L^+: Scalable Landmark Routing and Address Lookup for Multi-hop Wireless Networks

Benjie Chen Robert Morris

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts, 02139
{benjie,rtm}@lcs.mit.edu

Abstract

This paper proposes and analyzes modifications to the Landmark routing system that make it better suited to large ad hoc wireless networks. Most existing ad hoc routing algorithms scale badly in the sense that they generate protocol overhead whose per-node cost grows linearly with the total number of nodes. The Landmark routing protocol solves this problem by use of hierarchical addresses that contain routing hints; as a result, however, a node's address changes as the network topology changes. The Landmark system tracks node addresses with a distributed ID-to-address location service, but queries to this service require communication with random non-local nodes, which scales badly in large networks.

The main contribution of this paper is a set of modifications to the Landmark address lookup service to make it more scalable. The paper also improves the Landmark hierarchy maintenance and routing algorithms to help them react better to mobile nodes. Finally, it presents a simulation evaluation of the resulting system, L^+, from the point of view of scalability in wireless ad hoc networks. The evaluation shows that the per-node bandwidth requirement of L^+ grows very slowly as the number of nodes in the network increases. This is consistent with our analysis that the per-node communication cost of L^+ is $O(\log N)$.

1 Introduction

This paper addresses the problem of scalable routing in large ad hoc wireless networks of mobile nodes. Such networks are of interest because they do not rely on fixed infrastructure. As a result these networks can support a number of promising new technologies such as ubiquitous computing [24], sensor networks, rooftop networks [1, 19], and wireless PDAs. A major obstacle to the use of large ad hoc networks is the lack of an adequately scalable routing system. This paper describes and analyzes modifications to the Landmark routing system [22, 21, 23] to make it suitable for large ad hoc wireless networks.

An important way in which wireless ad hoc networks differ from wired networks, and wireless networks with wired backbones, is that they are likely to have severely constrained capacities. Each node's radio is likely to have the same capacity; an engineered high-capacity wireless backbone is likely to be awkward in many ad hoc scenarios. More fundamentally, the nodes are embedded on a plane, with connectivity only to nearby nodes. Assuming uniform node density, the expected distance between a random pair of nodes is $O(\sqrt{N})$ in both physical distance and number of hops, where N is the total number of nodes; similarly, the cross section bandwidth of the network is also $O(\sqrt{N})$. This means that if communication patterns tend to be long-distance, or even random, the average amount of traffic that any one node can originate scales as $\frac{1}{\sqrt{N}}$ [7, 12]. That is, the more nodes there are, the less long-distance traffic any one node can originate. This holds true even if the area of the universe (and thus the degree of spectrum re-use) scales with the number of nodes.

As a consequence of this capacity constraint, the dominant traffic patterns in large ad hoc networks will probably need to be local [12]; this would allow each node to originate an amount of traffic independent of the total size of the system. However, it is not enough that the traffic pattern be local: the per-node overhead generated by the routing protocol must also grow slowly with total network size. One consequence of this is that, ideally, the per-node routing overhead should be a constant independent of the size of the system. More practically, the per-node overhead should be a slowly growing function of the system size, such as $O(\log N)$. For example, this rules out standard distance-vector, which has a per-node communication cost of $O(N)$. For reactive protocols, which query for routes to destinations only as needed, capacity constraints suggest that queries should travel a distance proportional to the distance between the nodes desiring to communicate; otherwise local communication will generate global routing traffic, which won't scale well.

Few existing ad hoc routing protocols conform to the restrictions described above. For example, DSDV [17] uses a

distance-vector algorithm that imposes $O(N)$ per-node communication cost, where N is the number of nodes in the network, while DSR [4] and AODV [18] flood queries globally even for local communication. As a consequence, we should expect these protocols' overhead to exhaust node radio capacities relatively quickly as networks grow larger. In practice, the situation is not this simple. If the network topology does not change, these protocol's overheads can be made arbitrarily small. Even if the topology changes, DSR and AODV have caching and local re-query mechanisms that limit the cost of finding and repairing routes. Still, the overall scaling argument suggests that these protocols might work badly in very large ad hoc networks. Section 5 shows that this is true.

More scalable ad hoc routing protocols do exist. For example, the combination of geographic forwarding and the GLS location service [13] provides a routing system that scales as $O(\log N)$. However, both geographic forwarding and GLS require that nodes know their geographic locations, perhaps using the Global Positioning System (GPS). This dependence is likely to be impractical for many uses of ad hoc networks.

Landmark routing is a potentially scalable protocol that does not depend on GPS. Instead of using geographic locations as addresses, Landmark addresses nodes using their positions in a dynamically maintained hierarchy. Landmark addresses effectively encode an abbreviated route in the form of a path down the hierarchy. These addresses allow packets to be routed with very little per-node state, and thus little per-node routing overhead. Landmark limits the number of nodes in the network that any one node knows about to $O(\log N)$. Thus the per-node routing overhead is $O(\log N)$. However, since a node's address may change when the network topology changes, the complete Landmark system includes a distributed database that maps each node's permanent ID to its current address. Queries to this database require global communication; thus the complete Landmark system as originally described is not likely to scale well in large ad hoc networks.

This paper introduces L^+, a modified Landmark routing system designed for large ad hoc mobile networks. L^+ differs from Landmark mainly in its location service. Unlike Landmark, the number of hops each L^+ location query takes is proportional to the distance between the sender and the receiver. Hence, L^+ avoids global communication when the underlying traffic pattern is local. The location update traffic in L^+ also follows the power-law distribution, making it mostly local. Finally, L^+ modifies the Landmark hierarchy maintenance and routing algorithms to make them react better to mobility. In Section 5 we use simulations to show that L^+ scales well; the per-node bandwidth requirement of L^+ grows very slowly as the number of nodes in the network increases. This result is consistent with our analysis that the per-node communication cost of an L^+ node is $O(\log N)$. In addition, L^+ scales better than original Landmark, particularly for local communication patterns.

The rest of the paper is structured as follows. Section 2 describes the original Landmark routing system. Section 3 describes the L^+ location service. Section 4 describes the hierarchy maintenance and routing algorithms used in L^+. Section 5 analyzes performance of L^+, and compares it with Landmark and DSR. Section 6 discusses related work. Finally, Section 7 concludes.

2 Landmark Overview

This section reviews the original design of Landmark routing [21, 22, 23]. Landmark is a distributed routing protocol designed for large networks with loose administrative domains and changing topology. Landmark creates and maintains a hierarchy of nodes that reflects network topology. A node's address is its position in the hierarchy. Routing based on these addresses requires very little state; each node need only know its own parent (to forward up towards the root) and its own children (to forward down towards the leaves). It is the limited size of the per-node state, and correspondingly limited state update communication, that allows Landmark routing to scale to large network sizes.

Each Landmark node has a unique, unchanging identifier (ID); this might be, for example, an IP address. In addition, each Landmark node has an address, which changes. The Landmark system includes a distributed location service to map from IDs to addresses.

2.1 Landmark Hierarchy

The Landmark hierarchy is a tree of nodes, called landmarks. By convention, the leaves are called level 0 landmarks. Every node starts out as a level 0 landmark. Each level l landmark picks a level $l{+}1$ landmark within a radius of r_l hops as its parent. If no level $l{+}1$ landmark exists within r_l hops, the node participates in an election to choose a new level $l{+}1$ landmark. This means that roughly one node in each area of radius of r_i becomes a level i landmark. Eventually, one node becomes the root landmark of the entire hierarchy.

The radius at level 0, r_0, is 2 network hops. It doubles every level, so $r_i = 2r_{i-1}$. Consequently, the radius of the area covered by the top level landmark, at level H, is $O(2^H)$. As a result, the number of landmark levels needed for a network is $O(\log N)$, where N is the number of nodes in the network.

Landmarks learn about each other by running a modified distance-vector (DV) routing protocol. Each landmark places a limit on the number of hops that its information propagates in the DV protocol; this limit is $2r_l$ for a level l landmark. This radius is the *advertisement distance*. Consequently, the number of nodes that know about a particular landmark increases as the level of that landmark increases. Francis [21] shows that even though the number of landmarks elected at each level $l{>}0$ grows as $O(N)$, the number of landmarks a node knows about at each level $l{>}0$ stays at a constant value

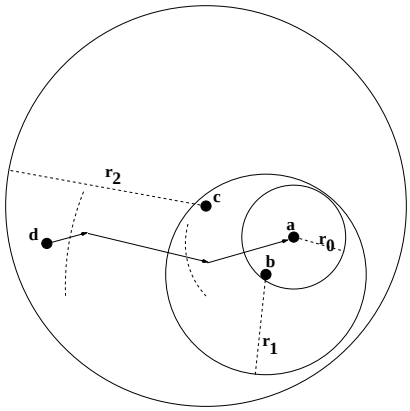

Figure 1: Example of a Landmark hierarchy. Node a is a level 0 landmark. Node b is a level 1 landmark, within radius r_0 of a. Node c is a level 2 landmark, within radius r_1 of b. Node a's address is $a.b.c$. The two dotted curves represents the advertisement boundary from b and a. A packet that d addresses to $a.b.c$ is forwarded along the path represented by the sequence of arrows.

until the total number of landmarks elected in the network at that level drops below the constant. Thus, the total number of landmarks each node knows about is $O(\log N)$.

As the network topology changes, the number of hops between a node and its parent may increase to the point where the node must choose a new parent. Similarly, topology changes may require new landmarks to be elected, or old ones to be demoted. A level l landmark increments its landmark level when there are no other $l+1$ landmarks that can cover all the level l landmarks in the vicinity. Similarly, a level l ($l > 0$) landmark decrements its level when all the level $l-1$ landmarks in the vicinity can be covered by another level l landmark.

Figure 1 shows an example of a simple hierarchy. In this figure, node a is a level 0 landmark. Node b is a level 1 landmark, within r_0 radius from a. Node c is a level 2 landmark, within r_1 radius from b. Node d, far away from a, knows about the root landmark c, but not b or a.

2.2 Landmark Routing

A node's Landmark address is composed of the node's ID, followed by its parent's ID, then the ID of the parent's parent, and so on, and eventually the root landmark's ID. For example, the address of node a in Figure 1 is $a.b.c$.

When a node receives a packet that it must forward, it looks for each component of the destination address in its own DV routing table. It will certainly find a routing table entry for the root landmark. As the packet moves towards the destination, or even towards the root, forwarding nodes are also likely to find other address components in their routing tables. A forwarding node uses the routing table entry corresponding to the left-most (lowest level) known component in the address. It forwards the packet to the next-hop node indicated by that entry.

A landmark may not forward a packet even if it is one of the components in the destination address of the packet. It is very likely that before the packet reaches this landmark, it was redirected toward another landmark to the left of this landmark in the destination address. For example, in Figure 1, when node d sends a packet to a, the packet moves along the path formed by the arrows. d first sends the packet toward c. When a forwarding node less than $2r_1$ hops (i.e. advertisement distance of b, represented by the dotted curve) from b receives the packet, it forwards the packet to b. Similarly, when a forwarding node less than $2r_0$ hops from c receives the packet, it forwards the packet to c.

Landmark routing does not typically forward a packet using the shortest path. Often when a source node sends a packet, the packet moves toward a higher level landmark before being redirected toward the destination. [21] provides detailed analysis in terms of path length increase as the size of the network grows.

It would be undesirable if a packet had to get within a few hops of one of the address components before it could start to be forwarded to the next more-specific component. This would cause, for example, the nodes around the root landmark to experience heavy forwarding load. This turns out not to be the case. Suppose that a packet is moving towards a level l landmark that is $2r_l$ away (the full length of the level l advertisement distance). This packet can be redirected towards a level $l-1$ landmark as soon as it is within the advertisement range of the level $l-1$ landmark, $2r_{l-1}$. Because the level $l-1$ landmark is at most r_{l-1} away from the level l landmark, at the point where redirection occurs, the packet is still, in the best case

$$\frac{2r_{l-1} - r_{l-1}}{2r_l} \cdot 2r_l = \frac{1}{4} \cdot 2r_l \qquad (1)$$

hops away from the level l landmark. Therefore, all nodes that are within $\frac{1}{4} \cdot 2r_l$ hops away from a level l landmark can be used to redirect packets to level $l-1$ landmarks. Since a packet will be redirected as soon as it enters this area, nodes on the perimeter of this area assume the role of the level l landmark. This implies that the load of the root landmark is spread among $O(\sqrt{N})$ nodes.

2.3 Landmark Location Service

One of the difficulties of Landmark routing is that Landmark addresses change as the topology changes. To solve this problem, Landmark provides an ID-to-address location service that works as follows. A node picks its location server by taking the hash of its own ID, and uses the hash result as the Landmark address of its location server. Every time a node's address changes, the node sends a location update to

its location server. When a sender wants to send a packet to a receiver, the sender computes the Landmark address of the receiver's location server by taking the hash of the receiver's ID.

A problem with this approach is that the hash of a node's ID would most likely not map into a usable Landmark address. To solve this problem, a hashed address is resolved into a real Landmark address level by level. A node sends the location update or query packet towards the root of the hierarchy first. When the node forwarding this packet is close enough to the root that it knows about all the root landmark's immediate children, it forwards the packet towards the child whose ID is closest to the hashed address. This process continues at each level until the packet is forwarded to a level 0 landmark. This level 0 landmark is the desired location server.

This mechanism for choosing a node's location server has the following good properties. It distributes the work of storing locations evenly across the nodes in the network. It allows any node to find a target node's location server, and thus the target node's address, given only the target node's ID. Finally, if a node's location server moves or fails, everybody automatically agrees on how to find a new location server.

```
choose-landmark(id, level)
    selected = −1
    closest = 0
    for each landmark l in DV table
        if (l.level == level)
            d = abs(hash(l.id)-hash(id))
            if (d < closest or selected == −1)
                closest = d
                selected = l.id
    return selected

choose-child(id, parent, level)
    selected = −1
    closest = 0
    for each landmark l in DV table
        if (l.level == level and l.parent == parent)
            d = abs(hash(l.id)-hash(id))
            if (d < closest or selected == −1)
                closest = d
                selected = l.id
    return selected
```

Figure 2: Procedures used when resolving the hash of an ID into a real Landmark address. Each node uses the `choose-landmark` procedure to select a landmark at each level to send location update or query to. Each landmark, subsequently, uses the `choose-child` to propagate the update or query downward in the hierarchy.

3 The L^+ Location Service

The Landmark location service does not scale well to large ad hoc wireless networks because its location update and query traffic is global. For every location update or query, a node must send a packet to a server chosen among all of the nodes in the network. This means on average, a location query packet travels $O(\sqrt{N})$ hops, even if the two nodes that want to communicate are close to each other. Landmark lookups effectively turn scalable local communication patterns into unscalable global patterns.

L^+ provides a location service that scales well if the communication pattern is local. An L^+ node sends location updates to more than one location server. The location servers are chosen such that the expected distance to each server is exponentially farther away from the node. When a node performs a location query for a destination, with a high probability, L^+ resolves the query using a nearby location server.

3.1 Server Selection

At each level l of the hierarchy, a node sends an update to a level l landmark it knows of (in its DV routing tables) whose hashed ID is numerically closest to the node's hashed ID, `hash(`id`)`. It chooses this landmark using the `choose-landmark(`id`,`l`)` procedure call. Pseudo code for the procedure is shown in Figure 2. This level l landmark then sends the update downward in the hierarchy, just as in the original Landmark location server mechanism, to its level $l-1$ child with hashed ID closest to `hash(`id`)`. It obtains this child by calling the `choose-child` procedure. For example, when a level 2 landmark a receives an update that needs to be propagated downward, it calls `choose-child(`id`,`a`,1)` to select one of its level 1 children. If this procedure returns the level 1 landmark b, the update is then sent to b. b, or a node near b, calls `choose-child(`id`,`b`,0)` to select the final location server.

An update or query sent to a level l landmark a does not need to reach a. A forwarding node close to a can start pushing the update or query downward in the hierarchy (i.e. calling `choose-child(`id`,`a`,`$l+1$`)`) if it has all the children of a in its DV table. This is always the case if the forwarding node is r_{l-1} hops away from a: every child of a is at most r_{l-1} hops away from a, and each of those children has an advertisement distance of $2r_{l-1}$ hops.

When a node a wants to look up the current address of a destination whose ID is b, it first sends a query to the level 1 landmark it knows of whose hashed ID is closest to `hash(`b`)`. This landmark, obtained by calling `choose-landmark(`b`,1)`, then sends the query downward in the hierarchy to its level 0 child with hashed ID closest to `hash(`b`)`. If that child doesn't know about b, a tries again at level 2, and so forth.

The intuition behind this scheme comes from the obser-

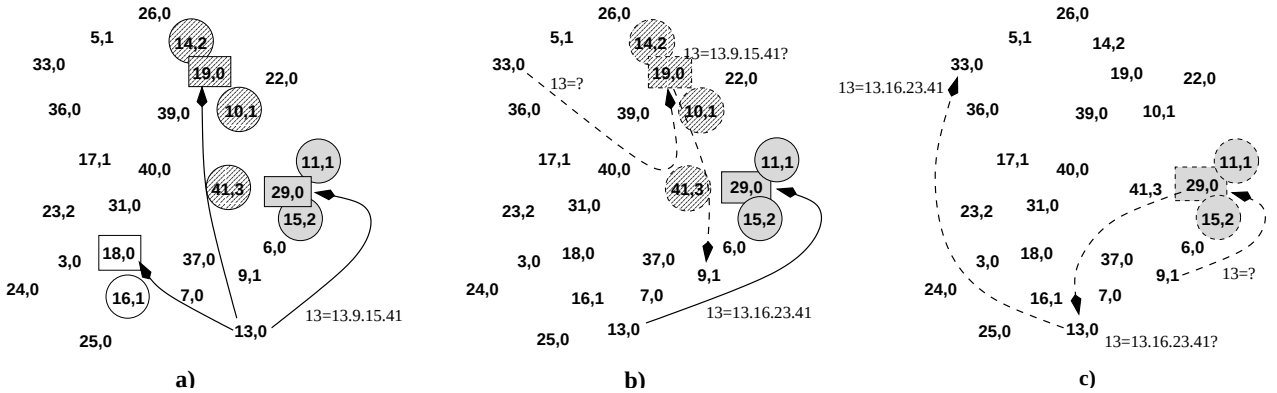

Figure 3: Updating and querying L^+ location servers. Each node appears as `hash(ID),level` in the picture. Solid lines represent location updates. Dotted lines represent location queries. Circled nodes are intermediate nodes in the address resolution process. Boxed nodes are the final location servers. Circled or boxed nodes with the same shade are part of the same resolution process. The example is explained in Section 3.2.

vation that the number of landmarks two nodes both see depends on the distance between these two nodes. If they are very close to each other, they may see the same set of level 1 landmarks. If they are slightly farther apart, they may see different level 1 landmarks, but the same set of level 2 landmarks. The closer the two nodes are, the more likely that calls to `choose-landmark` would return the same result at lower levels of the hierarchy. Thus nodes that are close physically can look up each others' addresses with local query communication.

3.2 Updating and Querying Location Servers

When a node's Landmark address changes, it only sends updates to a subset of its location servers, in a way that ensures local motion usually generates local update traffic. If the Landmark address changes starting at the level l component, an update is sent to the landmark at level $l+1$. Each update contains a timeout which is proportional to the expected interval between sending updates to that level of the hierarchy. In addition, nodes send location updates at a slow rate proportional to r_l to landmarks at each level, even when stationary.

The frequency at which the address of node a changes depends on the frequency at which a changes its parent and the frequency at which other components in the address of a change their parents. A level l landmark picks a new parent when its old parent is more than r_l hops away. Hence, the frequency at which the level $l+1$ component of an address changes depends on the mobility rate relative to r_l. Therefore, components of an address that correspond to the low levels of the hierarchy may change relatively often. On the other hand the higher level components of the address will change less frequently. Because low-level changes generate local updates, the location update traffic follows a power law pattern.

Sending updates less frequently to distant servers implies that the addresses stored at distant servers may become stale. Therefore, instead of answering a query directly, each location server forwards the query towards the destination using the address stored in the location database. If the address is stale, a node near the old location may still have the node in its DV table. A query with a stale address can also be incrementally refined to obtain a correct address.

The refining process works as follows. Assume node a sends a query for destination b very far away. Consider that one of a's queries reaches a location server that has an old address for b. The query is forwarded to b's old location in the hierarchy. When a forwarding node can no longer forward the query using the stale address, it may choose to refine the query. If forwarding failed because the level $l-1$ component of the address cannot be reached, the forwarding node sends a location query for b starting at level $l+1$. The intuition is that the node has not moved too far away from its old location, and therefore consulting a nearby location server is likely to produce the correct address. Furthermore, if the address broke at level $l-1$, that means it is likely that the level $l-1$ landmark changed its parent. In that case, an update would have been sent to a level $l+1$ server.

Figure 3 shows an example of how updating and querying L^+ location servers work. Each node in the network appears as `hash(ID),level` in the picture. Solid lines represent location updates. Dotted lines represent location queries. Circled nodes are intermediate nodes in the address resolution process. Boxed nodes are the final location servers. Circled or boxed nodes with the same shade are part of the same resolution process. In a), node `13` selects three landmarks to send location updates to using `choose-landmark(13,1)`, `choose-landmark(13,2)`, and `choose-landmark(13,3)`. The three chosen landmarks are `16`, `15`, and `41` respectively. Each of these landmarks forwards the location update by using the `choose-child` procedure. For example, when node `15` receives the update, it

calls `choose-child(13, 15, 1)`, which returns node 11. The update is then sent to node 11. Node 11 then calls `choose-child(13, 11, 0)`, which returns 29. 29 is the final location server. In b), node 13 moves and acquires a new address 13.16.23.41. Since the level 1 component of the address changed, it sends an update to the level 2 landmark computed using `choose-landmark(13,2)`. In this case it is still node 15. The location update eventually reaches location server 29. In the meantime, node 33 sends a query for 13 through the root landmark 41. The query is resolved via 14 first, then 10. Location server 19 finally forwards the query to 13's old location, 13.9.15.41. Node 9, however, can no longer forward to 13. In c), because forwarding failed at level 0, node 9 selects a level 2 landmark by calling `choose-landmark(13,2)`. The query is sent to node 15 again. It eventually reaches location server 29. 29 forwards the query to node 13. Finally, 13 answers the query using the source address in the query packet.

3.3 Scalability of L^+

This section considers the expected per-node bandwidth requirements of an L^+ node in a static network with local communication.

Four items contribute to the per-node bandwidth. First, the DV protocol used for Landmark hierarchy maintenance and routing. Francis [21] shows that this overhead is $O(\log N)$. Second, each L^+ location update packet contributes to the per-node bandwidth of every node on its forwarding path. Because a node sends location updates exponentially less often to far away location servers, the L^+ location update traffic pattern follows the power-law distribution. Hence, the per-node overhead from forwarding location updates is $O(\log N)$. Third, forwarding a L^+ location query packet also contributes to per-node bandwidth. If communication is local, then with high probability L^+ uses a nearby location server to resolve each query. Thus the overhead of forwarding location queries is $O(1)$. Fourth, the local communication pattern contributes an overhead of $O(1)$ to the per-node bandwidth. Adding them together, the expected per-node bandwidth of an L^+ node in a static network with local communication grows as $O(\log N)$ in the worst case.

In original Landmark, the expected per-node bandwidth requirement in a static network with local communication pattern can be dominated by the need to send location updates and queries to random nodes, imposing a $O(\sqrt{N})$ per-node communication costs in the worst case.

Mobility complicates the analysis of the overhead of forwarding location queries, since queries to nearby location servers may fail. If the probability is high that a query for a nearby destination can be resolved by a nearby server, then the per-node bandwidth remains $O(\log N)$ in the worst case.

4 Handling Mobility in L^+

This section describes several changes L^+ makes to the hierarchy and routing algorithms of the original Landmark system. These modifications make L^+ react better to mobility. In our simulations, we used a Landmark implementation with these changes.

Similar to Landmark, L^+ uses a distance vector algorithm to distribute information about landmarks. Each node periodically advertises its own information (i.e. node ID, landmark level, advertisement radius, how many potential parents it has, a chosen parent, and a secondary parent) in addition to the nodes in its routing table. A routing table entry is only advertised if the distance to the node is less than the node's advertised advertisement distance. Most nodes have an advertisement distance of $2r_l$ where l is the node's Landmark level. The root landmark and landmarks with the three highest IDs in the second highest level are designated as *global landmarks* and have advertisement radii of infinity.

4.1 Building the Hierarchy

If a node with Landmark level l cannot find a level l+1 parent within r_l hops, it considers incrementing its Landmark level to l+1. To prevent several nodes within r_l of each other from incrementing their Landmark levels at the same time, a node scans its routing table first. It increments its Landmark level only if it has the highest ID among all eligible nodes (i.e. nodes that advertise 0 as the number of potential parents) within r_l.

This election algorithm tends to promote just one landmark in each area that needs one. A disadvantage of the algorithm is that it may promote higher level landmarks slowly, since a node must wait long enough that news of a distant high-level landmark's promotion would reach it before it can promote itself.

With mobility, a level l landmark may move into a region and discover that all landmarks of level $l-1$ in this region have parents already. In L^+, a level l landmark decrements its Landmark level to $l-1$ if it sees that every level $l-1$ landmarks within r_{l-1} hops away has at least 2 potential parents. Using 2 instead of 1 provides both redundancy and stability, as other level l landmarks could be moving away. A node does not decrement its Landmark level if it sees that it will no longer have a parent after doing so.

Unlike Landmark, L^+ does not require explicit registration between parent and children. A level l landmark can pick any level l+1 landmark within r_l hops as its parent. In practice, to reduce the number of address changes, a node picks the nearest landmark among its potential parents. Additionally, each node also picks the second nearest landmark among its potential parents as a secondary parent. If there is only one potential parent, the secondary parent is the same as the parent.

Because the advertisement radius of a level l landmark is $2r_l$ hops, and every level $l-1$ landmark must have a level l parent within r_l hops, a node sees DV updates from all of its ancestors. Consequently, a node can compose its own Landmark address from its routing table.

A node composes two Landmark addresses for itself. The first address is composed using the node's parent, node's parent's parent, etc. The second address is composed using the node's secondary parent, the secondary parent's secondary parent, etc. In most cases, the two addresses differ at multiple components. To make routing work better, each node sends location updates with both addresses. An address lookup also returns both addresses, and every data packet is tagged with both addresses as well.

4.2 Moving Location Database Entries

A change in the Landmark hierarchy may change how a node's hashed ID maps to the nodes that act as its location servers, even if it doesn't change the node's address. If nothing special were done, it might take a long time before the node updated its new location servers. To address this problem, each L^+ node periodically scans the location entries it stores, looking for entries that should be stored by one of the other children of its parent. It forwards such entries to the relevant child.

4.3 Routing in L^+

The distance vector algorithm used by L^+ is similar to DSDV [17]. It differs from DSDV in that L^+ keeps more than just the shortest route to each destination. If the shortest distance to a destination is determined to be d hops, L^+ keeps a list of routes with distance d hops or $d+1$ hops. If d is in fact the shortest route, then a route with a distance of $d+1$ hops cannot contain a loop, since each loop causes at least 2 additional hops. The distance advertised for a destination is the distance of the first route in the route list. Keeping a list of routes instead of one route allows a node to use alternate routes when the shortest route breaks. To prevent loops, trigger update is used to propagate the metric change, if any, when the shortest route on the route list is removed either due to timeout or MAC transmit feedback (i.e. transmit to the next hop indicated in this route failed).

Packet forwarding in L^+ works as follows. When a node receives a packet that it must forward, it looks for each component of the destination address in its own routing table. A forwarding node uses the routing table entry corresponding to the left-most (lowest level) known component in the address. A routing failure occurs if the packet has previously reached the left-most known component in the address, or if no known component exists. We switch to the second destination address if a routing failure occurred using the first destination address. We drop the packet if a routing failure occurred using the second address.

5 Simulation Results

This section presents results from a L^+ implementation in the `ns-2` [14] network simulator, using 802.11 as the MAC. It first shows that routing in L^+, with a perfect location service, scales well. Then, using a static network and local communication pattern, we isolate the overhead of location queries. Finally, we demonstrate the scalability of L^+ in two potential styles of deployment: a mobile ad hoc network where all nodes are moving, and a rooftop wireless network where routers are stationary but clients of the network move. For comparison, results are also shown for DSR [4] and for the original Landmark system augmented with the routing and hierarchy maintenance mechanisms described in Section 4.

Our goal is to show that per-node communication requirements in L^+ grow slowly with the total number of nodes. In order to be able to observe the amount of per-node bandwidth required to support large networks, we effectively eliminated the capacity limit of the simulated 802.11 radios (by setting the capacity to 100 Mbps rather than 2 Mbps). The actual bandwidth recorded is the sum of transmit and receive bandwidth values in each 1-second interval, counting only successfully received packets. The bandwidth results show both the median and 99th percentile values of all 1-second measurements over all nodes. The 99th percentile gives an indication of how fast node radios would have to be in order to avoid congestion at the vast majority of nodes.

In our simulations, the radio range is 250 meters. Unless otherwise noted, each scenario starts out with an average node density of 10 nodes per radio range. We increase the area of the network accordingly as the number of nodes increases. Mobile nodes follow the random waypoint model with no pause time: initially, each node chooses a destination uniformly at random in the simulated region, chooses a speed uniformly at random between 0 and 10 m/s, and moves there with the chosen speed. Upon arrival, the node immediate picks another destination and speed and repeats the same process.

Unless otherwise stated, the simulated communication pattern is as follows. Each node in the system sends traffic to one other randomly selected node; the selection is done in a way that ensures that no node receives traffic from more than two other nodes. Each node sends a total of 15 128-byte packets at a rate of 3 packets per second. Each simulation lasts 1,200 seconds. The start time of each of these flows is randomly chosen over the last 400 seconds of each simulation. This allows the hierarchy to be constructed in the first 800 seconds of the simulation (most do not take nearly this long).

The DSR code in `ns` was modified to have a maximum route request length of 32 hops instead of the default 16 hops; this allows DSR to reliably find paths in the larger simula-

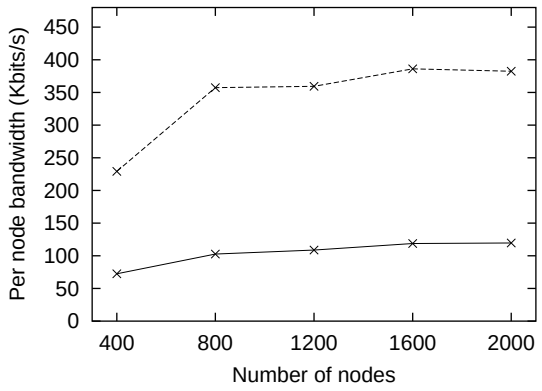

Figure 4: Per-node bandwidth of L^+ routing with a perfect location service. The dotted line represents 99th percentile values; the solid line represents median values.

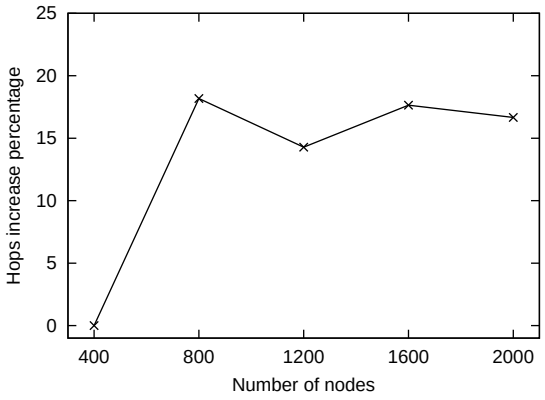

Figure 5: The percentage by which L^+ routes are longer than geographic shortest paths, as a function of total number of nodes in the network.

tions.

In the bandwidth graphs presented below, the dotted lines represent 99th percentile per-node bandwidth values, and the solid lines represent median values. Unless otherwise noted, each point represents results from just one simulation run. (We will fix this; we believe that lack of multiple runs doesn't affect the results much because node mobility causes constant change in the topology.)

5.1 Scalable Routing

Figure 4 shows the bandwidth required to route packets using L^+ as a function of the number of nodes in the network. The simulations in this graph operated with no location service: each node magically knows the current correct address of the node it is sending data to. Since the location service is not in use, DV routing updates make up most of the traffic. The shape of the required bandwidth curve can be explained by the fact that, in a Landmark hierarchy, the number of destinations advertised in each DV update is $O(\log N)$ [21]. Our simulation reports the same relationship between the number of Landmarks in each node's DV table and the number of nodes in the network.

The slow growth of the 99th percentile shows that no node or small set of nodes acts as a bottleneck; in particular, it is not the case that many packets have to be routed through the root of the hierarchy or the nodes immediately surrounding it.

Figure 5 compares per packet hop count between L^+ routing and shortest path routing. Shortest path routing was approximated using geographic forwarding with a perfect location service; the reason for not computing the actual shortest path is that it is not well defined, since nodes move while packets are in transit. The approximation is that each node forwards a packet through the neighbor geographically closest to the destination. The graph shows that L^+ uses routes that are a few tens of percent longer than shortest path.

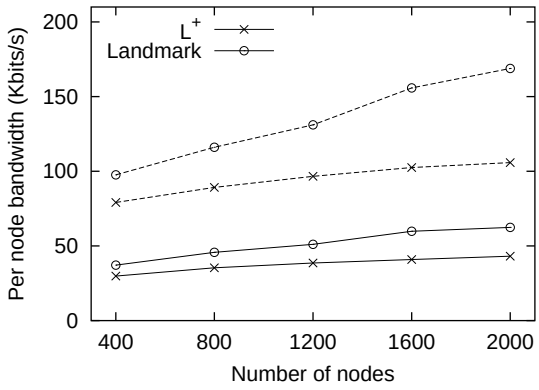

Figure 6: Per-node bandwidth as the number of nodes increases, for a lattice network with a local communication pattern. All packets were delivered without loss. The per-node cost of L^+ grows slowly compared to that of original Landmark.

5.2 L^+ Location Query Performance

Results in this section demonstrate how the query traffic generated by L^+ scales with network size, for a local communication pattern. Simulations in this section run in a static lattice network, where nodes are 150 meters apart. The local communication pattern is produced by each node choosing a destination between 1000 and 1400 meters away, out of a universe size of 3000 m^2 at 400 nodes to 6700 m^2 at 2000 nodes.

Figure 6 shows that the per-node communication cost of L^+ grows slowly compared to that of original Landmark, for local traffic. The results are consistent with the observation in Section 3.3 that Landmark scaling can be dominated by the need to send location updates and queries to random nodes, imposing $O(\sqrt{N})$ per-node communication costs. L^+ sends local queries for local communication, which would scale as

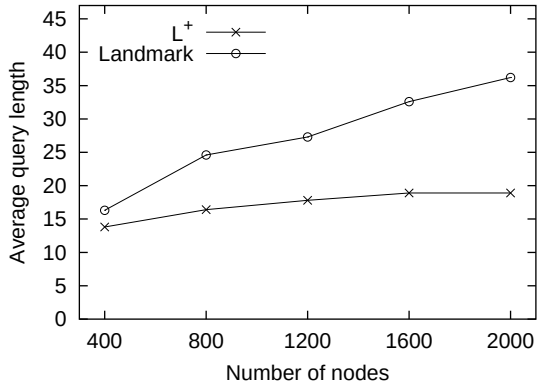

Figure 7: Average query length, in hops, as the number of nodes increases. The communication pattern is local; the number of hops between each source and destination pair is between 5 and 10 hops. Query length includes hop counts from failed queries as well as from successful queries.

Nodes	L2	L3	L4	L5
400	66.4 s	125.1 s	238.1 s	-
800	56.6 s	104.0 s	213.6 s	-
1200	54.9 s	98.0 s	163.8 s	309.9 s
1600	45.1 s	81.8 s	139.7 s	257.3 s

Table 1: Average intervals, in seconds, between updates that L^+ nodes send through various landmark levels. These follow a power law distribution, helping L^+ avoid non-scalable long-distance communication.

$O(1)$; however, it sends location updates to each level of the hierarchy at power-of-two intervals and has a $O(\log N)$ DV routing overhead, thus its per-node communication cost is $O(\log N)$.

Figure 7 illustrates the difference between L^+ and Landmark query traffic by showing query length in hops. Again, the fact that L^+ generates local queries for local communication means that its query lengths grow more slowly than those of Landmark.

5.3 Mobile Network Simulations

This section demonstrates that L^+ scales well in networks of mobile nodes. Experiments in this section use a random communication pattern, rather than the local pattern of the previous section.

Figure 8 shows that as the network size increases, the per-node bandwidth increases slowly under L^+ and Landmark. In contrast, per-node bandwidth increases linearly with DSR. The linear increase is caused by DSR flooding queries over the whole network, sometimes multiple times per connection as cached routes break due to node mobility.

L^+ and Landmark behave similarly in the mobile net-

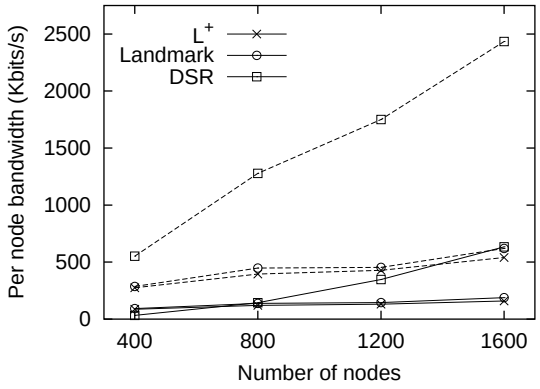

Figure 8: Per node bandwidth as the number of nodes increases, for mobile nodes. Dotted lines represent 99th percentiles; solid lines represent median values. Per-node bandwidths for L^+ and Landmark routing grow slowly as the number of nodes increases. In comparison, per-node bandwidth for DSR grows linearly.

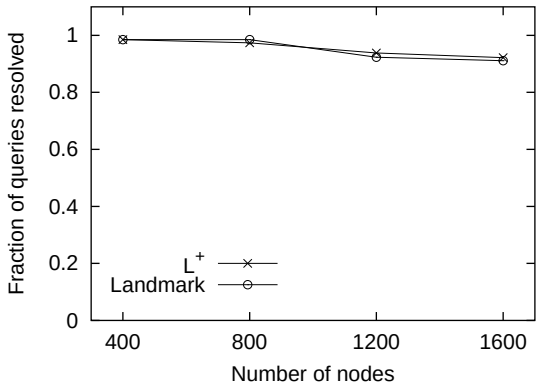

Figure 9: Query success rate as the number of nodes increases drops slightly under both L^+ and Landmark.

work. One reason is that the communication pattern is random, so the L^+ location service only benefits a small fraction of lookups. Another reason is that nodes following a random waypoint movement model tend to cluster near the center of the network. The high density at the center causes the DV broadcast packets to be large; these packets dominate per-node bandwidth, masking the savings from the L^+ location service.

Table 1 shows how often each node sends out location updates under L^+. Recall that in addition to periodic location updates, a node also sends location updates when it detects a change in its Landmark address. If the change occurred at the level l component of the address, the update is resolved through a level $l+1$ landmark. Table 1 confirms our belief that with this update scheme the update traffic follows a power-law distribution, since higher level components of a Land-

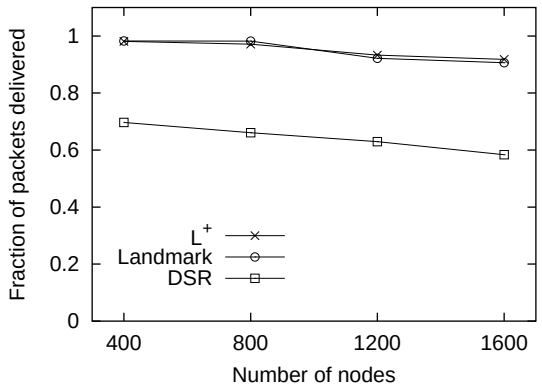

Figure 10: Fraction of packets delivered as the number of nodes increases. Both L^+ and Landmark routing deliver a high fraction of the packets as the number of nodes increases. Almost all the packet losses occur due to failed location queries. In comparison, DSR drops packets due to broken routes very early on.

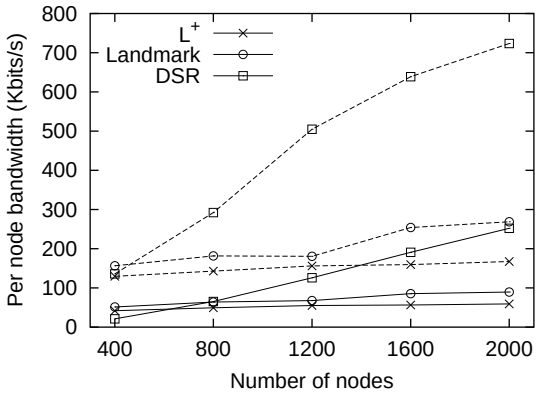

Figure 11: Per node bandwidth as the number of mobile clients increases for a rooftop network. The number of clients and number of Landmarks are the same. Landmarks are positioned in a static grid.

mark address are less likely to change with mobility.

Figure 9 shows that as the network size increases, the query success rate drops. Instability in the Landmark hierarchy (i.e. parent changes, nodes electing themselves to be landmarks, and nodes removing themselves as landmarks) cause most of the query failures. Hierarchy instability also increases location update frequencies. Figure 10 shows the fraction of data packets delivered; the fraction is high for L^+ and Landmark. The reason for the packet losses with DSR is that DSR is vulnerable to broken source routes due to mobility.

5.4 Rooftop Network Simulations

This section evaluates the performance of L^+ in a simulated rooftop network environment. The rooftop network contains both static and mobile nodes. The static nodes are laid out in a lattice formation with 150 meters between adjacent nodes. Only the static nodes participate in the Landmark hierarchy, and only the static nodes forward packets. The mobile nodes act as sources and sinks of data, but do not forward, participate in the Landmark hierarchy, or appear in the DSDV updates. The mobile nodes do send location updates and queries as described in Section 3. The Landmark address of a client is simply the Landmark address of the landmark closest to the client. When a landmark receives a packet with its address as the destination address, but with a different destination ID, it tries to send the packet to that node directly. If the transmit fails because the node has moved away, either the second Landmark address on the packet is used, or the packet is dropped.

In the following simulations, static nodes do not initiate data flows themselves. Communication between mobile nodes follows a local model: each sender sends packets to a

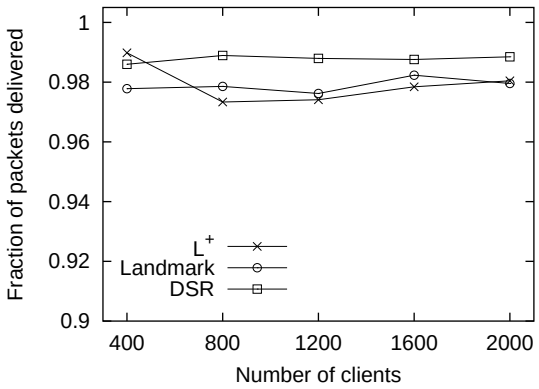

Figure 12: Fraction of packets delivered as the number of clients increases for a rooftop network. Landmarks are positioned in a static grid.

receiver between 800 and 1200 meters away at the time the communication starts. In each scenario there are equal number of static and mobile nodes. The density of mobile nodes is 10 nodes per radio range. The x-axis of each graph reflects the number of mobile nodes in the network.

Figure 11 shows that as the network size increases, the per-node bandwidth requirements of L^+ grow slowly. Its growth rate is consistent with the observation in Section 3.3 that L^+ has a per-node communication cost that is $O(\log N)$. L^+ scales slightly better than original Landmark, and substantially better than DSR. Again, DSR bandwidth requirements grow linearly due to global flooding of queries when source routes break due to mobility. The DSR curve starts to flatten after 1200 nodes because the implementation's maximum source route length is 32 hops, too small for the longest required paths in networks with 1600 and 2000 nodes. We were not able to run large simulations with higher maximum

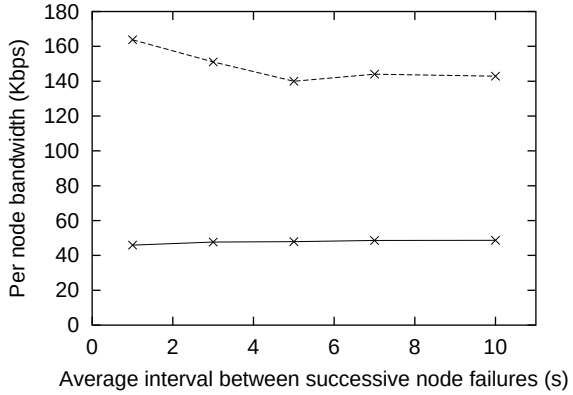

Figure 13: Per node bandwidth in an 800-node L^+ rooftop network as the average interval between successive node failure increases. Each node failure involves a randomly selected node dying for 120 seconds.

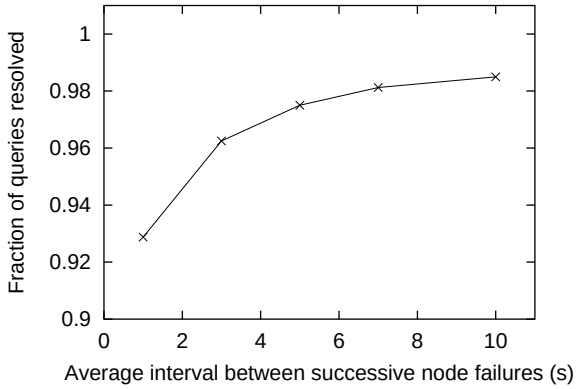

Figure 14: L^+ query success rate in an 800-node rooftop network as the average interval between successive node failure increases. Each node failure involves a randomly selected node dying for 120 seconds.

source route lengths due to memory constraints.

Figure 12 shows the packet delivery rates of L^+, Landmark, and DSR as the number of nodes increases. The delivery rates for all three protocols remain high even as the network size increases. DSR does not drop too many packets even though it has bad required bandwidth scaling because we are using 100 Mbps radios in these simulations. Both L^+ and Landmark have better deliver rate in rooftop networks than in mobile networks because the Landmark hierarchy, and thus the location service infrastructure, is fixed. Most packet drops by L^+ and Landmark, however, are still triggered by failed location queries.

If a client is moving away from the landmark that it had picked as a parent, it must wait until that landmark's information expires from its DV table before picking a new parent. If a client is moving slowly, there is a good chance that at least one of the two Landmark addresses it picked is current. However, if a client is moving quickly, it cannot pick new parents fast enough. This contributes to most of the query failures in the Landmark and L^+ simulations. This problem may be fixed by using a smaller timeout value for expiring entries from the DV table, or by using a more sensitive metric for picking parents, such as signal strength. With varying link conditions, however, these techniques may trigger unnecessary address changes.

Sometimes a sender in L^+ selects the wrong server to use at lower levels of the hierarchy. If the location servers corresponding to higher levels of the hierarchy contain stale addresses, L^+ counts on query refinement to forward the query to the destination. When a client is moving fast, query refinement does not work well: the node performing the query refinement and the client may obtain different results when they call `choose-landmark`. This contributes to the extra packet drops by L^+.

Figures 13 and 14 show the effect of node failures on the performance of L^+ in a rooftop network of 800 mobile and 800 static nodes. We model a node failure by turning off a randomly selected node for 120 seconds. Each node failure occurs at a randomly chosen time during the part of the simulation that involves communication (i.e. last 400 seconds). When a failed node turns back on, all entries in its DV table and most of the entries in its location database have expired. Unexpired entries contain stale addresses.

Figure 13 shows that as the frequency of node failure decreases, per-node bandwidth requirements decrease as well. When node failure is frequent, the hierarchy constantly changes, triggering large numbers of location updates. Figure 14 shows that as the frequency of node failure decreases, the query success rate increases. When node failure is frequent, hierarchy changes cause rapid changes in location server selection. These data show that L^+ can handle unstable nodes gracefully.

6 Related Work

Geographic Forwarding [5, 9, 13, 3] is the only other routing protocol we know of that scales well in large wireless ad hoc networks. In geographic forwarding, a node forwards a packet through the neighbor geographically closest to the packet's destination. Each node only needs to know the positions of its immediate neighbors; so the communication and storage costs scale as $O(1)$. Geographic forwarding, however, requires each node to know its geographic location, perhaps using the Global Positioning System (GPS); this is generally not convenient. Landmark and L^+ routing aren't quite as scalable as geographic forwarding; they have $O(\log N)$ costs. On the other hand, their use of a dynamically chosen hierarchy is likely to be more practical than dependence on GPS.

Geographic forwarding scales well, but requires that endpoints of conversations find each others' current geographic location. how a node acquires the geographic position of a destination. DREAM [2] nodes pro-actively flood position updates over the whole network, allowing other nodes to build complete position databases. LAR [10] nodes reactively flood position queries over the entire network when they wish to find the position of a destination. Both of these systems involve global flooding, making them best suited to small networks.

GLS is a scalable location service intended to track destination node positions in ad hoc networks using geographic forwarding [13]. Each node chooses its location servers from nodes positioned at exponentially increasing distances. Location updates use a scalable communication pattern that follows the power law [12]: frequency of updates lowers exponentially as the distance to the location server increases. Location queries mimic the actual communication pattern: queries for nearby nodes are answered by nearby location servers; whereas queries for distant nodes are answered by distant location servers. Landmark routing has an organization similar to that of GLS: both use a location service combined with a routing system that locations as addresses. Again, GLS (as well as geographic forwarding) depends on nodes having GPS receivers, or some equivalent. L^+ has similarly high scalability, but is self-contained, with no dependence on anything like GPS.

LANMAR [6] assumes that nodes move in predetermined groups, embeds a group number in each node's address, and runs an inter-group routing protocol as well as an intra-group protocol per group. This means that the amount of routing state per node is related to the number of groups plus the number of nodes in one group, rather than the total number of nodes. This scheme achieves good scaling in applications with natural group structure, but may not scale if nodes move predominantly as individuals instead of in groups.

A number of existing ad hoc routing algorithms make use of globally-distributed topology information. For example, AODV [18], DSR [4], DSDV [17], and TORA [15] flood routing queries or complete routing advertisements to the entire network. Route caching and local repair reduce the impact of this flooding, but there is a significant $O(N)$ per-node cost in these algorithms that does not scale well.

The Zone Routing Protocol (ZRP) [8] finds paths to nodes by globally flooding queries, but ensures that only one node in any given zone needs to process each query. A zone is defined as as certain radius of hops. Nodes accumulate a list of all the nodes in their zone using a limited-radius pro-active routing protocol, and use that list to answer queries. The global query flooding suggests that ZRP might have a $O(N)$ per-node communication cost component, which might scale badly in large networks.

Fisheye State Routing (FSR) [16] and Fuzzy Sighted Link State (FSLS) [20] are two protocols designed to scale to large networks by reducing the frequency and size of topology updates. Every node learns the topology of the entire network. In FSR, routing table entries are propagated with progressively decreasing frequency as the distances to these nodes increase. In FSLS, a node aggregates link changes and propagates them at frequencies that depend on how far the link changes occurred from itself. Nodes still need to keep a quantity of state that scales with the total size of the network, and that state may become stale if nodes move quickly; these factors may limit these algorithms ability to handle very large networks.

SCOUT [11] uses the Landmark hierarchy as a clustering technique for sensor networks. Each landmark propagates a summary of the objects its child sensors are monitoring. A remote sensor uses this summary to redirect queries to sensors with more detailed information about a particular object, until the query reaches the most relevant sensor.

7 Conclusion

This paper introduces L^+, a modified Landmark routing system designed for large ad hoc mobile networks. L^+ differs from Landmark mainly in its location service. Unlike Landmark, the number of hops each L^+ location query takes corresponds with the distance between the sender and the receiver. Hence, L^+ avoids global communication when the underlying traffic pattern is local. The location update traffic in L^+ also follows the power-law distribution, making it mostly local. Finally, L^+ modifies the Landmark hierarchy maintenance and routing algorithms to make them react better to mobility. Simulation results show that the per-node required bandwidth of L^+ grows slower than both Landmark and DSR. The results are consistent with our analysis that the per-node communication cost of L^+ is $O(\log N)$.

References

[1] Nokia Rooftop wireless routing system. `http://www.nwr.nokia.com`.

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

[3] L. Blazevic, L.Buttyan, S. Capkun, S. Giordano, J. Hubaux, and J.-Y. Le Boudec. Self-organization in mobile ad-hoc networks: the approach of Terminodes. *IEEE Communications Magazine*, 39(6), June 2001.

[4] J Broch, D Johnson, and D Maltz. The Dynamic Source Routing Protocol for mobile ad hoc networks. In *Internet draft, IETF Mobile Ad Hoc Networking Working Group*, December 1998.

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

[6] M. Gerla, X. Hong, and G. Pei. Landmark routing for large ad hoc wireless networks. In *Proceedings of IEEE GLOBECOM 2000*, San Francisco, CA, November 2000.

[7] P. Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[8] Zygmunt Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. IEEE ICUPC*, pages 562–566, October 1997.

[9] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, August 2000.

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

[11] S. Kumar, C. Alaettinoglu, and D. Estrin. Scalable object tracking through unattended techniques. Technical Report USC CS TR00-738, University of Southern California, 2000.

[12] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[13] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David Karger, and Robert Morris. A Scalable Location Service for Geographic Ad-Hoc Routing. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[14] ns Notes and Documentation. `http://www.isi.edu/vint/nsnam/`, 2000.

[15] Vincent Park and Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE Infocom*, pages 1405–1413, April 1997.

[16] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications*, pages 70–74, New Orleans, LA, June 2000.

[17] Charles Perkins. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the ACM SIGCOMM*, London, U.K., 1994.

[18] Charles Perkins and Elizabeth Royer. Ad hoc on-demand distance-vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[19] M. Ritter, R. Friday, R. Garces, W. San Filippo, C-T Nguyen, and A. Srivastava. Mobile connectivity protocols and throughput measurements in the Ricochet MicroCellular data network (MCDN) system. In *Proc. ACM/IEEE MobiCom*, pages 322–331, Rome, July 2001.

[20] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *Proceedings the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, California, October 2001.

[21] Paul Francis Tsuchiya. The Landmark hierarchy: description and analysis. Technical Report MTR-87W00152, MITRE Corporation, June 1987.

[22] Paul Francis Tsuchiya. The Landmark hierarchy: a new hierarchy for routing in very large networks. In *Proceedings of the ACM SIGCOMM*, pages 35–42, Stanford, CA, August 1988.

[23] Paul Francis Tsuchiya. Landmark routing: architecture, algorithms, and issues. Technical Report MTR-87W00174, MITRE Corporation, May 1988.

[24] Mark Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1991.