes that would not work if used.

That 40% of ROUTE REPLYs contain broken routes is not surprising considering how cached information is learned, and why a Route Discovery is initiated. Route Discovery is performed when a node wants to send a packet to a destination to which it does not have any routes. This occurs because either the node has never had a route to the

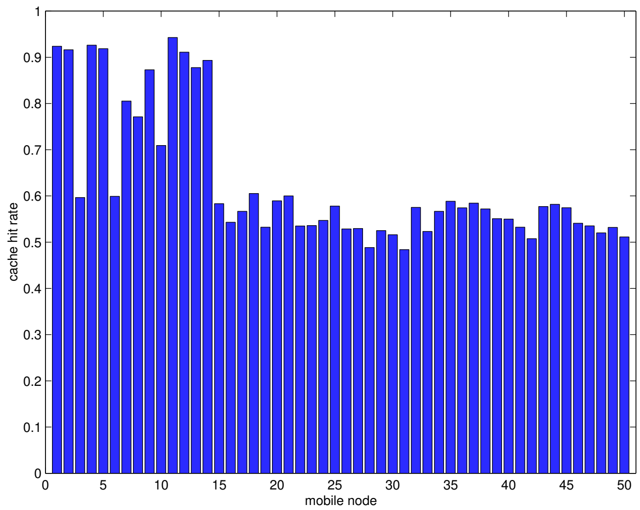

Figure 9 Cache hit rate for All-Opt DSR (rectangular site, constant motion). Nodes 1 through 14 are originating nodes.

destination, or because its last working route to the destination just broke, many times as the result of some major network topology change that altered the available paths to reach the destination. Since many nodes will have cached routes to that destination with links in common, the breakage of any of those links will cause many of the cached replies generated for that destination to be bad. In our simulations of All-OptDSR, the overwhelming majority of ROUTE REPLY packets are based on cached data, and only 59% of those replies carry correct routes. Though 84% of the links in the nodes' caches are good, the probability of a route being good given that it has been retrieved from a forwarding node's cache in our simulations is only 56%. Even replies from the target itself are not 100% correct, since routes can change while the REPLY propagates back to the requester.

If replying from cache is disabled and only propagating ROUTE REQUESTS are sent, a total of 8,413 ROUTE REPLY messages are received by initiators, of which 93% are good. While disabling the replying from cache mechanism significantly increases the number of working routes an initiator receives, it also increases overhead and latency as described in Sections 5 and 6.

The fact that DSR continues to have a very high packet delivery ratio (Figure 1) even when 40% of the routes returned by Route Discovery are incorrect, argues that Route Maintenance is performing well so that the discovery and dissemination of bad routes does not result in the continued use of bad routes.

8 Conclusions and Future Work

In this paper, we have presented a detailed examination of the fine-grain performance of on-demand routing protocols. We analyzed the latency of Route Discovery, the cost of Route Discovery, and the effect of on-demand behavior on routing cache consistency, drawing our examples for study from the Dynamic Source Routing (DSR) protocol. DSR provides a good source of examples for this study, since it is based entirely on on-demand behavior, and thus our results are not affected by any periodic or background activity within the protocol.

We have identified several mechanisms that can be used to reduce the cost of Route Discovery and isolated the performance improvement due to each of these mechanisms. As expected, we found that a naive approach to Route Discovery is expensive, both in terms of latency and the number of packets that are sent to complete the discovery. However, we have shown how the twin techniques of using route caches to answer route requests and using nonpropagating requests to limit the search performed by the routing protocol can reduce the mean latency of Route Discovery from 403 ms to well under 40 ms and decrease the total overhead from 102 packets per discovery to 17 packets. Examining the role of the route cache in an on-demand routing protocol, we found that the cache is able to acquire useful information about the overall network topology solely by extracting routing information from packets that pass through and near it. Nodes were able to learn about sufficiently many destinations to achieve a cache hit rate of 55%, whether the nodes were initiating Route Discovery themselves or not. Efficient Route Maintenance is critical in all systems with route caches, as we found that 16% of the links in the nodes' caches were stale, and up to 41% of the ROUTE REPLIES sent based on cached data contained broken routes. The DSR Route Maintenance techniques met the challenge however, as more than 90% of the data packets were successfully delivered even at constant node motion.

Table X The correctness of source routes returned by Route Discovery.

	# replies	% good
from target	119	95.2 %
from cache	6809	59.4%
total	6928	60.0%

Many of the techniques and lessons learned from this work can be applied to the other on-demand routing protocols, such as AODV, TORA, and ZRP. In particular, by adding a mechanism to share routing information among nodes in the network (e.g., having a subset of the data packets each record the route it takes through the network), protocols such as these three could make greater use of their route caches to control the overhead cost of Route Discovery.

At present, we are extending this work by exploring the effects of varying the number of nodes, the rate and pattern of node movement, and the type of communication pattern. We are also working to examine and isolate the effects of the other optimizations to DSR discussed in this paper, including gratuitous ROUTE REPLIES, salvaging, gratuitous ROUTE ERRORS, snooping, and tapping. In addition, we are presently experimenting with the idea of using a link-state cache which would allow us to combine the results of multiple Route Discoveries.

9 Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under CAREER Award NCR-9502725, by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061, and by the AT&T Foundation under a Special Purpose Grant in Science and Engineering. David Maltz was also supported under an Intel Graduate Fellowship. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, AFMC, DARPA, the AT&T Foundation, Intel, Carnegie Mellon University, or the U.S. Government.

References

[1] Josh Broch, David B. Johnson, and David A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-01.txt, December 1998. Work in progress.

[2] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, Dallas, TX, October 1998.

[3] M. Scott Corson and Anthony Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, 1(1):61–81, February 1995.

[4] Kevin Fall and Kannan Varadhan, editors. *ns* Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 1999. Available from http://www-mash.cs.berkeley.edu/ns/.

[5] Zygmunt J. Haas and Marc R. Pearlman. A New Routing Protocol for the Reconfigurable Wireless Networks. In *Proceedings of the 6th IEEE International Conference on Universal Personal Communications*, October 1997.

[6] Zygmunt J. Haas and Marc R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. In *Proceedings of ACM SIGCOMM'98*, pages 167–177, September 1998.

[7] Zygmunt J. Haas and Marc R. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. Internet-Draft, draft-ietf-manet-zone-zrp-01.txt, August 1998. Work in progress.

[8] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[9] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[10] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[11] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, January 1987.

[12] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of INFOCOM'97*, pages 1405–1413, April 1997.

[13] Vincent D. Park and M. Scott Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1: Functional Specification. Internet-Draft, draft-ietf-manet-tora-spec-01.txt, August 1998. Work in progress.

[14] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994. A revised version of the paper is available from http://www.cs.umd.edu/projects/mcml/pub.html.

[15] Charles E. Perkins and Elizabeth M. Royer. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet-Draft, draft-ietf-manet-aodv-02.txt, November 1998. Work in progress.

[16] Bruce Tuch. Development of WaveLAN, an ISM Band Wireless LAN. *AT&T Technical Journal*, 72(4):27–33, July/August 1993.

10 Biographies

David A. Maltz primary interest is in designing and evaluating protocols for ad hoc networks. As an active participant in the IETF, he is working to create and standardize protocols for both Mobile IP and ad hoc networks. His other interests include computer-supported cooperative work and collaborative information filtering. He is currently a Ph.D. candidate in the Computer Science Department at Carnegie Mellon University supported by an Intel Graduate Fellowship, and was previously supported by an IBM Cooperative Fellowship. He received the S.B. and S.M. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1994. E-mail: dmaltz@cs.cmu.edu

Josh Broch authored the first implementation of Mobile IP for IPv6 and is a co-author of a leading IETF proposal for multi-hop routing in ad hoc wireless networks. His recent work focuses on the development and analysis of routing protocols for ad hoc networks. He is currently a Ph.D. candidate in Electrical and Computer Engineering at Carnegie Mellon University. Josh received a B.S. in Mathematics and Computer Science from Barry University in 1995 and a M.S. in Information Networking from Carnegie Mellon University in 1996. He worked for five years as a Systems Engineer at Connections for Business (Hollywood, FL) and has interned at Oak Ridge National Lab and Microsoft. E-mail: broch@andrew.cmu.edu

Jorjeta Jetcheva, joined the Ph.D. program in Computer Science at Carnegie Mellon University in 1997. Since then, she has been working on design, analysis, simulation and measurement of routing protocols for multi-hop ad hoc networks. Her research interests include ad hoc networking, Mobile IP and network performance analysis. Jorjeta received a B.A. in Computer Science and Mathematics, with a summa cum laude in Computer Science from Mount Holyoke College in 1997. She has interned at the DataViews Corporation, Northampton, MA, where she worked on dynamic data visualization software. E-mail: jorjeta@cs.cmu.edu

David B. Johnson is an Associate Professor in the School of Computer Science at Carnegie Mellon University. He also holds a courtesy faculty appointment in the Electrical and Computer Engineering Department at Carnegie Mellon, and is a member of CMU's Information Networking Institute. His research interests include network protocols, distributed systems, and operating systems. Prior to joining the faculty at CMU in 1992, he was on the faculty at Rice University for three years as a Research Scientist and Lecturer in the Computer Science Department. He received a B.A. in computer science and mathematical sciences in 1982, an M.S. in computer science in 1985, and a Ph.D. in computer science in 1990, all from Rice University.

Professor Johnson is currently leading the Monarch Project at Carnegie Mellon University, developing adaptive networking protocols and architectures to allow truly seamless wireless and mobile networking. Related to this research, he has been active in the IETF for many years, were he was one of the main designers of the IETF Mobile IPv4 protocol and is the primary designer of Mobile IPv6. Professor Johnson was Program Chair for MobiCom'97 and Technical Vice Chair for Mobile Systems for ICDCS'99, and has served as a member of the Technical Program Committee for over 15 international conferences and workshops. He is an Executive Committee member and the Treasurer for ACM SIGMOBILE, is an Area Editor for the ACM/Baltzer journal Mobile Networks and Applications (MONET) and the ACM SIGMOBILE magazine Mobile Computing and Communications Review (MC2R), and has been a Guest Editor for issues of several journals. He is a member of the ACM, IEEE Computer Society, IEEE Communications Society, USENIX, Sigma Xi, and the Internet Society. E-mail: dbj@cs.cmu.edu