Scalable Routing Protocol for Ad Hoc Networks *

Seung-Chul M. Woo
Motorola Inc.
Schaumburg, IL 60196

Suresh Singh
Department of Computer Science
Portland State University
Portland, OR 97201

April 2, 2001

Accepted for publication in J. Wireless Networks (WINET)

Abstract

In this paper we present a scalable routing protocol for ad hoc networks. The protocol is based on a geographic location management strategy that keeps the overhead of routing packets relatively small. Nodes are assigned home regions and all nodes within a home region know the approximate location of the registered nodes. As nodes travel, they send location update messages to their home regions and this information is used to route data packets. In this paper, we derive theoretical performance results for the protocol and prove that the control packet overhead scales linearly with node speed and as $N^{3/2}$ with increasing number of nodes. These results indicate that our protocol is well suited to relatively large ad hoc networks where nodes travel at high speed. Finally, we use simulations to validate our analytical model.

1 Introduction

Ad Hoc networks are multi-hop wireless networks consisting of radio-equipped nodes that may be stationary or mobile. These networks provide connectivity in areas where there is no other networking infrastructure available. Key applications areas for these types of networks include disaster relief, battlesite networks and sensor networks. Several networking problems need to be solved in order to deploy ad hoc networks, and in this paper we focus on one of these problems – the problem of *scalable routing in ad hoc networks.*

Routing packets in ad hoc networks is a challenge because of the *constantly changing topology* of the network triggered by node mobility. Thus, when two nodes travel apart, they may no longer have a direct link between them. Likewise, if a node moves behind a hill, its links to its neighbors may be severed because of fading. Other reasons for changes in topology include jamming and the entry or departure of nodes from the network. Several authors have proposed routing algorithms for this environment (see section 5). However, few of these proposed algorithms have been evaluated with regard to their scalability in large ad hoc networks ([4] is one of the few papers that attempts to do a detailed analysis of several protocols via simulations but scalability is not addressed).

In this paper we present a new routing protocol and derive a theoretical bound that characterizes its scalability. Our routing protocol relies on a location update mechanism that maintains approximate location

*This work was supported by the NSF under grant number NCR-9706080.

information for all nodes in a distributed fashion. As nodes move, this approximate location information is constantly updated. To maintain the location information in a decentralized way, we map node IDs to a geographic sub-region of the network. Any node present in this sub-region is then responsible for storing the current location of all the nodes mapped to this sub-region. In order to send a packet to a node, the sender first queries the destination's sub-region for the approximate location of the destination and then uses a simple geographic routing protocol to forward the packet to the destination's approximate location. It is therefore easy to see that there is a location update cost in our protocol that is dependent on the speed of node movement (high speeds result in more location update messages) and is also dependent on the number of nodes in the network. In this paper we derive a theoretical bound on the protocol's performance and validate it via extensive simulations in medium to large size networks. We show that the *routing overhead* scales as:

- $O(v)$ where v is the average node speed, and
- $O(N^{3/2})$ where N is the number of nodes in the network.

The remainder of the paper is organized as follows:

- We present an overview of our routing protocol in section 2 and a more detailed description (including all the optimizations) in section 3.
- A detailed analysis of the protocol is then presented in section 4 where we derive the *routing packet cost per data packet* for our protocol.
- Section 4.6 presents additional performance results derived from simulation.
- A brief overview of other routing protocols and a high-level discussion of their scalability is presented in section 5
- The protocol has been implemented in the NS-2 simulation environment. We discuss some of the implementation details in the Appendix.

2 Overview of SLURP (Scalable Location Update-based Routing Protocol)

In large networks or in networks where nodes travel at high speeds, the routing information for source-destination pairs tends to become stale very quickly. This happens in large networks because longer paths have a shorter time to failure (a path fails when any link on it breaks). Likewise, the time to path failure is shorter in networks where nodes travel at higher speeds. In both of these environments caches (for routing information) also have shortened lifetimes which in turn implies that finding or maintaining accurate routing information can be expensive.

Our approach for developing a scalable routing protocol is to constantly maintain *approximate* location information about nodes in the network and to only find accurate routes to specific nodes when there are packets to be sent to them, this approach is similar to the one adopted in DREAM [2]. Specifically, when a node needs to send a packet to some destination, it first determines the destination's approximate location and then uses a simple geographic routing protocol to send the packet to the destination. This two phase approach allows us to reduce the cost of maintaining routing information while, simultaneously, providing us the ability to find relatively good routes inexpensively when needed. Thus, our algorithm, SLURP, is based on a combination of *(1) approximate geographical routing* and *(2) a simple static mapping* procedure to maintain approximate location information for nodes. In the remainder of this section we first describe the algorithm used to find the geographical location of a node and then we discuss the approximate geographic routing protocol used to route the packets to the destination.

2.1 Location Management

We assume that all nodes in the system are equipped with GPS (Global Positioning System) hardware that provides them with their current location. We further assume that the ad hoc network is in a rectangular region of dimension $l \times h$. All nodes are equipped with radios (transmission range r_t) and we assume that they are aware of the identity of their neighbors (this is information that can be provided by the MAC layer). Finally, we assume that all nodes have a unique ID (such as an IP address).

Figure 1 illustrates an ad hoc network where we divide the geographical area into rectangular regions of dimension $a \times b$. All of these home regions have well-defined IDs that are concatenations of the x and y coordinates of the bottom-left and upper-right corners. Next, we assume that there exists a *static mapping* f that maps a node's ID into a specific region (called its *home region*),

$$f(\text{Node } ID) \longrightarrow \textit{Region } ID$$

f is a many-to-one mapping that is static and known to all nodes of the ad hoc network. In Figure 1, node D is associated with home region *region 7*. The function f needs to satisfy several properties including:

- f should be such that every region has the same node density. The reason for this is to evenly distribute the queries throughout the network.
- Entry/departure of nodes from the network should be transparent to f.
- f should not depend on the shape or size of the geographical area covered by the ad hoc network. For instance, the same f should work in different disaster areas.

An example of a function that satisfies these criteria is:

$$f(ID) = g(ID) \bmod k$$

where $g(ID) \longrightarrow [0 \ldots \mathcal{N}]$ is a random number generating function that uses ID as seed and outputs a random number in some large range. k denotes the number of home regions that are present in the network. We assume that every node has a table containing home region addresses (the geographical coordinates) indexed by k. It is easy to see that even if the network is made up of non-contiguous geographical areas, the function f continues to provide us with an appropriate mapping.

2.1.1 Location Tracking Algorithm

Given the above structure of the network, we use the following algorithm to maintain approximate location information about nodes in the network.

Step 1: Mobility Triggered Updates A mobile always informs the nodes currently present in its home region of its location, i.e., the identity of the region it is located in (this is conceptually similar to MobileIP [1] where a roaming user informs its home agent of its current location). This information is only updated when the node moves out of its current region and into a new one. In Figure 1, node D sends *one* message to nodes in *region 7* when it moves from *region 9* into *region 14*.

Step 2: Update in Home Region The location update message sent by a node to its home region is *broadcast* to all nodes in the home region. Thus, D's location update message will be broadcast to all nodes in *region 7*. Note that the update message is uncast until it reaches one or more nodes located in *region 7*, whereupon one of these nodes broadcasts it to all others in *region 7*.

Step 3: Locating a Node Suppose node S in Figure 1 needs to send packets to D. It needs to determine D's current location first. To do this, S uses f to find the home region of D and sends a message to this region enquiring about D's current location. In our example, S sends a query message to *region 7*. The first node in *region 7* to receive this message (say node H) responds with D's current location (i.e., *region 14*).

Once S has D's current location, it can send packets to D's current region. The geographic routing algorithm to do this is discussed in the section 2.2. It is noteworthy that this same geographic routing algorithm is also used above in Steps 1–3 to send location update/query messages.

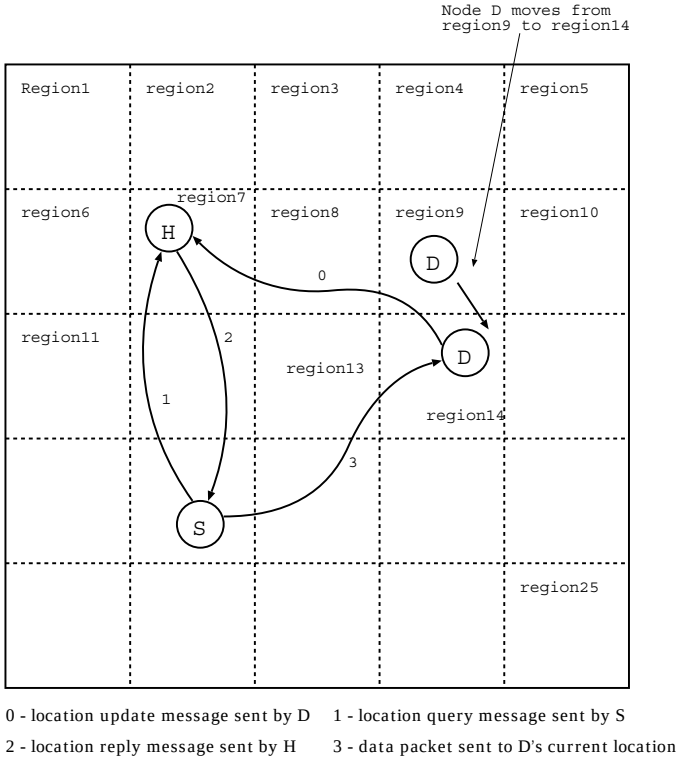


Figure 1: Location management in ad hoc networks.

2.2 Approximate Geographic Routing

The second part of SLURP deals with transmitting packets when the destination's region is known but we do not have topological routing tables. Our algorithm here is based on MFR (Most Forward with fixed Radius) [8] *without backwards progression*. To see how MFR works, consider Figure 2 where node S needs to send a packet to D. It has three neighbors, A, B and C. Of these three neighbors it selects A as the next hop because A is closest (in physical distance) to D. A then repeats the same procedure. Problems arise when a node has no forward neighbors in the same direction as the destination, etc. We address these questions in detail in section 3.

When the first data packet is sent to a destination, the source does not know the precise coordinates of the destination. It only knows the region that the destination is currently located in. Thus, to route packets using MFR, the source determines the next hop using the coordinates of the *center* of the destination's current region as the location of the destination. When the packet reaches a node in the region, if that node has a cached route to the destination, it uses that route to route the packet. Otherwise, it broadcasts a location_discovery packet (as in DSR [4]) to nodes within the same region. The destination eventually receives this packet and generates a response that contains the route. Thus, MFR is used to get the data packet to a region and then we use source routing to get the packet to the destination. Since regions tend to be very small, the performance of the source routing protocol is very good (DSR is very well-suited to small networks).

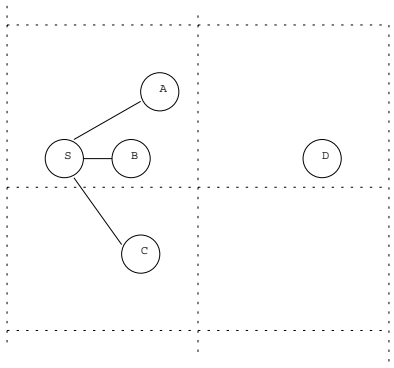


Figure 2: An example for MFR.

3 Protocol Details

In this section we look at details of SLURP. Recall that the two main components of the protocol are location management and geographic routing. Geographic routing is relatively simple as it is based on MFR and DSR. Most of the complexity (and innovativeness) of SLURP lies in the location management aspects and we describe these aspects in sections 3.1 – 3.4. In section 3.6 we describe several optimizations that were made to the protocol to enable it to run more efficiently. Before getting into the protocol details, however, we need to reiterate a key requirement of SLURP. We assume that each region has well-defined boundaries that are known to each node in the system. Furthermore, every node can determine the center of a region and its own location within a region using GPS information.

3.1 Location Discovery

Every node in the system maintains a *location cache* that contains the following information – *node ID,* (x, y) *coordinates, current region ID, best neighbor* for routing to that node. Say some node S wants to send a data packet to D. It needs to first find the current location of D. Therefore, it generates a location_discovery packet containing the destination's ID and, in addition, it contains the source's ID, current location and a sequence number. The sequence number is used to ignore older location_discovery packets. The location_discovery packet also contains a field called the *level* of discovery. This is explained in the section 3.2.

Upon reception of a location_discovery packet, each neighbor updates the source's location in its location cache (all nodes snoop packet transmissions). The neighbor that was the designated next hop, then forwards the packet again using MFR, and so on until the packet reaches some node in the destinations home region. This node generates a location_reply packet containing the destination's current region ID. We do not check the age of this information because a node may be stationary, in which case this information will have a high age but will still be correct.

3.2 Empty Home Region

It is possible that there are *no nodes in the home region* of a node. To handle this problem, we define the *level* of a node's home region as follows:

- Level 1 – this means that there is at least one reachable node in the home region.
- Level 2 – this means that the home region is empty, therefore, *all* the regions surrounding the home region are now considered the home region. In Figure 1, if *region 13* is empty, the home region for any node whose region was *region 13* now includes *regions 7, 8, 9, 12, 14, 17, 18, 19.*

The above definition can be extended to higher levels as well. However, the cost of broadcasting updates in large regions is high and we seldom need to go higher than Level 2.

In regard to the location_discovery message, the level number indicates whether the home region should be queried for the location of a node or should the extended region (Level 2) be queried in the event that the home region is empty.

3.3 Maintaining Home Regions

Nodes in a home region are also mobile, and thus the set of nodes in a region changes over time. In our protocol, all the nodes of a home region are responsible for keeping location information for nodes registered in that region. But what happens when a node moves out of a region, or when a new node enters a region? In SLURP, all nodes keep a list, node_list, of other nodes located within the same region as themselves. This list is updated as new nodes arrive or leave.

When a node moves into a new region, the node sends a home_location_request packet to its neighbors requesting location information ((x, y)coordinates and age of the location) of the nodes registered there. Any neighbor which has such location entries generates a home_location_reply to the node. To avoid a reply storm, a formula with random waiting time is used as described in optimization methods (section 3.6) later.

Similarly, when the node detected that it left the previous region, it sends a broadcast message in its previous region informing all nodes of its departure. All nodes in that region then modify their node_lists. When a node_list has just one entry, that entry represents the last node present in the region. When this node leaves the region, it generates a broadcast message in the *eight regions surrounding the empty region.* This broadcast message contains the location information for all nodes that regard the, now empty, region as their home region.

When a node enters an empty region, it collects the location information for nodes registered in that region from any node in the neighboring eight regions. This is done via a home_location_request_2 packet that is broadcast in the neighboring eight regions.

3.4 Location Update

When a node X moves out of the current region and enters to a new region, X sends a location_update packet to its home region. Each node knows its home region and a location_update packet is sent towards the center of the home region.

Any node receiving the location_update packet in the home region updates its cache and broadcasts a location_update_broadcast packet with the new location information to all nodes in the home region. Each location_update_broadcast has a list of the neighbors of the sender. Upon reception of location_update_broadcast, the neighbor compares the list with its own neighbor list and if there is at least one neighbor who is not on the received neighbor list, it re-broadcasts another location_update_broadcast with a neighbor list that is union of the received list and his own. Nodes in the region broadcast the same location_update_broadcast at most once.

3.5 Data Packet Delivery

A data packet contains the source id, source location, destination id, destination location. MFR is used to select the next hop up to the current region of the destination. After which a DSR-like protocol is used for local delivery of the packet. Intermediate nodes update their cache with location information contained in the packet. When a delivery failure happens at an intermediate node, an error message is reported back to the source, allowing the source to re-discover the location of the destination.

3.6 Optimizations

In addition to the basic protocol described in the previous sections, there are several optimization techniques to improve the performance of the basic protocol.

1. Location replies are issued by intermediate nodes if the requested entry in the location cache is recent enough. This is a trade-off between faster initial location discovery time and accurate location information. In our simulations, the cache entry is considered recent if the age of the cach entry is 10 sec or less.

2. An unsolicited location_reply packet can be issued by a node that overhears a location discovery and has recent enough location information for the requested node. This might cause a reply storm. However the reply storm is alleviated by using random waiting period that is a function of the age of the cached entry.

3. The location discovery level represents the area in which a node can return a location reply packet. The lowest level (level 1) allows nodes only in the home region of the requested node to reply. As the level increases, the area surrounding the home region gets bigger.

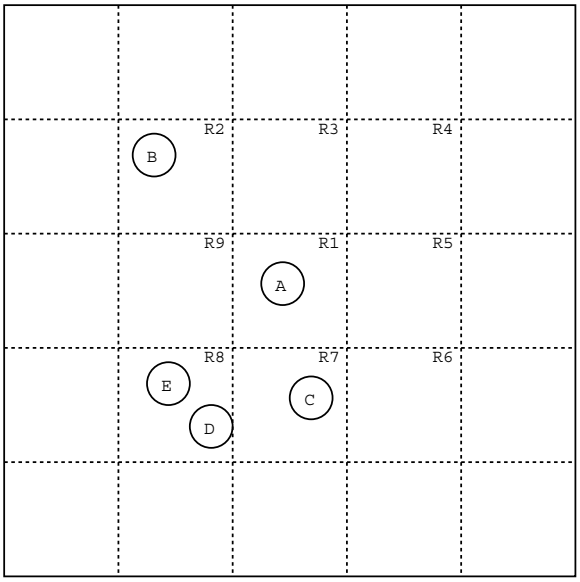


Figure 3: Location discovery level

In figure 3, suppose that the home region of the destination is R1 and the location discovery level is level 1 (the lowest). Only node A can reply back to the source. If the location discovery level is level 2, then any node (node B, C, D, and E) in any of R2–R9 can reply.

When a node wishs to find the location of a destination, the node sends out a location_discovery packet to the center of the home region of the destination, as explained previously. When the first location_discovery packet is sent out, the node sets a time interval for location reply and stores the current location discovery level. The first location_discovery packet contains the lowest location discovery level (level 1), which means that only nodes in the home region of the destination can reply. If the timer for a location reply expires, another location_discovery packet with next location discovery level is sent out. This process is repeated until the location discovery level reaches a pre-defined maximum value or a location_reply packet is returned to the node.

When a node receives a location discovery packet, it checks if it is inside the area using the location discovery level in the packet. If it is not inside the area, it relays the packet as usual. If the node is inside the area and has a recent cache entry, the node returns the location information to the source in a location_reply packet. Otherwise the node broadcasts a location_discovery_large_area packet. The format of this type of packet is the same as that of location discovery. Location_discovery_large_area

packets are broadcast only in the area that is limited by the discovery level. Upon reception of a location_discovery_large_area packet, the node repeats the process, checks to see if it is inside of the area, and has recent location information for the requested node.

Note that it is possible that multiple location replies are returned to the source. The source starts transmitting data using the first location reply while simultaneously updating the location information based on multiple replies. The newer location information is then used when previous data transmission fails.

4. If a next hop from a node X is not available because there are no neighbors in the direction of the destination, the node tries to find an alternative next hop by sending an alternative_hop_request packet to all its neighbors. Any neighbor Y that has a next hop (other than the requesting node X) to the destination replies back to X. Then X routes the packet through Y. The reply storm is reduced by snooping and dropping redundant replies as mentioned previously. X avoids the same procedure (request and reply) for the subsequent packets to the same destination in order to save resources by registering Y as a temporary next hop in its location cache (or a different data structure) until X finds his own next hop.

5. For large networks, a group of nodes like a platoon in a battlefield may move together as a unit. If a node X wants to send packets to a destination, it is a good idea to ask other nodes in the group for the location of the destination instead of directly querying to the home region of the destination. The node X broadcasts a limited hop location_discovery packet with a propagation level that starts with propagation level 1 and increases gradually to a certain maximum. Propagation of this type of packet is limited only to the nodes whose distance from the source is smaller than or equal to the propagation level in the packet. The source uses a timeout mechanism to increment the propagation level.

6. In the basic protocol, MFR is used every time to decide on the next hop. This is fine in location discovery, reply, and update because their end-to-end service happens only infrequently. However data transmission between a pair of end-to-end nodes typically consists of a sequence of packets. Because the speed of intermediate nodes is relatively slow (almost stationary) compared with the speed of data transmission, the next hop can be stored in a routing table and be used until the next hop becomes unavailable.

7. For very fast moving nodes, periodic location updates from the fast moving destination to the source is necessary. This is especially true when there is an on-going session between the source and the destination and the destination is moving fast. Note that the speed of the source and intermediate nodes does not matter much because, as long as the location of the destination is accurate, packets will be delivered correctly since intermediate hops are dynamically selected.

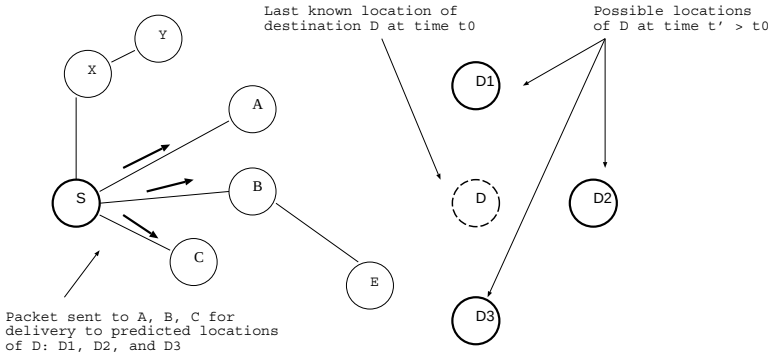


Figure 4: Selective flooding of data

In LAR [12], two methods have been proposed to reduce the size of the area where route discovery packets are propagated. A similar method can be used to deliver data packets for very high speed nodes when using small size regions. To see how this works, consider Figure 4. D is the location where the destination was at time t_0. D1, D2, and D3 are three possible farthest locations of D computed by source S using (current time $t' - t_0$) * (speed of D at t_0). Any node that satisfies the condition of MFR for at least one of the three points relays the data packet. In Figure 4, node S selects A, B, and C as the set of next hops because A, B, and C satisfy the condition of MFR for D1, D2, and D3 respectively.

8. One assumption made in our protocol is that nodes do not move out of the area defined by the regions. This is justified in many instances (e.g., disaster recovery where the entire disaster hit region forms the geographic extent of the ad hoc network) but there are cases where the ad hoc network may drift into new geographic areas. In these cases, the region boundaries need to be redefined and updates need to be sent to all the nodes. We expect this function to be performed by a management layer on top of the routing layer. This management layer needs to perform the following tasks:

   - Periodically obtain location updates of all nodes and use this information to determine if the network's geographic spread is changing.
   - Recompute region boundaries (and perhaps create new regions).
   - Inform all nodes of the new region boundaries.

   We do not address this management function here because it can be easily added to existing network management protocols. We have defined and implemented an Ad hoc Network Management Protocol (ANMP) [6] in which the network manager does collect geographic information about nodes. The functions specified above can be added to ANMP in a simple way.

4 Scalability Analysis of SLURP

Recall from our discussion in the introduction that scalability of a protocol is best characterized by the control message overhead required to route data packets between nodes. In this section we develop a theoretical model to describe SLURP's scalability. Specifically, we show that,

- SLURP scales *linearly* with increasing node speeds ($O(v)$), and
- SLURP scales as $O(N\sqrt{N})$ with increasing numbers of nodes.

As a first step in the analysis, it is necessary to identify the three major components of the protocol that contribute to message cost:

1. The cost of maintaining a node's location as it moves – we call this the *location update cost*,
2. The cost of maintaining home regions. As nodes move into new regions, they need to inform nodes of the previous region that they have exited and collect the location information of the various nodes registered in the new region – we call this the *location maintenance* cost, and,
3. The cost of locating a node when a data packet needs to be sent to it – the *location finding* cost.

It is important to mention that our analysis ignores the savings obtained by the use of caches in the system. In other words, our model provides an *upperbound* for the total cost.

The remainder of this section is organized as follows. In the next subsection we provide a glossary of notation used in the remainder of this paper. In addition, this section also describes a simulation model that we use throughout our derivation to illustrate as well as to validate our model. Section 4.2 presents our analysis for the location update cost, section 4.3 presents the location maintenance cost and section 4.5 puts all the pieces together and provides the total cost of using SLURP.

4.1 Notation and Definitions

As the number of nodes in the network grows, we expect the cost of routing to also increase. However, the cost of routing depends a great deal on how we model increasing network sizes. For instance, if the number of nodes increases but the area covered by the network remains the same (this is also called the playground size), the average degree of a node increases and the diameter of the network shrinks (to a value approximately equal to the ratio of the playground size and the transmission radius). This will automatically *reduce* the cost of routing because of two reasons: caching will be more efficient since the network has a higher density, and the number of hops reaches a bound. We believe that to properly model increasing network sizes, the *playground size must also increase* with an increase in the number of nodes N so that the *average degree is kept constant*. This model better reflects our notion of scale and provides a more meaningful measure of a protocol's performance.

4.1.1 Simulation Example

In order to illustrate and validate our theoretical model, we implemented a detailed simulation of SLURP and measured its performance. The model that we used had the following characteristics:

- The number of nodes was varied from 50, 100, 300, 500 and 1000.
- The playground size was selected so that the average node degree remained approximately the same (~ 6.5) for all cases. Thus, for 100 nodes the playground size was 1056x1056 meters while for 1000 nodes it was 3168x3168 meters.
- The transmission radius of each node was 250 meters.
- The velocity of the nodes varied from 10, 15, 20, 25 meters/second.
- Nodes moved randomly as follows – each node selects a random direction to move in (uniformly distributed between $[0, 2\pi]$) and a random distance for which it moves in the selected direction (exponentially distributed with mean 333 meters). When it reaches the end of this distance, it selects another direction and distance (unlike [4], nodes do not pause).
- The data model we chose was the following: packets arrive into the network according to a poisson process (we used 40 packets/sec and 240 packets/sec system-wide to represent low and high load conditions). For each packet we select a random source and destination. Note that this model differs significantly from the packet-train model used in many other papers (see, for instance, [4]). We believe that our model represents a *worst-case* scenario for any routing protocol because the cache entries (for routes or location information) will have a greater probability of being invalid resulting in a larger routing packet cost.
- The region size was a square 264x264 meters.

The simulations run 20 times and we computed 95% confidence intervals. The metrics we measured included the following:

- Number of routing packets per data packet – this metric is interesting in that it highlights the benefits and limitations of on-demand routing protocols. Thus, a low data load will result in a higher number of routing packets per data packet as compared to the case when there is a high data load. This is because, in the case of a low data load, most cache entries will be stale resulting in a higher cost of finding routes.
- Location update cost – this is a speed related cost of our protocol and exists even when no data packets need to be sent.

- Percentage of packets lost due to buffer overflow or misrouting (recall that packets not finding a route because of our use of MFR may get lost).
- Average packet delay from the time the packet arrives into the network until the time it reaches its destination.

4.1.2 Notation

We use the following notation in this section:

N	Number of nodes	v	Speed (m/s)
r_t	Transmission radius	A	Area of playground
a	Area of region	γ	Average number of nodes in a region
G	Number of regions (N/γ)	$2R \times 2R$	Region size
b	Broadcast cost in a region	u	Number of hops to a region
ρ	Number of region crossings/sec/node	$\bar{n}$	Number of nodes within transmission range

We need to note that for our analysis we sometimes approximate square regions ($2R \times 2R$) by circles of diameter $2R$. This is done to make the expressions more manageable and, as the simulations show, this approximation is reasonable.

4.2 Location Update Cost c_u *per node per sec*

When a node moves into a new region, it generates a location update message that is sent to its home region. Upon arriving at the node's home region, the location update message is broadcast to all other nodes within that region. The cost of performing this update is $b + u$ where b is the cost of one broadcast within a home region and u is the cost of sending the location upate message to the home region (the unicast cost). If the node crosses region boundaries at a rate of ρ per second, the cost of location update can be written as,

$$c_u = \rho(b + u) \text{ packets/sec/node} \tag{1}$$

We can estimate ρ quite easily by assuming a circular region of radius R meters (in our simulations we used square regions but we use circular regions in this analysis for simplicity – the value for ρ is within 5%). As illustrated in Figure 5(a), once the node enters a region, it travels for some time within the region before exiting. The distance the node travels while in the region can be seen to be $2Rcos(\theta)$ where θ is the angle the velocity vector makes with the tangent at the point of entry into the region. Thus, the *average* distance traveled by a node within a region is,

$$\frac{2}{\pi}\int_0^{\pi/2} 2Rcos(\theta)d\theta = \frac{4R}{\pi}$$

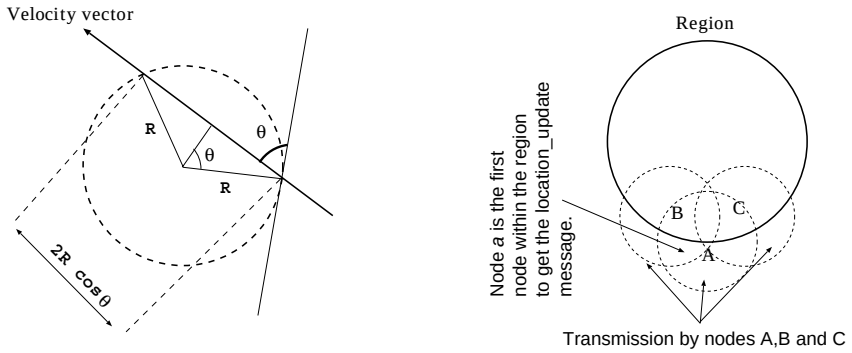
Therefore we get,

$$\rho = \frac{v}{4R/\pi} = \frac{\pi v}{4R} \tag{2}$$

In order to estimate the broadcast cost b, consider Figure 5(b). Let us assume that A is the first node within the region to receive the location update message. A now initiates a broadcast to disseminate the information within the region. A's broadcast is forwarded by nodes B and C, and so on. The approximation for the broadcast cost b is thus the number of transmissions needed to cover the region with circles of radius R plus one transmission for the first node A. Thus,

$$b = 1 + \left\lfloor \frac{a}{\pi r_t^2} \right\rfloor \tag{3}$$

where a is the area of the region and r_t is the transmission radius of a node.



(a) Distance travelled by a node within a region.

(b) Estimate for the broadcast cost b within a region

Figure 5: Estimates for ρ and b.

Finally, we can estimate u, the number of hops between the node generating the location update message and its home region, as,

$$u = d/z$$

where d is the physical distance between the node generating the location update message and the center of its home region. z denotes the average *forward progress* made towards the destination in the course of one transmission. Recall that we are using MFR [8] as our routing protocol of choice. Therefore we can rely on a previous result from [8] to determine z.

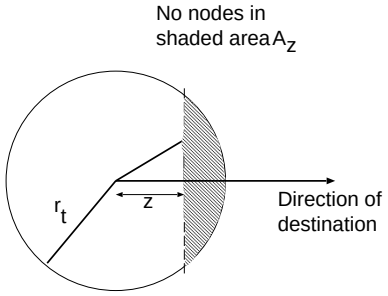


Figure 6: Distance travelled in one transmission.

Figure 6 shows the actual physical distance (z) covered by one transmission. The circle represents the transmission range of the node located at the center of the circle. The arrow indicates the direction in which the destination is located. In this diagram, the node closest to the destination is distance z away from the center and the region A_z is empty. To determine z, let us assume that nodes are placed in the playground according to a two dimensional Poisson process with mean density λ nodes per unit area. The average number of nodes within transmission range is therefore,

$$\bar{n} = \lambda \pi r_t^2$$

From [8] we can then write the pdf (probability density function) for z as,

$$f_z(z) = \frac{2\lambda\sqrt{r_t^2 - z^2}}{1 - e^{-\bar{n}/2}} e^{-\lambda A_z}$$

where the excluded region A_z is,

$$A_z = r_t^2 (cos^{-1}(z/r_t) - (z/r_t)\sqrt{1 - (z/r_t)^2})$$

Thus, the average progress made towards the destination in one hop is,

$$\bar{z} = \int_0^{r_t} z f_z dz$$

Unfortunately we cannot solve for $\bar{z}$ explicitly but numerical integration techniques can be used to solve it. Figure 7 plots the value of $\bar{z}$ versus $\bar{n}$ for the case when the transmission radius is 250 meters. In our example $\bar{n}$ is approximately 6.5 and therefore the average progress made towards the destination in one hop is approximately 130 meters.

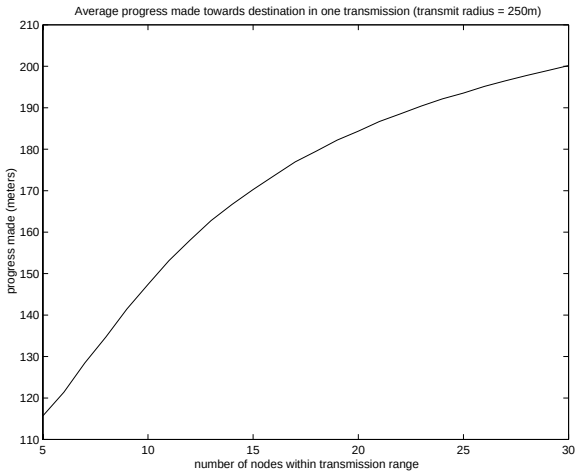


Figure 7: Average progress made towards the destination in one transmission.

To determine u we also need to find the average distance between a node and its home region. If we assume that the playground is shaped like a torus, we can determine $\bar{d}$ as follows. Let us assume that each region is a square (as in our simulations) and the length of a side of the region is $2R$. The home region of the node can be any of the G regions in the network with equal probability. Thus, we have,

$$\bar{d} = \frac{1}{G} \sum_{d_i=0}^{D} n_i d_i$$

where D is the maximum distance to any region and n_i is the number of regions at distance d_i. In Figure 8 we assume that our node is located in the shaded region. We indicate the physical distance between this node and each region by the values shown in the figure. For instance there are four regions distance R away and four that are distance $\sqrt{2}R$ away. Likewise there are eight regions distance $\sqrt{10}R$ away, and so on. Using this, we can now write,

$$\begin{aligned}\bar{d} = \; & \tfrac{1}{G}\left(4R(1+\sqrt{2}) + 4R(3+2\sqrt{10}+3\sqrt{2}) + 4R(5+2\sqrt{26}+2\sqrt{34}+5\sqrt{2}) + \right. \\ & \left. \cdots + 4R(k+2\sqrt{k^2+1^2}+2\sqrt{k^2+3^2}+\cdots+2\sqrt{k^2+(k-2)^2}+\sqrt{k^2+k^2})\right) + \tfrac{1}{G}(G-k^2)k\sqrt{2}R \end{aligned} \quad (4)$$

where,

$$k = \begin{cases} \lfloor\sqrt{G}\rfloor, & \text{if } \lfloor\sqrt{G}\rfloor \text{ is odd} \\ \lfloor\sqrt{G}\rfloor - 1, & \text{if } \lfloor\sqrt{G}\rfloor \text{ is even} \end{cases}$$

The last term in the expression for $\bar{d}$ is an overestimate of the distance to the remaining regions.

For a specific system we could quite easily simplify the above sum and determine the value of $\bar{d}$. For our purposes, however, it suffices to use bounds. We can write $\bar{d}$ as,

$$\tilde{d} \leq \bar{d} \leq \sqrt{2}\tilde{d}$$

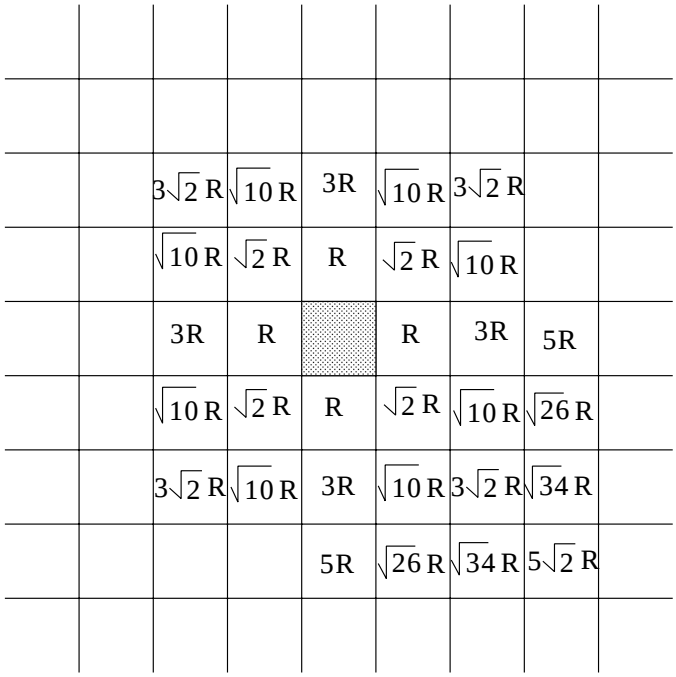


Figure 8: Average distance to a node's home region.

where,

$$\tilde{d} = \frac{R}{3G}\left(k^3 - 12k^2 + (8+3G)k\right)$$

This bound is obtained by setting the sum of the series (some l),

$$(l + 2\sqrt{l^2+1^2} + 2\sqrt{l^2+3^2} + \ldots + 2\sqrt{l^2+(l-2)^2} + \sqrt{l^2+l^2})$$

to $l(l+1)$ for the lower bound and $\sqrt{2}l(l+1)$ for the upper bound. Note that each term in the expression for $\bar{d}$ (equation 4) is in the form of the above series.

Theorem 1 *The average location update cost c_u is $\propto v\sqrt{N}$.*

Proof 1 *The average location update cost per node per second is,*

$$\bar{c}_u = \bar{\rho}(\bar{b} + \bar{u})$$

Substituting values for $\bar{\rho}, \bar{b}$, and $\bar{u}$, we get,

$$\bar{c}_u = \frac{\pi v}{4R}\left(1 + \left\lfloor \frac{a}{\pi r_t^2} \right\rfloor + \frac{\bar{d}}{\bar{z}}\right)$$

Simplifying,

$$\bar{c}_u \propto \frac{v}{R}(R^2 + \frac{RG\sqrt{G}}{G})$$

where $\bar{d}$ is proportional to $\frac{RG\sqrt{G}}{G}$ and $\bar{z}$ is a constant (proportional to the transmission radius). Recall that $G = N/\gamma$, therefore eliminating terms we get,

$$\bar{c}_u \propto v\sqrt{N}$$

Experimental Validation: In Figure 9 we plot the number of location update messages sent per node per second versus speed for two representative cases – $N = 100$ and $N = 1000$. We also plot the theoretical values for comparison. As we can see, the theoretical values mimic the measured cost but overestimate if because of the various simplifying assumptions we made.

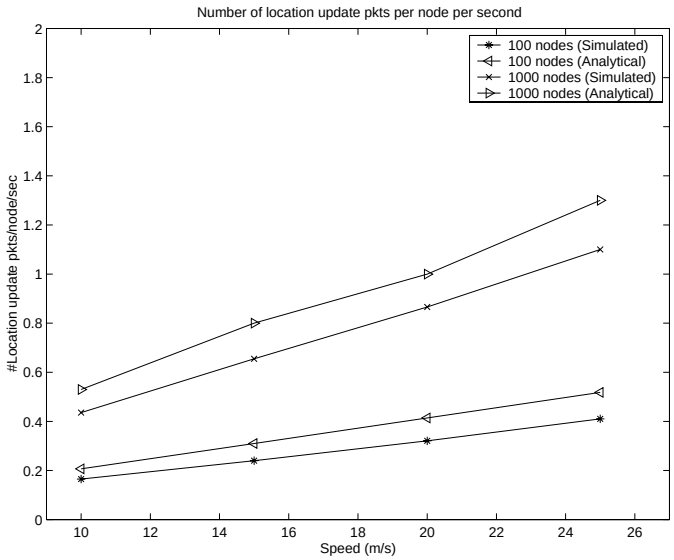


Figure 9: Location update cost – theoretical and experimental values.

4.3 Cost of Maintaining Home Regions c_m *packets/second*

When a node moves from region A into region B, it needs to inform nodes in region A that it has left and, simultaneously, it needs to inform nodes in region B of its arrival. It also needs to collect location information about all the nodes registered in region B. The message cost of performing this activity is denoted by c_m and in our analysis below we use the units *packets/sec* to measure this cost. We can write,

$$\begin{aligned} c_m = \quad & \text{number of nodes crossing boundaries per second} \times \\ & (\text{cost of broadcasting the departure/arrival in the old/new region} + \\ & \text{cost of collecting location information}) \end{aligned} \tag{5}$$

Thus, we can write the mean cost of maintaining home regions as,

$$\bar{c}_m = (N\bar{\rho})(2\bar{b} + \delta)$$

where $\bar{\rho} = \pi v/4R$ is the number of boundary crossings per second per node, $\bar{b}$ is the broadcast cost in a home region and δ is a constant. We can estimate δ as the cost of collecting location information about nodes registered in the new region. In the worst case, the new node would get as many as γ duplicate responses (recall that γ is the average number of nodes within a region) to its request for this information. In the best case, only one node would respond. Thus,

$$1 \leq \delta \leq \gamma$$

Using the above formulation we have,

Theorem 2 *The cost of maintaining home regions c_m is $\propto vN$.*

Proof 2 *We can simplify the expression for c_m as,*

$$c_m = \frac{N\pi v}{4R}\left(2(1+\left\lfloor\frac{a}{\pi r_t^2}\right\rfloor)+\delta\right)$$

thus,

$$c_m \propto Nv$$

4.4 Cost of locating a node c_l *packets*

When a node receives a data packet for some destination, it needs to find the current location of the destination before sending the packet to it (using MFR). The cost of finding the location might be zero or a constant if either the node itself or one of its neighbors has the location information available in a cache (this happens if there are several packets destined for the same destination). In our analysis, however, we will make the pessimistic assumption that the location information is not cached and therefore the source node needs to contact the destination node's home region to find the destination's location. This cost as,

$$c_l = 2u \tag{6}$$

where $u = d/z$ is the unicast cost of sending the location request message to the destination's home region. On the average therefore,

$$\bar{c}_l = 2\frac{\bar{d}}{\bar{z}} \propto \sqrt{N}$$

4.5 Putting it all together – c routing *packets per second*

Now that we have estimated each of the individual costs involved in maintaining location information and finding the location information, we can estimate the total cost of routing packets in SLURP. Let us assume that packets arrive at *each* node at a rate of λ packets/sec according to a Poisson process. Then we can write the average cost of routing *per second* is,

$$\bar{c} = \text{Number of data packets per second} \times \bar{c}_l + \bar{c}_m + \text{Number of nodes} \times \bar{c}_u = N\lambda\bar{c}_l + \bar{c}_m + N\bar{c}_u \tag{7}$$

We have the following theorem:

Theorem 3 *The cost of routing packets per second in SLURP is $\propto v$ and $\propto N\sqrt{N}$.*

Proof 3 *Simplifying the above equation we obtain,*

$$\bar{c} \propto N\sqrt{N} + Nv + Nv\sqrt{N} \propto vN\sqrt{N}$$

Experimental Validation: In Figure 10 we plot the total cost of routing as a function of speed for $N = 100, 300, 500$. The dotted line indicates the theoretical prediction while the solid line shows the experimentally measured values. In the case $N = 500$ the 95% confidence intervals were quite large[1] but were less than 5% of the point values in all other cases (and hence we have not drawn them). It is easy to see that our theoretical estimates match up quite well against the simulated results. This is an important fact because simulating larger networks is a very time consuming task and rather than simulations we can rely on the analytical model for SLURP to predict how SLURP scales. Figure 11 is a 3D plot of the number of routing packets per data packet (theoretical) as a function of the number of nodes N and the speed v. The number of nodes varies from 50 to 3000 nodes and the speed varies from 0 meters/sec to 30 meters/sec. It is intersting to observe that for a given N the number of routing packets per data packet is a linear function of node speed while for a given v, this quantity scales as $N\sqrt{N}$ (see the projection onto the two far surfaces).

[1] This was because with 500 nodes we see a much higher variance in the data values. This highlights the need for analytical models because simulating large networks to get statistically significant values is very time consuming.

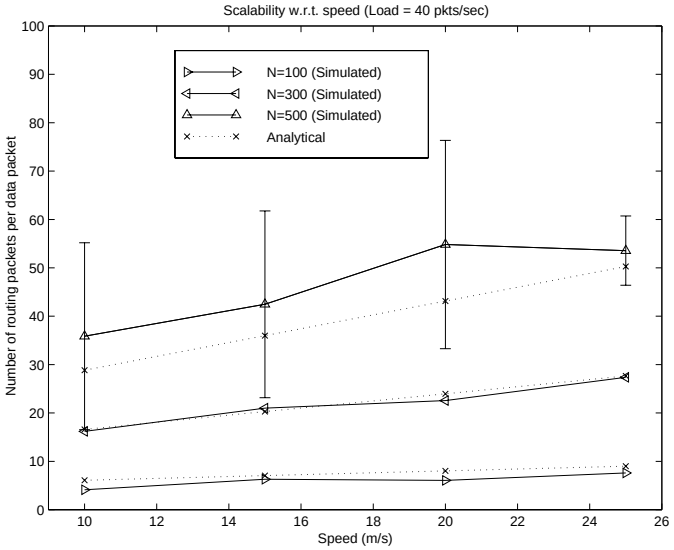


Figure 10: Overall routing overhead for different v and N.

4.6 Other Performance Results

In addition to collecting information about routing packet overhead, we also measured the loss and delay behavior of SLURP. Specifically, we measured the *percentage of lost packets* and the *average end-to-end delay* experienced by packets between the time they enter the network and the time they leave. We conducted the experiments for two load scenarios – 40 packets/sec and 240 packets/sec (system-wide).

Figure 12 plots the packet loss rate versus speed for the high load and the low load case respectively. We observe that the packet loss rates are very low in both cases regardless of N, the number of nodes in the network. The reason is that packets are not buffered for long at the source. The source can determine the current region of the destination in $2\sqrt{N}$ amount of time and begin sending the packet using MFR. This behavior of SLURP differs from protocols such as DSR that rely on a global broadcast to find a route to the destination – all the while the packet is buffered at the source which may cause packet loss due to buffer overflow.

Figure 13 plots the average end-to-end delay experienced by packets for the high and low load cases. The unit used is timesteps where one timestep is $1/2730$ seconds and represents the time to transmit one packet over one hop. First, it is interesting to observe that the delays are almost constant (for a fixed N) even as node speed increases. This is because the location of a node is constantly updated via location_update messages sent by the moving node and therefore changes in the topology have little effect on the delay. This is unlike demand based protocols like DSR where source routes become stale quickly with increased speeds. Another interesting observation is that the average delay (for a fixed N) is *lower for the high load case*. The reason for this is that at high loads, there are more packets being sent to any destination. Thus, the location information (i.e., the current region) of a destination is likely to be cached at several nodes in the network. Thus, location_discovery packets evoke a response from intermediate nodes causing a reduction in end-to-end delay for data packets.

4.7 Summary

From the above discussion it is clear that:

- SLURP scales as $O(vN\sqrt{N})$ with speed and network size.
- The percentage of lost packets is very small.

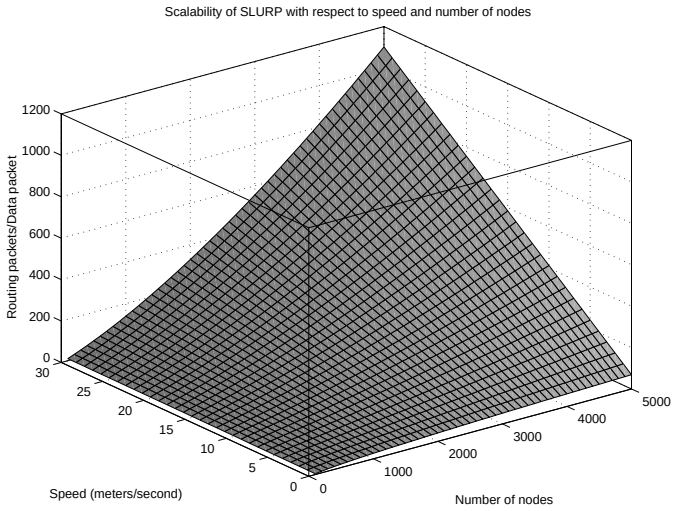


Figure 11: Theoretical performance of SLURP as a function of N and v.

- The end-to-end packet delay is independent of node speed.

Thus, we believe that SLURP is a scalable protocol that can be used in large-scale ad hoc networks. It would be interesting to develop some notion of a lower bound for routing in ad hoc networks as this would enable us to make more meaningful statements about scalability. However, this is beyond the scope of the present work.

5 Literature review

A large number of protocols have been proposed for routing in ad hoc networks (there were over 30 protocols at last count but we have only selected a representative few due to space restrictions). Unfortunately, however, there has been few, if any, attempts at developing analytical models of performance for these protocols. This makes it difficult for us to compare SLURP against these competing protocols for *large* networks (it is impractical to implement and simulate these various protocols in large networks because the total simulation time for such an effort would quite easily run into years). In this section, therefore, we make a *simplistic* attempt at characterizing the scalability of these protocols to provide us with some idea of how they would perform in larger networks. As a first step we classify the available protocols into five categories and then make general statements about the scalability of each category.

Non-hierarchical Minimum Cost Routing Protocols: These protocols maintain routing tables via periodic exchange of routing information. Examples include Destination-Sequenced Distance Vector (DSDV [16]), Wireless Routing Protocol (WRP [14]) and Global State Routing (GSR [5]).

Non-hierarchical On-Demand Protocols These protocols typically find routes only when needed, unlike the protocols discussed above. Some examples of these type of protocols are, Dynamic Source Routing (DSR [4]), Temporally-Ordered Routing (TORA [15]), and Ad Hoc on-demand Distance Vector (AODV [17]). In DSR, the source of a packet floods a route discovery packet. The destination (or some intermediate node with a cached route) responds with the complete source route to use. AODV is a hybrid of DSR and DSDV.

Non-hierarchical Protocols for Large Networks The Zone Routing Protocol (ZRP [7]) is designed for large networks. In ZRP each node has a zone that is defined as all nodes that are within h hops of the node. Routing within a zone uses a protocol such as DSR. For sending a packet to a node outside the zone, the source sends a route request packet to nodes at the periphery of its zone. These nodes query their zones or their peripheral nodes, etc.

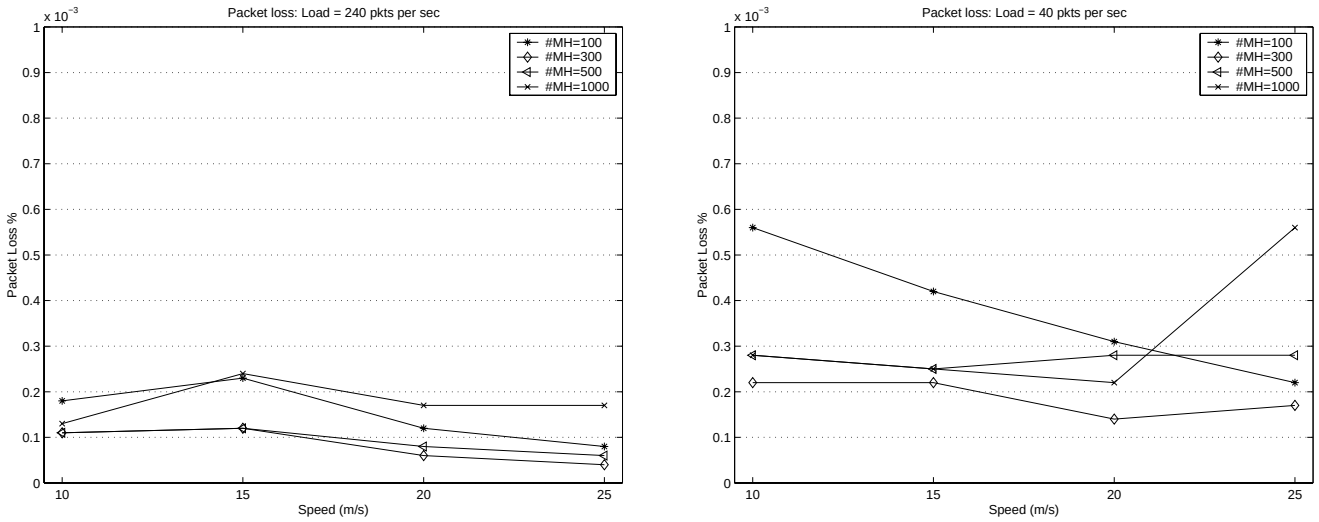


Figure 12: Packet loss for high and low load cases.

Hierarchical Routing Protocols These protocols rely on the construction and maintenance of a hierarchy in the ad hoc network. One set of protocols use a clustering algorithm at the lowest level. Communication between nodes from two different clusters takes place via the clusterheads. Examples of some protocols in this category include SPINE, Partial Knowledge Spine Routing (PSR [19]), and Cluster Based Routing protocol (CBRP [10]).

Location-based Protocols Another category of protocols that recently appeared in the literature is routing protocols based on the geographic location of devices. Such protocols include Distance Routing Effect Algorithm for Mobility (DREAM) [2] and Location-Aided Routing (LAR) [12]. A recent location service protocol [13] uses a mechanism similar to ours for maintaining the current location of nodes. Here, every node sends updates of its location to a set of location servers. Likewise, location queries are also sent to these location servers. [11] is another location based routing protocol that is very similar to MFR in its simplest form but has an interesting mechanism to recover from cases when there is no neighbor in the direction of the destination. In SLURP we use a simple backtracking mechanism to recover from these routing failures but [11] uses a more sophisticated graph traversal mechanism that reduces the probability of lost packets. Finally, the graph-based mechanism used in [11] is similar to the work in [3].

Recently a new class of geographic routing protocols for sensor networks have been developed, see for example [9]. In these protocols, sensors collect information and collaboratively diffuse it through the network. This routing (or diffusion) is based on the application's goals (e.g., sense a moving vehicle). Thus, unlike the model of ad hoc networks discussed earlier, sensor networks are very application specific and thus do not require the traditional networking machinery.

5.1 Scalability Issues

To get some sense of the scalability of these various protocols let us make the following assumptions: the number of nodes is N, speed of nodes is v, the average node degree is γ, data packets arrive at each node at a rate of λ packet trains (of size m packets – m is a constant) per second and the destination is randomly selected for each packet train.

Non-hierarchical minimum cost protocols perform table exchanges whenever the topology of the network changes. Furthermore, these table exchanges continue until the routing information becomes consistent network-wide. A crude estimate of the cost of doing this can be determined as follows. There will be of the order of $\log_\gamma N$ table exchanges needed (equal to the network diameter) for the routing information to become

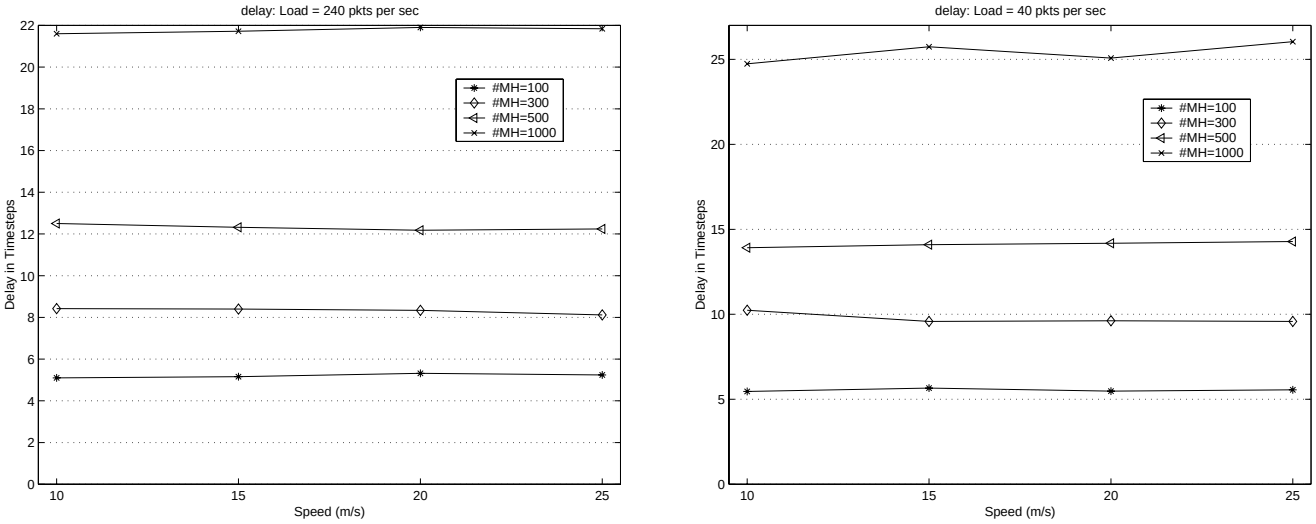


Figure 13: Packet delay for high and low load cases.

consistent; the size of each table is proportional to N; a given node breaks a link at a rate proportional to v per second (in other words, with N nodes in the network, over vN links break/form per second). Putting all these together, the message overhead per second is *at least* $O(vN^3 \log N)$. As some papers have reported, however, a bigger problem with these protocols is that of stability – the information never converges because the topology changes at a rate faster than the protocol can converge.

On-demand protocols (like DSR, AODV) that determine routing information as and when needed have significantly better performance in large networks. These protocols rely for performance improvements on the use of sophisticated caching mechanisms (that rely on packet trains for performance gains). When a packet arrives, let us suppose that the probability of finding a cached entry for the route is p_c. Then the number of packets that will not have cached entries is $N\lambda(1 - p_c)$ per second (even if the first packet of a packet train fails to find a cached entry, the remaining $(m - 1)$ will be able to use the cache). Each of these packets will cause a broadcast to determine the route to the destination (network-wide or restricted to some portion of the network – in either case the number of messages is $O(N)$). Thus, the cost of routing is at least $(1 - p_c)N^2\lambda$ per second. The value of p_c depends on N and v and is difficult to compute. However, in larger networks and in networks with fast moving nodes, p_u is actually smaller because paths fail sooner. This same analysis also applies to protocols that use geographic information to restrict the scope of their broadcast (like DREAM).

Hierarchical routing protocols depend on the maintenance of a hierarchy to perform routing. If the hierarchy is of depth 2 with a average cluster size c nodes, then a weak lower bound for the routing cost is $vN^2 \log(N/c)$ where the log term comes about for the need to maintain inter-cluster routing tables (as in the non-hierarchical minimum cost protocols), $O(Nv)$ is an estimate for the number of boundary crossings per second and the final N term is the amount of routing information exchanged at the top-level of the hierarchy to maintain inter-cluster routing tables.

Finally, while ZRP purports to be designed for large networks, its performance does not scale well because of the way in which routes are found. When a source has a packet for a non-local destination, it sends the packet to its peripheral nodes. These nodes, in turn, forward the request to their own peripheral nodes, and so on. In other words, the route discovery process is at least as bad as flooding (a node may be the peripheral node for more than one node and will receive the same route inquiry many times). Some optimizations have been proposed for ZRP but its is unclear if the cost of the protocol reduces significantly.

We would like to reiterate that the above analysis is crude at best and was only included to get a sense of how competing protocols perform. Clearly all of these protocols would benefit from a more detailed analysis

and that constitutes a major portion of the current work for the authors [18].

6 Conclusions

In this paper we presented a new routing protocol for large networks and developed a theoretical model for predicting its scalability with respect to increased node speeds as well as increasing network sizes. We showed that the routing packet overhead scales linearly w.r.t. speed and as $N^{3/2}$ w.r.t. the number of nodes in the network. The specific properties that make SLURP so well-behaved is the use of location tracking to maintain approximate location information for nodes in the network. Our future work will be aimed at determining a lower bound for routing overhead and generalizing the formalism to enable us to analyze other routing protocols as well.

References

[1] See http://www.ietf.org/html.charters/mobileip-charter.html, March 08, 2001.

[2] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. WoodWard, "A Distance Routing Effect Algorithm for Mobility (DREAM)", *Proceedings 4th Annual Conference on Mobile Computing and Networking (Mobicom 1998)*, Dallas, TX, October 25 – 30, 1998.

[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Wireless Ad Hoc Networks", *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM'99)*, August 1999.

[4] 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", *Proceedings 4th Annual Conference on Mobile Computing and Networking (Mobicom 1998)*, Dallas, TX, October 25 – 30, 1998.

[5] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks", *Proceedings of IEEE International Communications Conference (ICC'98).*

[6] W. Chen, N. Jain and S. Singh, "ANMP: Ad hoc network Network Management Protocol", *IEEE J-SAC Vol. 17(8)*, August 1999, pp. 1506 – 1531.

[7] Zygmunt Haas and Marc Perlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", draft-ietf-manet-zone-zrp-01.txt, August 1998.

[8] Tingh-Chao Hou and Victor O.K. Li, "Transmission Range Control in Multihop Packet Radio Networks", *IEEE Transactions on Communications, Vol. COM-34, No. 1,* January 1986.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proceedings 6th Annual Conference on Mobile Computing and Networking (Mobicom 2000)*, Boston, MA, August 6 – 11, 2000.

[10] Mingliang Jian, Jinyang Li, Yong Chiang Tay, "Cluster Based Routing Protocol (CBRP) Functional Specification", Internet-Draft, draft-ietf-manet-cbrp-spec-00.txt

[11] B. Karp and H. T. Kung, GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", *Proceedings 6th Annual Conference on Mobile Computing and Networking (Mobicom 2000)*, Boston, MA, August 6 – 11, 2000.

[12] Young-Bae Ko and Nitin H. Vaidya,"Location-Aided Routing (LAR) in Mobile Ad Hoc Networks", *Proceedings 4th Annual Conference on Mobile Computing and Networking (Mobicom 1998)*, Dallas, TX, October 25 – 30, 1998.

[13] J. Li, J. Jannotti, D. S. J. DeCouto, D. R. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing", *Proceedings 6th Annual Conference on Mobile Computing and Networking (Mobicom 2000)*, Boston, MA, August 6 – 11, 2000.

[14] S. Murthy and J.J. Garcia-Luna-Aceves,"An Efficient Routing Protocol for Wireless Networks", *ACM Mobile Networks and Applications Journal*, Special Issue on Routing in Mobile Communications Networks, 1996.

[15] V. D. Park and M. S. Corson,"A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", *Proc. IEEE INFOCOM'97*, Kobe, Japan, (1997).

[16] Charles E. Perkins and Pravin Bhagwat, "Routing over multi-hop wireless network of mobile computers", In Tomasz Imielinski and Henry F. Korth, editors, Mobile Computing, pages 183–205. Kluwer Academic Publishing, 1996.

[17] Charles Perkins, "Ad Hoc On Demand Distance Vector (AODV) routing", Internet-Draft, draft-ietf-manet-aodv-00.txt, November 1997.

[18] S. Singh, "A Mathematical Toolbox for Analyzing the Performance of Routing Protocols in Ad Hoc Networks", Manuscript.

[19] Raghupathy Sivakumar, Bevan Das, Vaduvur Bharghavan, "The Clade Vertebrata: Spines and Routing in Ad Hoc Networks",*Proc. IEEE Symposium on computers and Communications,* 1998.

Appendix A *Implementation Notes*

SLURP has been implemented in the NS-2 simulation environment and the source code is available from the second author's web site. In this appendix we briefly summarize the data structures and packet formats used in SLURP.

Data Structures

The primary data structures stored at each node i are the following:

- A location table (LT) that stores the current location of all nodes that are registered in i's current region.
- Location tables for zero or more neighboring regions that are empty (recall that if a region is empty, then all neighboring regions are regarded as the new home region of the nodes registered in the empty region).
- A cache containing the last known location of various nodes with a time stamp. The cache also contains the ID of the next hop neighbor to get to the destination.
- A cache containing the ID of nodes located within the same region and source routes for all these nodes.
- Other information stored includes a representation of the function $f()$, the number of regions in the network.
- A table containing the (Lat, Long) coordinates of the center of each region. The table also describes the boundary of the region (i.e., the region may be a polygon in the general case and this shape is stored in the table).

Location Tables (LT)

Let us assume that the node i is in Region R and the neighboring regions are $N_1, \cdots, N_m$. Node i maintains a LT for region R (call this Level1LT) and possibly one table for each neighboring region (Level2 LTs). A bit vector (of length m) is used to indicate to node i if it is responsible for maintaining location information for nodes registered in a specific neighboring region as well. Each LT contains the following columns:

Node ID of registered nodes – 32 bits
Sequence number of last location_update packet – Integer
Current Region ID of node – Integer
Time of last update – Integer
Last reported Velocity – (Integer, Integer)

The table is sorted by the Node IDs in the first column. We used 32 bits in the original implementation because we rely on IP addresses. The region ID consists of an index into the Region Table (RT) discussed next. The sequence number field is used to distinguish old location_update packets from new ones. The fourth value denotes the time that the last update was received. Each time a location_update packet is received, this value is reset to zero. After that it counts up every second until it reaches the maximum value after which it stays at that max value until reset by a new location_update message or until it is removed from the table (this corresponds to the case when that node has left the network). The last entry corresponds to the last reported speed (in centimeters/sec) and direction (0 to 2π) of the node. We use cm/sec to get a reasonable accuracy in representing the speed via integer values. Likewise, the direction is represented by dividing the range 2π into 3600 discrete values and rounding off the actual direction to the closest value.

Region Table (RT)

Each node maintains a table that includes information about all the regions present in the network. The information stored in these tables is as follows:

Region ID (Integer)	Latitude (Integer)	Longitude (Integer)	Shape Vector (Pointer)

The Region ID is a number 1 to k where k is the total number of regions in the network. The latitude and longitude represents the center of the region. We use integers to store this information with the a resolution of 1/10th of a degree (note that we could use finer resolutions as well because an integer gives us plenty of accuracy but this sufficed for our purposes). Finally, the shape vector points to a linked list containing the following information:

Latitude1 (Integer)	Longitude1 (Integer)	Latitude2 (Integer)	Longitude2 (Integer)	Pointer

The pair of latitude and longitude coordinates represents the coordinates of the two endpoints of one edge of the polygon that surrounds the region.

Location Cache

This is a table that stores the last known location of nodes that node i either communicated with or overhead a transmission to/from. The entries are:

Node ID (32 bits)	Region ID (Integer)	Timestamp (Integer)	Next Hop (32 bits)

The timestamp is set to a THRESH value when the entry is updated. After that it is decremented every second and when the value reaches zero, that entry is discarded. The last entry in the table indicates the best 1-hop neighbor to use for forwarding data packets to that node.

Region Cache

This cache (also called node_list) contains the node ID of all nodes currently located in node i's current region. The cache also contains source routes to the nodes (note that the source routes may be stale - they are only updated when there is a need to send packets to that node). We use DSR within a region to route packets between nodes.

Node ID (32 bits)	MAC address (48 bits)	Timestamp (Integer)	Source Route (List of 32 bit node IDs)

Other

The function $f()$ is stored either as a hash table or as a function that gets executed every time a mapping neds to be made. In our implementation we used a hash table to speed up lookup. Other stored information includes the number of regions k, location of any static nodes in the network (these may be gateway nodes that connect the ad hoc network to the Internet), and any other information.

Packet Formats

Note that *all* packets have a TYPE field (we use an Integer). However, we have not included this field in the description below for clarity.

location_update and location_update_broadcast Packets

This packet is sent by a node to its home region whenever it crosses a region boundary. The format for this packet is:

Seq. No.	Node ID	Dest Region ID	Old Region ID	New Region ID	Velocity

The sequence number field is used to deal with old location_update packets. The Dest. Region ID field is the ID of the node's home region and is used for hop-by-hop routing purposes (each intermediate node uses MFR to find the next hop to reach the destination region). The Old Region ID is the nodes previous region and the New Region ID is the ID of the new region the node has entered. The velocity vector (two integer values) denotes the current speed and direction of the node.

The first node receiving the location_update packet in the home region generates a location_update_broadcast packet (same format, different type field) which is broadcast within that region. Note that if the home region is empty, this broadcast is conducted in each neighboring region via a separate location_update_broadcast packet.

location_discovery and location_reply Packet

This packet is sent when a node wants to find the location of some other node. The packet contains the following information:

Seq. No.	Sender Node ID	Destination Region ID	Source Region ID	Queried Node ID	Discovery Level

The Queried Node ID field is the IP address of the node whose location is being queried. The Source region ID is the current region of the querying node (sender node) and the destination region ID is the ID of the queried node's home region. Sequence numbers are included to handle stale location discovery packets. The discovery level field can be either 1 or 2 and is used as follows: if the home region of the queried node is empty then a level 2 query implies that the location_discovery message will be broadcast in the neighboring regions as well. A level 1 query does not spread the query to the neighboring regions. Note that if the home region does have at least node then that is the only region queried in both cases.

The location_reply packet is generated by the first node in the queried node's home region. This packet has the following structure:

Seq. No.	Seq. No of Location_Discovery Packet	Receiver Node ID	Receiver Region ID	Region ID of Queried Node

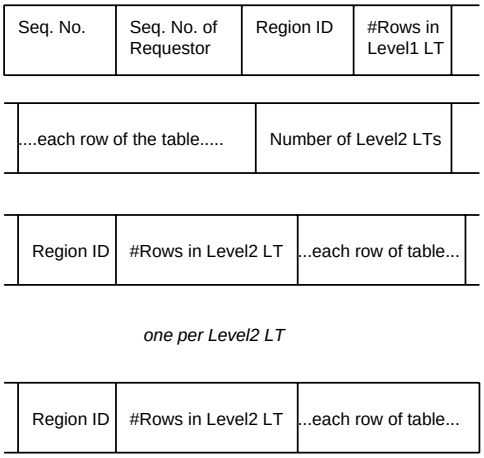
The reply packet has its own sequence number and, in addition, echos back the sequence number of the location_discovery packet. It also contains the ID and region ID of the node that had sent the location_discovery packet. Finally, the region of the queried node is sent back as the data part of the response.

6.0.1 home_location_request and home_location_reply Packet

When node i moves into a new region, it sends this message to all its neighbors in the new region to collect location information about nodes registered in this region. The format of this packet is as follows:

Seq. No.	Node ID	Region ID

The Node ID is the ID of node i, the region ID is the ID of the new region that node i has entered (this information is sent because of the case when we have overlapping regions and a node is attempting to discover location information of a region that overlaps). The response to this packet contains the all the LTs (Location Tables) maintained by nodes in this region. The format of this message is:



The reply packet has its own unique sequence number but it also echos back the sequence number of the query packet. In addition, it contains the Region ID of the current region. The next part of the packet contains a count of the number of rows in the LT for that region followed by the LT sent in row-major form. It is possible that nodes in this region also maintain location information for nodes registered in empty neighboring regions. This means that the nodes in this region may have zero or more level2 LTs for neighboring regions. Therefore, the next entry in the packet denotes the number of Level2 LTs that are being sent followed by a repeating sequence of <Region ID of neighboring region, # Rows in Level2 LT, all rows of Level2 LT> for all these Level2 LTs.

exit_region and empty_region Packet

This packet is broadcast by a node when it leaves a region. The broadcast is confined to the region it just left. If, however, the node is the last node that leaves the region then it does not send the exit_region packet but sends a empty_region packet to all the surrounding regions. The empty_region packet contains the LT for the empty region. The format of the exit_region packet is:

Seq. No.	Node ID	Region ID

The Node ID is the departing node's ID and Region ID is the ID of the region the node is exiting. The broadcast is confined to nodes within this region only. The format of the empty_region message is as follows:

Seq. No.	Node ID	Source Region ID	Destination Region ID	<Level1 LT for region>

The <Level1 LT for region> entry above denotes transmission of the location table in row-major form. The above packet is sent to each of the neighboring regions (Dest. Region ID) of the now-empty region (Source Region ID).

home_location_request_2 and home_location_reply_2 Packets

When a node enters an empty region, it transmits a home_location_request_2 packet to all neighboring regions. The format of this packet is the same as ther home_location_request packet. Nodes receiving this request reply with the home_location_reply_2 packets (same format as the home_location_reply packets). When nodes in a neighboring region receive the home_location_request_2 packet, they can discard the Level2 LT corresponding to the region (that was empty but now has one node in it).