Location Proxies and Intermediate Node Forwarding for Practical Geographic Forwarding

Douglas S. J. De Couto and Robert Morris*

Abstract

Two main problems prevent the deployment of geographic forwarding in real systems: geographic forwarding requires that all nodes know their locations, and it has trouble routing around local dead ends. This paper presents practical solutions to each problem.

The *location proxy* technique allows a node that does not know its location to find a nearby location aware node to use as a proxy for geographic forwarding. The technique works well over a large range of densities of location aware nodes, and allows a tradeoff between bandwidth used for routing information and expense of providing location information.

The *intermediate node forwarding* (INF) mechanism is a probabilistic solution for routing around bad geographic topologies via intermediate geographic locations. Existing solutions unrealistically assume that nodes have identical radio propagation; INF works on a restricted set of situations but makes assumptions that better match reality.

Experiments using the *ns* simulator show that location proxies and INF are effective enough to make geographic forwarding practical. We believe geographic forwarding will enable scalable ad hoc networking.

1 Introduction

Ad hoc networks are attractive because they are easy to deploy: no network administration is required when computers join or leave the network. Geographic forwarding is a scalable, low overhead technique for building mobile ad hoc wireless networks. By using geographic locations to route packets, geographic forwarding can make purely local decisions to route packets, avoiding the routing protocol overhead incurred by other ad hoc routing protocols in large networks [14, 12]. Furthermore, geographic forwarding provides network participants with a rich datatype: location. Location information enables context-sensitive computing and many location specific applications, such as service and resource discovery and mapping.

Although geographic forwarding has the potential to be the foundation for scalable ad hoc networks, it has three main problems: first, geographic forwarding can only send data to network clients with known locations; second, geographic forwarding assumes each computer on the network knows its own position for making forwarding decisions; and third, geographic forwarding performs poorly with some network topologies. The first problem requires that a working geographic forwarding system include a location service to map destinations to locations, and has been solved in [14]. We address the second two problems in this paper, with *location proxies* and *intermediate node forwarding*.

1.1 Location Information

An important part of a geographic forwarding system is the position information infrastructure: each device must determine its own location. The Global Positioning System (GPS) [2] is a potential source of this information, but doesn't work well in common situations. Adding a GPS receiver to a small mobile device will increase the device's weight, size, and cost. More importantly, GPS receivers will not work in the areas where most computing devices are concentrated: inside homes, offices, and enclosed public spaces.

One possible solution to this problem is to use positioning technologies such as Cricket [17] or BAT [10]. Unfortunately, the power, cost, and size constraints of these technologies, like GPS, can be the most burdensome on the small

*Parallel and Distributed Operating Systems Group, MIT Laboratory for Computer Science. Email: {decouto, rtm}@lcs.mit.edu. This research was funded in part by NTT corporation under the NTT-MIT collaboration.

mobile computing devices most in need of them. It is unlikely that a single positioning technology will be adopted in all areas, and mobile nodes will not be able to accommodate all the possible positioning technologies. Thus even the most well-equipped mobile device may be unable to directly determine its location as its user moves around.

Although many devices won't know their locations, some will. Any computer in a machine room or on a desk can be statically configured with its location, just as today it is statically configured with an IP address, netmask, and gateway address. Other machines may learn their own positions using indoor location tracking and notification systems. The location proxy technique presented in this paper allows machines with location information to serve as location proxies for clients that do not have location information, connecting the clients to the geographic forwarding network using a local routing protocol.

Finally, we would like our network performance to scale as we add more location sensors to network nodes, or add more location ignorant nodes. Then we are free to choose the right tradeoff between supplying network participants with location information (e.g. installing and supporting location sensor infrastructure), or compensating for the lack of location information with routing protocol mechanisms.

1.2 Routing Holes

Geographic forwarding works best when the spatial density of network nodes is high relative to the radio coverage. Otherwise, we can find cases where geographic forwarding's greedy choices fail to find routes. Geographic forwarding will fail at a node when the packet has to travel backwards around a topology *hole*—when no neighbor is closer to the destination. The device currently forwarding the packet has no routes to any devices that are closer than itself to the packet's destination. A practical geographic forwarding system must handle these cases, as node distributions will vary unpredictably in the real world.

Although there are theoretically guaranteed techniques [12, 4] to route around topology holes, they assume that all nodes have radios with identical ranges. This is not likely to be even approximately true, since obstructions and interference drastically modify radio ranges. The intermediate node forwarding (INF) technique presented in this paper provides a probabilistic solution for handling topology holes, and does not assume uniform radio ranges.

1.3 Paper Organization

Section 2 discusses the details of the basic routing protocol, which location proxies and INF build upon. Section 3 describes the location proxy protocol, and Section 4 describes the details of INF. We present simulation results that show the performance and costs of location proxies and INF in Section 5, discuss related work in Section 6, and conclude in Section 7.

2 Basic Routing Protocol

The location proxy and intermediate node forwarding techniques are both extensions of a basic routing protocol. The basic protocol uses geographic forwarding [8] in conjunction with a location service such as GLS [14]. A limited-radius variant of the Dynamic Destination-Sequenced Distance-Vector (DSDV) protocol [15] is used to increase the number of neighbors each node can use to make geographic forwarding decisions.

In order for geographic forwarding to work well, nodes must be aware of their neighbors' positions, and ideally they should have neighbors in a wide variety of directions. Regular DSDV is a global loop-free distance vector routing protocol; we modify DSDV so that route entries are only propagated a fixed number of hops from the route's destination. The modified DSDV works well with geographic forwarding because it pro-actively discovers nearby nodes. Reactive protocols such as DSR [11] or AODV [16] are hard to use in this context because they search for particular nodes, while geographic forwarding needs to search for the node closest to the destination.

The modified DSDV routing protocol has constant per node overhead, since route messages all have a fixed maximum size, and are sent at a fixed maximum rate by each node. Although we place no explicit limit on the size of the DSDV routing table, the maximum table size experienced in the network is limited by the DSDV hop radius, the spatial density of nodes, and the radio range. Figure 1 shows the route message and route entry formats.

When a node sends a new packet onto the network, the packet header's source and destination location fields (Figure 1) are filled in. The basic routing protocol assumes that every node knows its own location, and that a location service is available to provide destination locations. When forwarding or originating a packet, a node uses the algorithm

Route broadcast header	Route entry fields	Data packet header
Transmitter ID	Destination ID	Source ID
Transmitter location	Destination location	Source location
Sequence number	Sequence number	Destination ID
Entries[ ]	Next hop	Destination location
	Hop count	

Figure 1: Packet and route entry formats. Data packets are addressed with the destination's ID and geographical location; the source's location is included so the destination can reply

$d = p.\text{dest\_id}$
if $R = \emptyset \vee$ interface output queue is full
  $\text{drop}(p)$
else if $d \in R$
  $\text{transmit}(p, R[d].\text{next\_hop})$
else
  $l_d = p.\text{dest\_loc}$
  choose $n_{i+1} \in R$ to minimize $\text{dist}(n_{i+1}, l_d)$
  if $\text{dist}(n_{i+1}, l_d) \geq \text{dist}(n_i, l_d)$
    $\text{drop}(p)$
  else
    $\text{transmit}(p, R[n_{i+1}].\text{next\_hop})$

(a) Basic protocol algorithm.

$d = p.\text{dest\_id}$
if interface output queue is full
  $\text{drop}(p)$
else if $d \in R$
  $\text{transmit}(p, R[d].\text{next\_hop})$
else
  $R' = \{r \mid r \in R \wedge R[r].\text{has\_loc}\}$
  if $R' = \emptyset$
    $\text{drop}(p)$
  else
    $l_d = p.\text{dest\_loc}$
    choose $n_{i+1} \in R'$ to minimize $\text{dist}(n_{i+1}, l_d)$
    if $n_i.\text{has\_loc} \wedge \text{dist}(n_{i+1}, l_d) \geq \text{dist}(n_i, l_d)$
      $\text{drop}(p)$
    else
      $\text{transmit}(p, R[n_{i+1}].\text{next\_hop})$

(b) Proxy algorithm.

Figure 2: The algorithms for choosing packet p's next hop node at each node n_i. The routing table is R. A node's neighbor table does not contain a route to the node itself. $n.\text{has\_loc}$ is true iff n is a proxy.

in Figure 2a to choose the packet's next hop. The node first checks its local DSDV neighbor table for a route to the destination. If a DSDV route exists, the node forwards the packet to the next hop indicated in the routing entry. Otherwise the node tries to pick a next hop by searching the DSDV neighbor table using geographic forwarding rules.

3 Location Proxies

Using only basic geographic forwarding, nodes that do not know their own position cannot participate in the network. The *location proxy* technique uses an adaptive local routing protocol to extend the geographic forwarding network to location ignorant nodes. Any node that knows its own position (a *location aware* node) can serve as a location *proxy* for nodes that do not know their position (*location ignorant* nodes). We will refer to location aware nodes as proxies, and location ignorant nodes as clients. Proxy nodes enable us to use geographic forwarding for large-scale routing, while the local protocol takes care of routing between proxies and clients, and between proxies.

A location ignorant node selects a nearby location aware node as its location proxy. To receive packets, the node advertises its proxy's location as its own; this causes a packet addressed to the node to be delivered to its proxy via geographic forwarding. The proxy in turn forwards the packet to the node using the local routing protocol. When forwarding or originating packets, all nodes use the local protocol's neighbor table to move packets towards the best next node for geographic forwarding. Figure 2b shows the modifications to the basic geographic forwarding algorithm that are needed when some nodes do not know their locations.

We must solve three problems for location proxies to work. First, in order to use geographic forwarding and advertise a location to the location service, a location ignorant node must learn the route to at least one location aware node, which will serve as its proxy. Second, in order for the node to receive packets, its proxy must learn a route to it.

Third, proxies must be able to geographically forward packets amongst themselves by way of location ignorant nodes when neccessary, to keep the geographic forwarding network connected. Our location proxy technique solves each of these problems.

3.1 Finding Location Proxies

In order for location ignorant nodes to find proxies, they run the local DSDV routing protocol, with the following two modifications:

- Keep any route advertisement for a location aware node no matter how many hops to that node.
- Dedicate a fraction of regular route table broadcasts to routes for location aware nodes. Advertise these routes in a round-robin fashion.

These two modifications ensure that every location ignorant node eventually learns a route to a potential location proxy. Any location ignorant node $\mathbf{N}_1$ that is one hop from a location aware node **P** will have a route to **P**, thanks to **P**'s route table broadcasts. Because $\mathbf{N}_1$ will propagate its one-hop route for **P**, any node $\mathbf{N}_2$ within $\mathbf{N}_1$'s radio range will have a route to **P** that is no longer than two hops. Similarly, any of $\mathbf{N}_2$'s neighbors will have a route to **P** that is at most three hops, and so on.

Since packets are forwarded through proxies and other location aware nodes, if proxies do not have routes to each other, the network is effectively disconnected. If a packet's destination is not in a proxy's local routing table, the proxy must forward the packet to another location aware node, because node locations must be compared to choose a next hop. These local routing protocol modifications also allow proxies to learn routes to other proxies, which keeps the network connected.

Each location ignorant node chooses as its proxy the location aware node in its routing table which is the least number of hops away. The closest potential proxy is chosen, to minimize the number of extra hops that a packet must travel using the local routing protocol. However, other factors could be considered, such as the forwarding load, network capacity, and battery life of the potential proxy. If no such node exists, then the location ignorant node has no proxy.

These modifications adaptively extend the local DSDV's radius limit to ensure that every node hears about nearby location aware nodes. If a fixed maximum DSDV radius were used, the maximum radius parameter would be difficult to set.

3.2 Receiving Packets From Proxies

Once each location ignorant node has a proxy, it can forward outgoing traffic through that proxy, and advertise the proxy's location as its own in order to receive packets. Incoming packets will be delivered to the proxy via geographic forwarding. However, a node's proxy may not have a route to that node: the propagation of a potential proxy's route entry only sets up a one-way route. To ensure that each location aware node has a route to the nodes that have chosen it as a proxy, every client ensures that its route is propagated for enough hops to reach the proxy. An extra *radius* field is added to route entries. This field is advertised in each node's routing table broadcast, and is handled as follows for client entries:

1. Each location ignorant node sets its own radius field to the number of hops to its proxy. Location aware nodes and nodes without proxies set the field to zero, so that it is ignored.
2. When a node processes a route advertisement entry with a non-zero radius field, it keeps the route entry and decrements the field.
3. Each node includes any route entry with a non-zero radius field in its regular route table broadcasts.

These three steps ensure that if a location ignorant node chooses a proxy that is h hops away, that node's route entry will be propagated to every node within h hops, including the proxy itself. Once the proxy learns a route to the node, it can use local routing to deliver any packets for the node that arrive at the proxy via geographic forwarding.

There is no explicit information in a packet that informs a proxy it has received one of its clients' packets. Nevertheless, the protocol is still loop free in the steady state. If a packet arrives at the destination's proxy, and the proxy

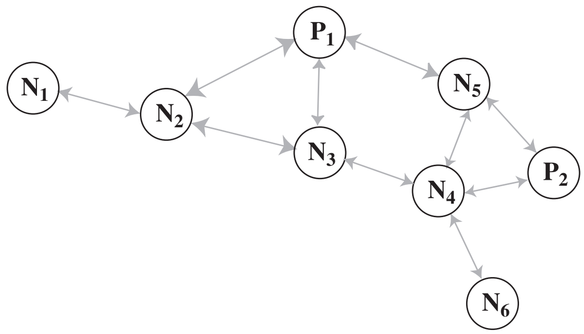


Figure 3: Packets need not travel out of the way to go through proxies. In this example with a two hop DSDV radius, $\mathbf{N}_1$ is sending to $\mathbf{N}_6$; their proxies are $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively. Because of the proxy route propagation rules, $\mathbf{N}_2$ has an explicit route to $\mathbf{P}_2$, via $\mathbf{N}_3$. Therefore packets will travel from $\mathbf{N}_1$ to $\mathbf{N}_3$, bound for $\mathbf{N}_6$'s proxy. But $\mathbf{N}_3$ has an explicit route to $\mathbf{N}_6$, and the packets avoid both $\mathbf{P}_1$ and $\mathbf{P}_2$.

has no entry for that destination in its neighbor table, the packet will be dropped. This is because the destination's advertised location is the proxy's location. No entry in the proxy's neighbor table could be closer to that location than the proxy itself, and referring to Figure 2b, we see that the packet will be dropped in this case, rather than looping.

3.3 Discussion

One alternative to using location proxies would be to run some sort of position estimation protocol (PEP) on the network. A PEP might use information such as nearby location aware nodes and radio propagation times to produce a position estimate for each location ignorant node. This estimate would be used when directly measured positions are not available for making geographic forwarding decisions. Although some sort of PEP may be neccessary for applications in mobile networks, and could be used by the geographic forwarding network when available, it is unneccessary—the proxy technique is adequate for delivering packets. While creative approaches [6] may produce a PEP accurate to within one radio range, no known system is reliable enough to use by itself for geographic forwarding.

A client using the location proxy system does not actually send every packet through its proxy, but towards the location aware node closest to the destination. Furthermore, as a packet travels towards the chosen location aware node, it may be forwarded by a node that knows of a location aware node even closer to the destination. Thus the location aware nodes act as guides for geographic forwarding, but do not in general need to forward the packets. Figure 3 shows an example.

The limited flooding used to build routes from proxies to clients is robust, but advertises the route to many nodes that do not need to know it. We considered having clients unicast their routes along the path to the proxy, ensuring that only the proxy and nodes between the client and proxy deal with the client's route. But unicast client routes could be fragile as nodes along the route move, and offer fewer opportunities for routing shortcuts, as discussed above. A directed flood would decrease route fragility, especially when guided by accurate node movement predictions.

3.3.1 Proxy Route Propagation

The particular rule for propagating proxy routes has a drawback: it ensures that all client nodes that form a connected subgraph will learn and advertise all the same proxy routes. With some kinds of network topologies, this could mean that every client learns about every proxy—proxy routes would be flooded globally. We considered other alternatives rules for propagating proxy routes, but they all potentially resulted in a disconnected network.

We first considered that each client should only remember and propagate the k closest proxies, in terms of hops. This rule guarantees that proxy routes are propagated exactly far enough so that every client learns a route to some proxy, and no farther. Unfortunately, for any value of k we can produce a network topology where this rule causes the proxies, and therefore the network, to be disconnected. If the network topology were constrained so that all proxies were connected, forming a geographic forwarding infrastructure, then this rule would be ideal. One example would

Intermediate location
INF mode

Figure 4: Intermediate node forwarding header additions.

be a metropolitan rooftop network constructed of fixed nodes with GPS receivers, providing infrastructure for highly mobile nodes without location sensors.

Another solution might be for proxies to selectively retain and propagate proxy routes using heuristics. For example, clients might advertise closer proxies more often, and probabilistically remember proxy routes based on their hop distance. But any rule that constrains the flooding scope of proxy routes may also partition the network by failing to propagate a critical proxy route.

4 Intermediate Node Forwarding

Although geographic forwarding works well in uniformly dense networks, it handles large *holes* badly: cases when a forwarding node must drop a packet because there is no better next hop through which to send the packet. The intermediate node forwarding (INF) scheme allows nodes originating packets to probabilistically route packets around holes. The basic idea is that when using INF, nodes pick random intermediate points through which to forward their packets. Packets are routed from the source to the intermediate point using geographic forwarding, and from the intermediate point to the destination using geographic forwarding again. The intermediate location serves as a weak source route. Eventually, an intermediate point can be chosen so that packets can be sent far enough out of the way of holes and other bad network topologies.

Nodes do not normally send packets using INF. However, if packets are unable to reach a destination using geographic forwarding, a sending node starts using INF for that destination: it picks an intermediate location and labels packets to the destination with the intermediate location. If packets still fail to reach a destination using INF, the node picks a new intermediate location. For the situations in which this approach works, the source node will eventually pick an intermediate point that causes packets to be routed around an intervening hole.

The following subsections describe the detailed mechanisms of INF, and discuss variations on these mechanisms.

4.1 Forwarding Details

We extend the geographic forwarding protocol by adding two new fields to data packet headers: *INF mode*, and *intermediate location* (Figure 4). Each node also maintains an INF table, which maps destination nodes to intermediate locations. Entries in this table are periodically expired.

When a node originates a packet, it checks to see if there is an entry for the packet's destination in the INF table. If so, the packet's INF mode is set to *TO-INF*, and the intermediate location is copied into the packet header from the INF table. If there is no entry in the INF table, the INF mode is set to *NO-INF*. When a node forwards a packet, it makes its forwarding decision based on the packet's INF mode. If the packet is in the *TO-INF* mode, the packet is forwarded to the intermediate location; otherwise, the packet is forwarded to the destination location.

If a node forwarding a *TO-INF* packet has no neighbor closer than itself to the intermediate point, it switches the packet to *FROM-INF* mode. From then on, the packet's real destination is used to make forwarding decisions.

4.2 Detecting Packet Drops

Basic geographic forwarding does not provide any feedback about packet drops. Therefore, we also extend the geographic forwarding protocol by adding negative acknowledgement packets (NAKs): when a forwarding node drops a packet due to lack of a good next hop, the forwarding node sends a NAK to the packet's original sender. NAKs are also routed using INF: they are forwarded geographically back through the same intermediate location (if any) as the dropped packet. To facilitate NAK routing, nodes that switch a packet's INF mode from *TO-INF* to *FROM-INF* before the packet is near the intermediate location must copy their location into the packet's intermediate location field. Otherwise, NAKs would be routed to the originally chosen intermediate point, which is not where the original packet traveled.

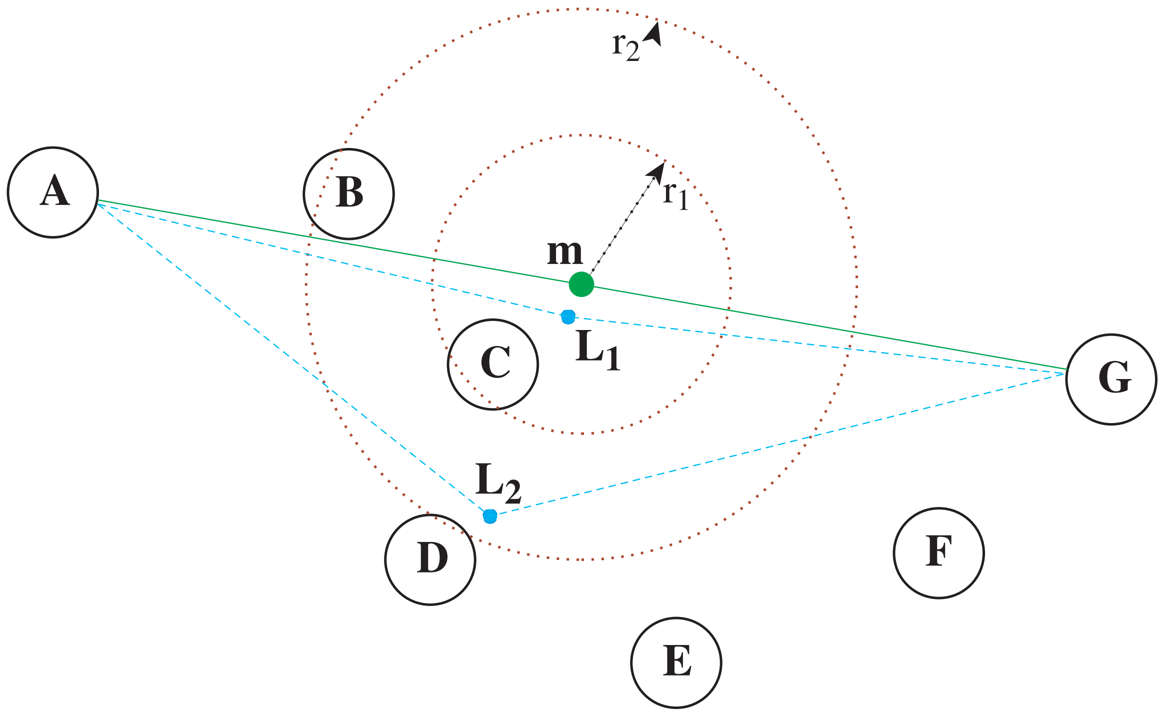


Figure 5: Intermediate forwarding example. $\mathbf{A}$ is sending to $\mathbf{G}$. Their midpoint is $\mathbf{m}$. Each node is only in range of its closest neighbors, and there is a route $\mathbf{ABCDEFG}$. Packets initially traverse $\mathbf{ABC}$, until they are dropped: $\mathbf{C}$ is closer to $\mathbf{G}$ than $\mathbf{D}$. $\mathbf{A}$ receives the NAK, and first initiates INF with a radius of $\mathbf{r}_1$, with $\mathbf{L}_1$ as the intermediate location. Packets traverse $\mathbf{ABC}$, and are again dropped at $\mathbf{C}$: $\mathbf{C}$ is close enough to $\mathbf{L}_1$ to switch packets out of *TO-INF* mode, but has no neighbor closer to $\mathbf{G}$. $\mathbf{A}$ chooses the new intermediate location $\mathbf{L}_2$ from the disc with radius $\mathbf{r}_2$. Packets can now make it to $\mathbf{G}$: they travel via geographic forwarding to $\mathbf{D}$, which switches them into *FROM-INF* mode, and sends them via $\mathbf{E}$ and $\mathbf{F}$.

4.3 Choosing Intermediate Locations

A node first uses INF to reach a destination after it receives a NAK for a packet to that destination. The node then creates an entry for the destination in the INF table. The intermediate location for a destination is chosen randomly from a disc centered around the location halfway between the sending node and the destination, as shown in Figure 5. The radius of this disc is initially set to one quarter of the distance between the sending and destination nodes; successive NAKs received for the same destination cause the radius to be doubled, and a new intermediate location to be chosen from within the larger disc. The disc radius for each destination is stored in the INF table with that destination's intermediate location.

4.4 Discussion

The INF algorithm here is not fully general: it is possible to construct multiple-hole scenarios in which paths exist that INF cannot find. On the other hand, INF does work well in some common multiple-hole situations; Section 5 shows that it improves delivery rates in grids of city blocks.

The version of INF discussed here uses explicit NAKs to trigger INF. A real system might add timeouts, perhaps based on hints from upper layer protocols; this would help if the routing of a NAK failed.

Another key design decision is when to pick the intermediate points. We always double the radius of the disc when NAKs are received. A variation would be to only double the radius every k NAKs. The other NAKs would still cause a new intermediate point to be chosen, but from a disc with same radius.

If one intermediate location is not sufficient to route a packet to its destination, we can use multiple intermediate points for routing the packet. If extra intermediate points are chosen by nodes other than the the packet's originator,

Radio	
MAC	IEEE 802.11
RTS/CTS	For all unicast packets
Nominal range	250 meters
Radio capacity	2 Mbps

Mobility	
Movement model	Random waypoint
Maximum speed	10 m/s
Pause time	0 seconds

Local Routing Protocol	
Route broadcast period	2 seconds
Route expiration timer	18 seconds
Triggered update period	1 second
Route entry lifetime	30 seconds
DSDV radius	2 hops
Maximum route message size	1000 bytes (22 entries)
Maximum route table size	unlimited

Location Service	
Update rate	0.25 seconds
Lookup latency	0 seconds
Protocol overhead	none

Table 1: Shared simulation parameters. All simulations use the above values of the listed parameters; other parameters such as simulation size, density, and time vary.

they must be chosen consistently to avoid loops.

Finally, when checking the INF table for a new destination, we could use the INF information from any entry whose destination location is near to the current destination's location. Packets to both of these destinations will probably travel through the same network topology, so they may be able to use the same intermediate point.

5 Evaluation

We performed a series of simulations to evaluate the performance of the location proxy protocol and INF. We implemented the protocols with version 2.1b1 of the *ns* simulator [7] and the CMU wireless extensions [9]. We do not provide results for other protocols such as AODV and DSR, since they were designed for relatively small scale networks. Previous work [14, 12, 5] shows that these protocols require too much routing overhead to scale to large networks with hundreds of mobile nodes.

The simulations use a perfect location service to obtain destination locations for the geographic forwarding protocol. This service has no network or computation costs. Nodes may periodically advertise a new location to the location service. However, nodes may also refrain from updating their position in the location service, including when they first join the network. Therefore the location service may not know the location of a node if that node has never advertised a location. Every node can lookup another node's most recently advertised location (if any) at any time.

5.1 Simulation Parameters

Unless noted otherwise, all of our simulations use a common set of parameters for the radios, DSDV local routing protocol, and location service. These parameters are summarized in Table 1. All simulations occur in a square universe.

All traffic is constant bit rate traffic. In each simulation, half of the nodes initiate conversations to a randomly chosen node at a randomly chosen time. Each conversation is 80 packets of 128 bytes each, sent at 4 packets per second.

We use the random waypoint model [5] to model mobile nodes. In this model, nodes start in uniformly randomly placed locations; each node then randomly chooses a destination in the simulation universe, and moves to that destination at a randomly chosen speed. Upon arrival, the node may pause movement for a random period of time before choosing a new destination; in our simulations nodes do not pause.

Unfortunately, this model produces an uneven distribution of nodes across the universe. Because each node picks its destination location from a finite universe, it is more likely to travel away from the closest edge of the universe at any time. A fair model would allow nodes to cross edges, leaving the universe at one edge and reentering at the same point on the opposite edge. To disambiguate movement directions, nodes would have to follow a rule about how to reach each destination, such as always taking the shortest path. Alternatively, nodes could move unambiguously by choosing a destination direction and distance, rather than location.

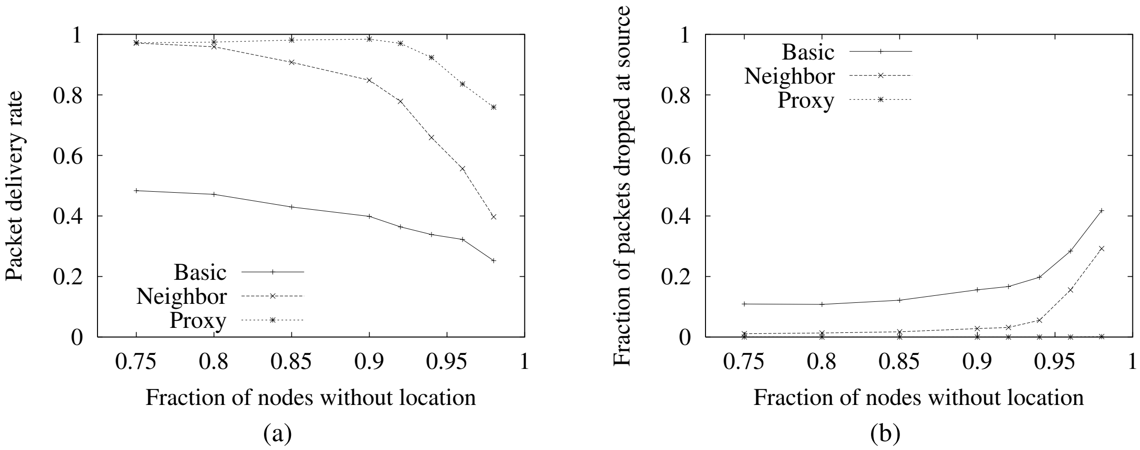


Figure 6: (a) The location proxy technique delivers more packets. Proxies make geographic forwarding useful when as many as 95% of nodes do not know their location. (b) All nodes find proxies. No packets are dropped at the source using the proxy technique, implying that every node manages to find a proxy.

5.2 Location Proxies

To evaluate our location proxy technique, we simulated networks with a varying fraction of nodes without location information. For comparison, we also simulated networks running only the basic protocol, as well as a simple neighbor protocol. In the simple neighbor protocol, a location ignorant node advertises as its own the location of any location aware node it can reach with its local routing protocol.

The simulations used 300 nodes in a square universe, 2,000 meters on a side. Up to half of each route message was dedicated to advertising proxy route entries. Simulations ran for 300 seconds of simulated time, and about 12,000 data packets were originated.

Figure 6a shows how the data packet delivery rate decreases as fewer nodes know their location. An insignificant number of packets are dropped due to congestion. As fewer nodes know their location, the location proxy technique provides an increasing advantage in delivery rates. Furthermore, this technique works well when only ten percent of the nodes know their location. This result will also vary with the network density.

To explain why the basic protocol and the simple neighbor protocol fail differently as fewer nodes know their location, Figure 6b shows the fraction of dropped packets that are dropped before they have a chance to be transmitted by the sender. With the basic routing protocol, many dropped packets never even leave the sender. A node that does not know its position can only originate a packet if the destination is in the node's route table, or if the node knows a route to some location equipped node that is within two hops *and* the destination has advertised its location. Thus as the number of nodes without location increases, the number of packets dropped at the sender grows quadratically. Since the neighbor protocol sometimes allows nodes without positions to advertise a position to the location service, more packets can be originated than in the basic protocol.

Figure 6b implies that the proxy technique finds a proxy for every location ignorant node. If we refer to the algorithm in Figure 2b, we see then that only proxies can drop packets due to bad routes; client nodes can always forward packets towards some proxy. A proxy drops a packet either because it is the destination's proxy, but has no route to that client, or because it can find no proxy geographically closer to the client's proxy. Thus all routing drops are caused by a lack of routing bandwidth: routes were not advertised quickly enough between a client and its proxy, or between two proxies. Because our protocol limits routing overhead to a fixed maximum, and because routing entries timeout at each node, some routes may be expired before they can be propagated when there are too many routes.

To verify that our technique has low overhead, Figure 7a shows the maximum local route table size over all proxies for the entire simulation. Route table size actually *decreases* as fewer nodes know their locations. This is because every node retains all routes to location equipped nodes that it receives, regardless of the DSDV hop radius. When fewer nodes know their locations, there are fewer such routes to be stored.

Given the node density we are using, and the 2 hop DSDV radius, the average routing table size for the basic routing protocol is around 60 entries, which is one half to one third the maximum size observed. However, route table

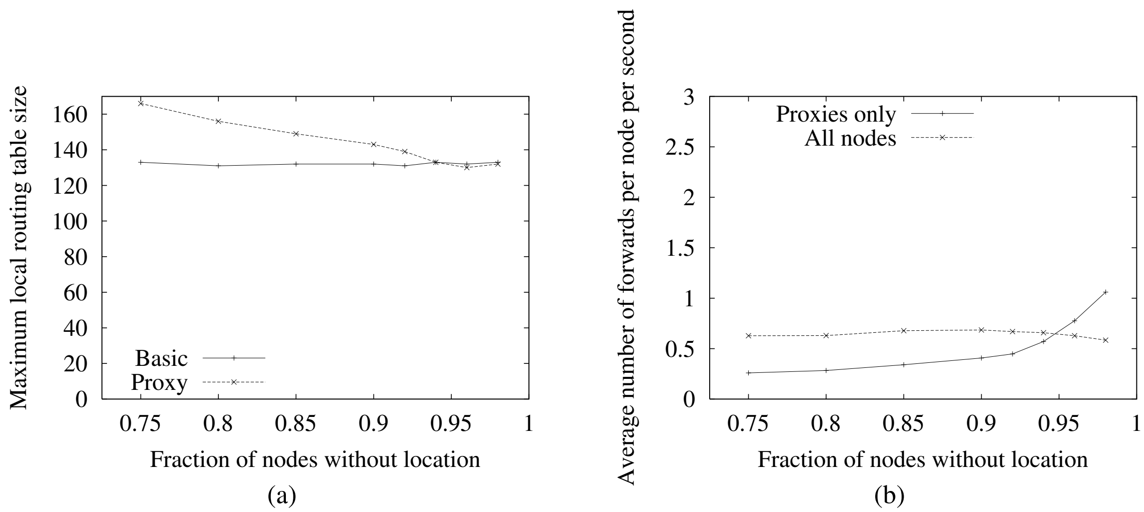


Figure 7: (a) Location proxies slightly increase the maximum route table size, which is measured across all nodes for the entire simulation. (b) Location proxies do more work and handle a larger percentage of traffic when fewer nodes know their location.

sizes will be larger for nodes near the center of the simulation, due to pathologies in the simulation—nodes tend to move to the center. The route table size does not grow to cover all the nodes because the presence of proxies in the network limits how far a client's route will propagate. Client routes will only propagate enough hops to reach the closest proxy.

By design, per-node protocol overhead is constant, since route broadcasts and triggered updates are fixed sizes and sent at fixed intervals.

Figure 7b shows that each proxy forwards more traffic as the fraction of location aware nodes decreases. This is because there are fewer proxies to handle the incoming traffic for slightly more clients. The proxy load increases reasonably. An artifact of the simulation causes proxies in general to forward less traffic than regular nodes, because proxies are never the source of any traffic in our scenarios: they never send packets on the first hop. This data ignores the inflated per node forwarding overhead caused by temporarily but rapidly looping packets. Since our network is not experiencing congestion, we can ignore the overhead of the looping packets.

These results show that when the general node density is large enough, it is quite feasible to build a geographic forwarding network where only one in ten nodes has a GPS receiver or a fixed location,

Furthermore, it is not the absolute fraction of proxy nodes that matters, but the density of them throughout space. As long as potential location proxies exist in the network at a high enough density, location ignorant nodes will only be a few hops away from a proxy. We can increase the total number of nodes in the network as much as we like, as long as the proxy machines can handle the extra traffic. In fact, if the density of proxies is high enough, we could even use the simple neighbor protocol.

The random network scenario results underestimate the utility of location proxies. In any network topology where a location ignorant node is further away from a proxy than the local DSDV hop radius, the simple neighbor protocol will not work. One example of this topology is a chain of nodes extending down a hallway or outdoors. In these cases, we *must* use the proxy technique.

The location proxy technique takes advantage of a resource tradeoff in mobile networks. We are trading proxy location sensor resources for network bandwidth. As we transform more nodes to proxies by equipping them with location sensors, the average distance between a node and its proxy decreases. Therefore, route entries can be propagated for fewer hops, consuming less network bandwidth. As we decrease the number of proxies, we must advertise route entries for more hops to keep the network connected, consuming more network bandwidth. This allows us to deploy networks with the right balance of resource consumption for a particular application or environment.

5.3 Intermediate Node Forwarding

We performed two different kinds of simulations to examine the performance of intermediate node forwarding.

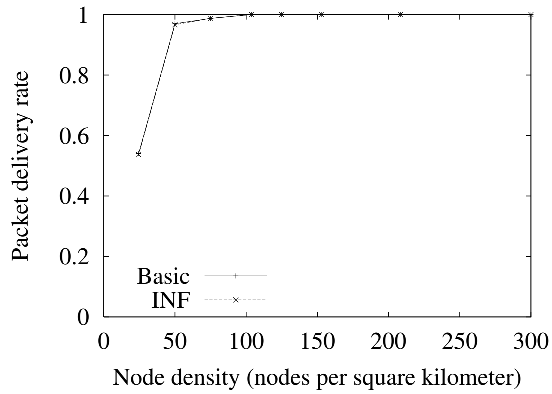


Figure 8: In uniformly random networks, INF is ineffective: it does no better than the basic routing protocol as the network becomes disconnected.

5.3.1 Random Networks

We simulated 300 nodes in networks with varying node densities, for 300 seconds; about 12,000 data packets were originated. Nodes moved using the random waypoint model. Figure 8 shows that INF provides no advantage for this network topology. The problem is that as the node density decreases, not only do holes appear, but the network becomes disconnected. Previous work [14] shows that the optimum density for geographic forwarding in a random network with similar parameters is above 75 nodes per square kilometer; this is reflected in Figure 8. In these simulations, all packet drops were caused by routing failures. Just under half of the dropped packets in each simulation never even left the sender, and local routing table sizes decreased rapidly with density, indicating that many nodes were disconnected from the network.

5.3.2 Urban Networks

Our urban network scenario is designed to model nodes moving in a modern city, with a grid of regularly placed blocks of buildings and streets. Starting in one corner, identically oriented and sized rectangles (city blocks) are laid out in rows and columns aligned with the sides of the simulation universe. The rectangles are separated in both dimensions by gaps (streets) of equal width. We chose rectangle dimensions of 200 meters by 66 meters, equivalent to a common New York City block size [1]. Street gaps within a scenario are all equal. At least two adjacent edges of the simulation universe are clear of blocks: they have street gaps laid along them. Figure 9a illustrates this scenario.

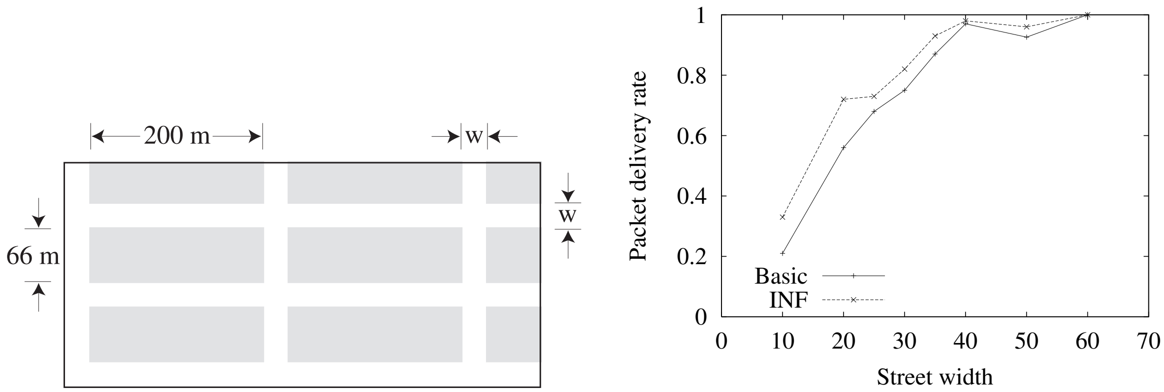
The edges of each block are completely opaque to radio signals; they block radio transmissions whose line of sight crosses the edge. If a transmitted packet's sender and receiver are on opposite sides of an edge, the packet does not arrive at the receiver. Furthermore, in our simple model, the blocked packet's transmission does not contribute to noise or interference at the receiver. In the real world, however, radio signals can squeeze through gaps, or travel around some corners. In some ways, our scenario provides a harsher environment than reality.

Nodes cannot move across block edges; they are constrained to move only in the street gaps between blocks. This scenario choice models mobile devices in the street communicating with vehicles or other externally mounted radios. A real network would have relay nodes that carried network traffic in and out of buildings; we do not model that here.

Figure 9b shows that INF provides a small but definite advantage in urban networks. The simulation universe was one square kilometer, with fixed size blocks. Simulations were carried out with 300 nodes and various street widths between the blocks, for 90 simulated seconds. Between 2,200 and 3,500 data packets were originated in each simulation; more packets could be originated when wider streets were used, and if INF was used.

We see that as streets become wider, the advantage of INF becomes less. This is slightly surprising, because the fixed size of the simulation universe causes the density of nodes in the streets to decrease as the streets become wider.

Given the simulation parameters, a street width w, and ignoring edge effects, we can calculate the actual density of nodes in streets as $\rho = \frac{(200+w)\times(66+w)}{w^2+200w+66w} \times \frac{300}{1000^2}$. For $w = 20$, this density is $\rho = 9.92 \times 10^{-4}$, or more than an order of magnitude greater than the density required to use geographic forwarding in a random network, which is 7.5×10^{-5}. Yet the delivery rate of geographic forwarding in the urban scenario is less than half of the random scenario's rate.



(a) (b)

Figure 9: Intermediate node forwarding (INF) improves packet delivery in urban scenarios. (a) shows the urban scenario used in simulation. Nodes can only move in the streets (white), avoiding the city blocks (gray rectangles). Packet transmissions cannot travel through city blocks. (b) shows INF's improvement in forwarding rate over the basic routing protocol.

The key is the strict model that we use for radio signal propagation: nodes are unable to learn of any neighbors that are around the corner of a block from them. In general, nodes can only hear neighbors who are on the same street as them, unless they are very close to the intersection. Therefore, packets can only travel around the corners of blocks if there is a node in the intersection.

The expected number of nodes n in the intersection is $E[n] = \rho w^2$. The values of $E[n]$ are very close to linear from $w = 10$ to $w = 60$, with $E[20] \approx 1$, $E[35] \approx 2.25$, and $E[60] \approx 4.5$. We can calculate the probability that *no* nodes are in the intersection. Let $a = w^2$ be the area of the intersection, and A be the total street area that can be occupied by nodes in the simulation universe. Then the probability that a given node is in a given intersection is $p = \frac{a}{A}$, assuming a uniform random distribution of nodes. We use a binomial distribution to find the probability that the intersection contains x nodes:

$$
\begin{aligned}
p(x = k) &= \binom{n}{k} p^k (1-p)^{n-k} \\
p(x = 0) &= (1-p)^n \\
&\approx 1 - pn \\
&\approx 1 - \frac{a}{A} n
\end{aligned}
$$

For a simulation in a 1000 by 1000 meter universe, with 200 by 66 meter blocks, and 20 meter wide streets, $p(x = 0) = 0.626$. That is, for a node in some street, odds are that there is no node in the intersection that can forward packets around the corner. If we consider the intersections at both ends of the street, the probability that there is no node in either is $0.626^2 = 0.392$. Although nodes may be able to send packets a few blocks down the street before the packets can turn a corner, many nodes will not be able to send packets to any destination that is not on the same street.

6 Related Work

We know of no other work that integrates location ignorant nodes into a geographic forwarding network. However, the location proxy technique is similar in spirit to the use of geographic forwarding as outlined by Finn [8]. He proposes building large metropolitan-wide networks with a regular mesh of routers. Leaf nodes in the network are connected to a router with some local network, such as a LAN or local radio network. A node sends all outbound packets to its router, which forwards them to the destination's router using geographic forwarding.

The location proxy technique uses geographic forwarding for scalable long distance packet forwarding, relying on a local routing protocol for nearby destinations. We improve the performance of geographic forwarding by adding a local protocol. Systems such as DREAM [3] and LAR [13] take the converse approach. DREAM uses location information to improve the behavior of a proactive routing protocol over long distances: it constrains the propagation of data packets through the network based on the destination's direction. LAR performs explicit route discovery using flooding route request messages, but constrains the flooding based on the destination's location.

INF's pragmatic algorithm differs sharply in spirit from previous methods for dealing with geographic forwarding holes. Karp and Kung [12] and Bose et al. [4] independently present a theoretical technique that can always find routes around holes, by sending packets around the perimeter of the hole. However, their perimeter routing algorithms make unrealistic assumptions about radio ranges and neighbor information, and neither [12] nor [4] analyze the behavior of the algorithms when the assumptions are violated, such as when there are obstructions to radio signals in the network. Karp presents simulation results for perimeter routing with a plausible network model, but does not compare the algorithm to basic geographic forwarding. Our experience suggests that basic geographic forwarding performs just as well as perimeter routing under the parameters in [12].

Since we want to build real systems, we can't make any false assumptions. We also want a robust approach; we'd prefer not to rely on (for example) all nodes having up-to-date and consistent neighbor lists, as required by [12] and [4].

7 Conclusion

Geographic forwarding is a potential technique for building scalable networks of mobile devices, but a practical geographic forwarding network faces two hurdles. It must operate despite partial location information, and it must adapt to bad network topologies. We presented two solutions which address these problems. With location proxies, location ignorant devices can participate in geographic forwarding networks, without too much overhead. The proxy technique also allows us to flexibly allocate resources in an ad hoc network. Intermediate node forwarding is a probabilistic technique which can help route packets around holes in urban networks. These two techniques make practical geographic forwarding networks a reality.

References

[1] MapQuest, 2001. `http://www.mapquest.com`.

[2] USCG Navigation Center GPS page, May 2001. `http://www.navcen.uscg.gov/gps/default.htm`.

[3] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proc. ACM/IEEE MobiCom*, pages 76–84, October 1998.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM99)*, 1999.

[5] 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 *Proc. ACM/IEEE MobiCom*, pages 85–97, October 1998.

[6] Lance Doherty. Algorithms for position and data recovery in wireless sensor networks. Master's thesis, University of California at Berkeley, May 2000.

[7] Kevin Fall and Kannan Varadhan. *ns* notes and documentation. Technical report, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997.

[8] Gregory G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. ISI/RR-87-180, ISI, March 1987.

[9] CMU Monarch Group. CMU Monarch extensions to *ns*. `http://www.monarch.cs.cmu.edu/`.

[10] A. Harter and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[11] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[12] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. ACM/IEEE MobiCom*, August 2000.

[13] Young-Bae Ko and Vaidya Nitin H. Location-Aided Routing (LAR) in mobile ad hoc networks. In *Proc. ACM/IEEE MobiCom*, pages 66–75, October 1998.

[14] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proc. ACM/IEEE MobiCom*, August 2000.

[15] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proc. ACM SIGCOMM Conference (SIGCOMM '94)*, pages 234–244, August 1993.

[16] Charles E. Perkins and Elizabeth M. Royer. Ad hoc On-demand Distance Vector (AODV) routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[17] Nissanka Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket location-support system. In *Proc. ACM/IEEE MobiCom*, August 2000.