Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 51 (2007) 3693–3714

Computer Networks

www.elsevier.com/locate/comnet

On the scalability of rendezvous-based location services for geographic wireless ad hoc routing

Saumitra M. Das, Himabindu Pucha, Y. Charlie Hu [*]

Center for Wireless Systems and Applications in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, United States

Received 26 March 2006; received in revised form 15 November 2006; accepted 18 March 2007
Available online 6 April 2007

Responsible Editor: I. Stavrakakis

Abstract

Geographic routing protocols allow stateless routing by taking advantage of the location information of mobile nodes and thus are highly scalable. A central challenge in geographic routing protocols is the design of scalable distributed location services that track mobile node locations. A number of location services have been proposed, but little is known about the relative performance of these location services. In this paper, we perform a detailed performance comparison of three rendezvous-based location services that cover a range of design choices: a quorum-based protocol (XYLS) which disseminates each node's location to $O(\sqrt{N})$ nodes, a hierarchical protocol (GLS) which disseminates each node's location to $O(\log N)$ nodes, and a geographic hashing-based protocol (GHLS) which disseminates each node's location to O(1) nodes.

We present a quantitative model of protocol overheads for predicting the performance tradeoffs of the protocols for static networks. We then analyze the performance impact of mobility on these location services. Finally, we compare the performance of routing protocols equipped with the three location services with two topology-based routing protocols, AODV and DSR, for a wide range of network sizes. Our study demonstrates that when practical mobile ad hoc network (MANET) sizes are considered, the constants matter more than the asymptotic costs of location service protocols. In particular, while GLS scales better asymptotically, GHLS transmits fewer control packets and delivers more data packets than GLS in MANETs of sizes considered practical today and in the near future. Additionally, in contrast to the complex GLS design, the simplicity of GHLS provides significant resilience to performance degradation from mobility. Finally, although XYLS has a comparable packet delivery ratio to GHLS, it achieves this ratio with a higher overhead.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Geographic routing; Location services; MANETs; Scalability

1. Introduction

A mobile ad hoc network (MANET) consists of a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized

[*] Corresponding author. Tel.: +1 765 494 9143.
E-mail addresses: smdas@purdue.edu (S.M. Das), hpucha@purdue.edu (H. Pucha), ychu@purdue.edu (Y.C. Hu).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.03.006

administration. In such a network, each node operates not only as a host, but also as a router, forwarding packets for other mobile nodes.

A fundamental challenge in MANETs is the design of scalable and robust routing protocols. Existing routing protocols fall into two categories: (1) *Topology-based* routing protocols which assume no knowledge of the mobile nodes' positions. Examples include proactive protocols such as DSDV [24], reactive protocols such as DSR [12], AODV [25], and hybrid protocols such as ZRP [8]. These protocols rely on discovering and maintaining global states in order to route packets. As a result, they have limited scalability. (2) *Geographic* routing protocols that utilize the location information of nodes available from positioning systems such as GPS to forward packets. Since local information (neighbor's locations) is used for global routing, geographic routing protocols have the potential to scale to a larger number of nodes than topology-based protocols.

However, before routing a packet using a geographic routing protocol, the source node needs to retrieve the location of the destination node. As such, a central challenge in geographic routing protocols is the design of efficient location services that can track the locations of mobile nodes and reply to location queries for them. Such a service has to be scalable to preserve the scalability of geographic routing, and it itself should ideally operate using geographic routing. More specifically, a protocol that implements an effective location service has to exhibit the following essential properties:

- *Efficiency*: The protocol should transmit few packets in performing location update and query operations.
- *Robustness*: The service should not be disrupted by node mobility, for example, when nodes cross the boundary of any artificially imposed hierarchy of grids, or node (location server) failures or disconnection from the network.
- *Load balance*: The protocol should place balanced update and query load on the nodes in the network so that no single node becomes the bottleneck.
- *Scalability*: The protocol should maintain all of the above properties when the network scales to a large number of nodes.
- *Locality-awareness*: The distance of the location server from the querying source should not be much larger than the distance between the source and destination node. In other words, location queries for nodes nearby should not travel far away in the network. However, the benefit of this feature relies on the presence of local data traffic patterns.

Numerous protocols for location services [17,1,20,7,32,6,31,35,21,36,9,5] have been proposed to solve the location tracking and retrieval problem in geographic routing for MANETs. Although each protocol is proposed along with some theoretical and/or simulation analysis, little is known about the relative performance of these protocols, especially in a realistic setting, i.e., a mobile environment.

This paper is the first to provide a realistic, quantitative analysis comparing the performance of a variety of scalable location service protocols. We focus on rendezvous-based location services as they are inherently more scalable than flooding-based ones. We perform an in-depth performance comparison of three representative rendezvous-based location services which cover a wide range of design choices: a quorum-based protocol (XYLS) which disseminates each node's location to $O(\sqrt{N})$ nodes, a hierarchical hashing-based protocol (GLS) which disseminates each node's location to $O(\log N)$ nodes, and a geographic hashing-based protocol (GHLS) which disseminates each node's location to $O(1)$ nodes. The major contributions of this paper are:

- We present the first detailed comparison of three scalable location services that focuses on design tradeoffs rather than protocol nuances.
- We present a quantitative model for measuring location service overhead and verify its accuracy in predicting the performance tradeoffs of different design approaches for static networks.
- Since the quantitative model cannot predict the impact of node mobility and MAC layer interference, we perform a simulation study of all three location services in a detailed simulator and analyze the performance impact of mobility on all three location services.
- We compare the performance of geographic forwarding when equipped with the three location services with two topology-based routing protocols, AODV [25] and DSR [12], for a wide range of network sizes to characterize the applicability of geographic routing protocols.

The key insights from our study are: (1) when practical MANET sizes are considered, i.e., up to a few thousand nodes, the constants matter more than the asymptotic costs of location service protocols, and (2) simplicity of design provides resilience to performance degradation from mobility by reducing the opportunities for routing state inconsistency.

The major findings of our performance study are:

- For static networks, although GLS makes a large number of short distance updates and a small number of large distance updates, our model shows that it has lower update overhead than GHLS only in large networks of more than 25,000 nodes. In a 600-node static network, GLS incurs 3 times higher update overhead than GHLS. GLS has higher query overhead disregarding the network size, from always forwarding queries to the destination nodes.
- Simulations of a 600-node network with and without node mobility show that node mobility has a much higher impact on the protocol overhead of GLS than on those of GHLS and XYLS. The higher overhead increase comes from handling inconsistencies caused by maintaining consistent hashing in the grid hierarchy.
- In simulations of mobile networks of up to 2000 nodes, GHLS provides a more robust and efficient location service than GLS or XYLS.
- For small to medium sized networks (up to 400 nodes), reactive non-geographic routing protocols like AODV can outperform geographic routing protocols based on location services like GLS which incurs higher overhead in sending proactive beaconing packets and location lookups. Local repair and hop-by-hop routing state in AODV contributes to it usability in such networks, whereas DSR suffers due to a lack of intermediate repair. However, GHLS enables a geographic routing protocol that is comparable to reactive protocols in not only packet delivery but also routing overhead in small to medium sized networks.

The rest of the paper is organized as follows. Section 2 presents a taxonomy of location services. Section 3 describes the three location services considered in our study. Section 4 presents an analysis of the overhead of the three protocols. Section 5 describes the simulation methodology. Section 6 presents the performance comparison of the three location services, and Section 7 presents the scalability of geographic routing using these location services. Section 8 discusses related work and Section 9 summarizes the paper's contributions.

2. A taxonomy of location services

Fig. 1 depicts a taxonomy of the location services proposed so far. At the top level, location services can be divided into *flooding-based* and *rendezvous-based* approaches. Flooding-based protocols can be further divided into proactive and reactive approaches. In the *proactive flooding-based* approach, each (destination) node periodically floods its location to other nodes in the network each of which maintains a location table recording the most recent locations of other nodes. The interval and range of such flooding can be optimized according to the node's mobility and the "distance effect". For example, flooding should be more frequent for nodes with higher mobility, and flooding to faraway nodes can be less frequent than to nearby nodes. DREAM [1] serves as a good example of proactive flooding-based location services. In *reactive flooding-based* approaches [17], if a node cannot find a recent location of a destination to which it is trying to send data packets, it floods a scoped query in the network in search of the destination. The previous location of the destination node can be used to narrow the scope of the flooding.

In rendezvous-based protocols, all nodes (potential senders or receivers) in the network agree upon a mapping that maps each node's unique identifier to one or more other nodes in the network. The mapped-to nodes are the *location servers* for that node. They will be the rendezvous nodes where periodical location updates will be stored and location queries will be looked up. Two different approaches of performing the mapping, *quorum-based* and *hashing-based*, have been proposed. In the *quorum-based* approach [7], each location update of a node is sent to a subset (update quorum) of available nodes, and a location query for that node is sent to a potentially different subset (query quorum). The two subsets are designed such that their intersection is non-empty, and thus the query will be satisfied by some

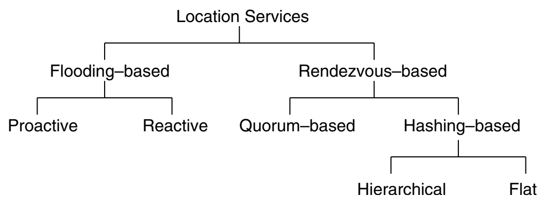

Fig. 1. A taxonomy of location services.

node in the update quorum. Several methods on how to generate quorum systems have been discussed in [7]. In this paper, we choose a design that is conceptually simple and efficient in terms of update and query complexities. In the so-called column–row quorum-based protocol [32], the location of each node is propagated in the north–south direction, while any location queries are propagated in the east–west direction. Effectively, each node's location is disseminated to $O(\sqrt{N})$ location servers.

In *hashing-based* protocols, location servers are chosen via a hashing function, either in the node identifier space or in the location space. Hashing-based protocols can be further divided into hierarchical or flat, depending on whether a hierarchy of recursively defined subareas are used. In *hierarchical hashing-based* protocols (for example, [21,36]), the area in which nodes reside is recursively divided into a hierarchy of smaller and smaller grids. For each node, one or more nodes in each grid at each level of the hierarchy are chosen as its location servers. Location updates and queries traverse up and down the hierarchy. A major benefit of maintaining a hierarchy is that when the source and destination nodes are nearby, the query traversal is limited to the lower levels of the hierarchy. Since the height of the hierarchy is $O(\log N)$, effectively each node's location is disseminated to $O(\log N)$ location servers. The best example of hierarchical rendezvous-based location service is GLS [21], which is chosen in the comparative study in this paper.

In *flat hashing-based* protocols (for example, [6,31,35]), a well-known hash function is used to map each node's identifier to a *home region* consisting of one or more nodes within a fixed location in the network area. All nodes in the home region maintain the location information for the node and can reply to queries for that node; they serve as its location servers. Typically, the number of location servers in the home region is independent of the total number of nodes in the network, and thus effectively each node's location is disseminated to $O(1)$ location servers. In this paper, we propose a flat hashing-based protocol called Geographic Hashing Location Service (GHLS) that pushes the concept of geographic hashing to the extreme: the home region consists of only one node – the node whose location is closest to the hashed location. We use it as the representative of the flat hashing-based protocols.

Since flooding-based protocols scale poorly with the network size, we primarily focus on rendezvous-based protocols in the rest of the paper. However, we have included comparisons with some sample flooding based approaches in Section 6.5 to provide a frame of reference for understanding the performance of the studied protocols.

3. Location service protocol descriptions

In this section, we describe geographic forwarding and the three representative rendezvous-based location services under study, along with implementation decisions made in our comparison study. For each protocol, we describe how it (1) chooses the location servers, (2) performs a location update, and (3) performs a location query.

3.1. Geographic forwarding

All of the location services in our study are implemented by leveraging geographic forwarding.

(a) *Basic operations*: Each node determines its own geographic location – latitude and longitude – with the assistance of some global positioning systems such as GPS [34]. Each node periodically announces its IP address and location to its one-hop (within the radio transmission range) neighbors by broadcasting "beacon" packets. Each beacon contains a node's IP address and current location. Each node maintains the IP and location information of its neighbors. Subsequently, a packet can be routed to a destination node using this local state at each node. Each packet contains the destination address in the IP header and the destination's location (x- and y-coordinates) in an IP option header. To forward a packet, a node consults its neighbor table and forwards the packet to its neighbor closest in geographic distance to the destination's location.

Under the above greedy geographic forwarding, forwarded packets may potentially reach a node that does not know any other node closer to the destination than itself. This indicates a hole in the geographical distribution of nodes. Recovering from holes is possible using face routing [2,15,19].

All the location services are evaluated using the same greedy geographic forwarding protocol described above. This enables a fair comparison based only upon the features of the location services.

(b) *Optimizations*: The following optimizations are incorporated into our basic geographic forwarding implementation.

- *Packet salvaging*: Whenever the MAC layer is unable to deliver a packet, it performs a callback to the network layer. The entry in the neighbor table corresponding to the unreachable one-hop neighbor is subsequently removed and the packet and all other such packet in the interface queue are salvaged by forwarding them to one of the remaining nodes in the neighbor table that is closest to the destination of the packets.
- *Two-hop beaconing*: To reduce the probability of encountering a routing hole, i.e., when node cannot find a node closer than itself to the destination and the destination is not its one hop neighbor, two-hop beaconing is used. In addition to its own location, every node when beaconing sends a list of its one-hop neighbors and their locations as well. When forwarding a packet, a list of potential next hops are constructed from the table of neighbors both one and two hops away. If the node closest to the destination is a two-hop neighbor, the one-hop neighbor closest to that node is sent the packet. If the node closest to the destination is a one-hop neighbor, the node is directly sent the packet. This two-hop beaconing is a integral part of GLS [21].

We now describe the three major location services studied in this paper.

3.2. Column–row location service (XYLS)

XYLS is a proactive location service that disseminates locations in a direction such that a query can intersect it subsequently. It is similar to the column–row quorum-based location service originally proposed in [32].

1. *Selecting location servers*: At any given point of time, for each destination node, the nodes that are located along the north–south direction in the geographic area form the *update quorum* (location servers). Similarly, for each source node, the nodes that are located along the east–west direction form the *query quorum* for the node. Thus each query is expected to intersect one of the location servers.

2. *Performing updates*: When a node decides a location update is needed, it propagates the location update along the north–south direction, i.e., with the goal of reaching all the nodes in the same column in the geographic area. The thickness of the column controls the tradeoff between the update overhead and the robustness of the location service: the thicker the column, the higher the update overhead, but also the more likely a query will be satisfied. Our implementation constructs sufficiently thick columns using a low overhead approach: Each node selected as a location server in the north or south direction broadcasts the update to its one hop neighbors in addition to unicasting it to the next location server in the update direction. This increases the thickness of the column. An update is sent after every movement of distance d (update threshold) or when the timeout window expires first due to the change of speed. To facilitate caching of the location by location servers and forwarding nodes, each update packet has a timeout window which specifies the validity of the update. The timeout window is predicted as the time taken to travel the distance d based on the current mobility of the node. An update is cached for the timeout period by all forwarding nodes and stored for the timeout period by the location server.

3. *Performing queries*: When a source node initiates a location request for a destination node, the query is propagated along the east–west direction, i.e., along its row of nodes in the geographic area. The query contains the time of the most recent location of the destination known to the source. If a node along the row has a cached location for the destination node that is more recent than the time in the query, it sends a reply packet via geographic forwarding back to the source.

4. *Routing structures*: XYLS maintains three tables to provide its distributed location service. A *neighbor table* is used to maintain the current neighbors of a node along with their locations. A *location table* holds the node's portion of the distributed location database. Each entry consists of a node's ID, its geographic location and the timeout associated with the reported location. A *location cache* similar in structure to the location table is used to store the location information learned by a node while forwarding update packets.

3.3. Grid location service (GLS)

GLS [21] is a hierarchical hashing-based location service. It is motivated by consistent hashing [14] and performs hashing in the node identifier space, as opposed to the geographic location space. The geographic area in which the nodes reside is recursively divided into a hierarchy of smaller and smaller squares, and the smallest square is referred to as an order-1 square. In GLS, the side length of the smallest square is chosen to be the transmission range of 802.11, i.e., 250 m. The four order-($n-1$) squares sharing the same order-n square as the parent square are *sibling* squares. For each node, one node from each sibling square of the node's square at each hierarchy level is selected as its location server. The geographic forwarding for GLS uses the two-hop beaconing protocol as described in Section 3.1. This ensures that each node knows the location of all nodes in its order-2 square. Fig. 2 (from [21]) shows an example of a hierarchy. GLS assigns a unique, random ID to each node, by applying a strong hash function to the node's *unique name*. The unique name could be a host name, IP address, or MAC address. In the circular identifier (ID) space, the node *closest* to a node X is defined as the node with the least ID greater than X. For each node B, GLS selects a location server in each sibling square at each hierarchy level as the node whose ID is closest to node B in that square. In Fig. 2, the location servers at B's order-1 sibling squares are nodes 2, 23, 63, respectively. In B's order-2 sibling squares, the location servers are 26, 31, and 43, respectively.

1. *Performing queries*: We first describe how a location query is routed, assuming all nodes have updated their location servers of their identifiers and locations. At each step, a query makes its way to the closest node at successively higher hierarchical levels. If node A (76) requests for the location of node B (17), it looks up the same order-1 square for the best node, in this case, itself. It then looks for the best node in each of its order-1 sibling squares, in this case, node 21 which it knows about from two-hop beaconing. In the next step, node 21 finds node 20 which is the best node in node A's order-2 sibling squares, and node 20 has the location of node B since it is one of node B's location servers in B's order-3 sibling squares. Recall 21 is the best node in its own order-2 square. This ensures that no nodes in that square have IDs between 17 and 21. Now consider a node X in 21's sibling order-2 squares. If X's ID is between 17 and 21 then X must have chosen node 21 as a location server since there are no other nodes closer to X than 21 in 21's order-2 square. Thus node 21 knows all nodes in its order-2 sibling squares that lie between 17 and itself, including the minimum such node, which is 20 in this case. Similarly, at the next step, node 20 knows all nodes in its order-3 sibling squares between 17 and itself. In other words, node 20 is a location server for such nodes, including node 17.

Once the query reaches node 20, it forwards the query to the destination node 17 so that the destination node can reply to the source using geographic forwarding with its current location. This detour to the destination node is necessary in GLS due to the distance effect in location updates as explained below. Note that a query or a query reply stays inside the smallest square containing the source and the destination.

2. *Performing updates*: Similarly as how queries are routed, a node X inserts location updates to all of its location servers, i.e., the node whose ID is closest to X in each sibling square at each level, without knowing who the location servers are. Since there are 3 sibling squares and thus 3 location servers at each level of the hierarchy, and the hierarchy height is $O(\log_4 N)$, the total number of locations server per node is $O(3\log_4 N)$. Each update packet from a node carries its ID, its current location, and a timeout window explained below. Similarly as in XYLS, updates are cached at forwarding nodes.

3. *Location update interval*: The location update packets for a node are sent out at a rate proportional to the node's speed, and the range of the updates takes into account the distance effect [1]. A node updates its location servers in order-i sibling squares after each movement of $2^{i-1}\cdot d$, where d is the update threshold. Similarly as in XYLS, a timeout is advertised except that the timeout calculated is based on the order of the location server to be updated (e.g., for location servers in order-2 sibling squares, the timeout is based on the time taken to travel $2\cdot d$ distance). Since a node updates its high-order servers much less often, those servers can have fairly inaccurate location information, and thus they need to forward a query to the desti-

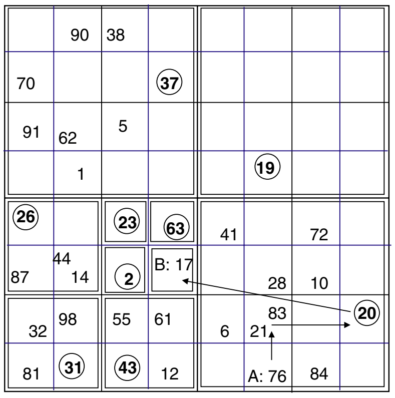

Fig. 2. Location update and query in GLS.

nation node which will then reply to the querying node with its current location.

4. *Routing structures*: GLS maintains three tables to provide its distributed location service. A *neighbor table* is used to maintain the node's current neighbors and their neighbors along with their locations. A *location table* holds the node's portion of the distributed location database. Each entry consists of a node's ID, its geographic location and the timeout associated with the reported location. A *location cache* similar in structure to the location table is used to store the location information learned by a node while forwarding update packets.

5. *Impact of grid boundaries*: The existence of a grid cell based structure has implications to GLS updates and queries. When a node B moves into a new grid cell, it has to update a subset of its location servers since *B*'s position in the hierarchy has changed. Additionally, since B may be acting as a location server for other nodes, there may now be no location server for those nodes in the grid cell B just left.

To find a node that has just crossed a grid boundary, GLS uses a "forwarding pointer" technique which requires node B to broadcast a forwarding pointer message (containing its new location) on entry to a new grid cell. Nodes in the previous level-0 cell propagate this forwarding entry to their neighbors during the regular beaconing process.

6. *Implementation changes*: The ns-2 implementation of GLS provided by its authors [21] was faithfully ported to Glomosim [38]. To confirm the correctness of the porting, we ran simulations with the same parameters using the original ns-2 code and using our Glomosim implementation, and the results from the two implementations matched closely.

3.4. Geographic hashing location service (GHLS)

Compared to hierarchical hashing-based protocols, flat hashing-based protocols avoid the complexity of maintaining a hierarchy of grids and the consequent maintenance due to nodes moving across grid boundaries. Previous flat hashing-based protocols, however, still introduce a geographic region (either a rectangle or a circle surrounding the hashed location) as the home region for the location servers for each node. In this paper, we propose a flat hashing-based protocol called GHLS that pushes the concept of geographic hashing to the extreme: the home region consists of only one node – the node whose location is closest to the hashed location.

1. *Selecting location servers*: In GHLS, each node is assigned a single location server, by hashing the node's *unique name* into a location in the geographic area, and designating the node whose location is currently closest to the hashed location as the node's location server. The unique name could be a host name, IP address, or MAC address. Thus, each node stores its location on only one location server, although a node may be a location server for zero or more nodes.

2. *Performing updates*: Similarly as in GLS, the updates of the nodes are reactive to their movement patterns and cached at forwarding nodes. A key difference between the update protocols for GLS and GHLS is that GHLS does not have multiple timeout windows and update frequencies for multiple location servers since there is only one location server per node.

3. *Performing queries*: To look up the location for a destination node, a sending node hashes the node's unique name using the same hashing function to generate the location, and sends a query packet which will be routed towards the server location via geographic forwarding. Upon reaching the location server, the stored location information is sent in a query reply packet from the location server back to the source of the query packet, again via geographic forwarding.

4. *Decoupling server and destination movement*: Due to node mobility, location information stored on a server may need to be migrated to other nodes that become closer to the location. One way to achieve this is to rely on a new update packet to arrive from the destination node periodically to refresh this state. This is difficult as it requires each node to adapt its update frequency to the movement of its location server. GHLS solves this problem by decoupling the movement of location servers from the movement of the originators of the updates via a lightweight *handoff* procedure. Every node that acts as a location server executes the following procedure when it receives a beacon packet from another node: it checks whether the source of beacon packet is closer to the hashed location of each of the sources for whom it is acting as the location server. If the source of the beacon packet is a better match for a subset of locations it stores, the node hands off these location information to the source of the beacon packet.

5. *Incorporating locality*: A potential drawback of flat geographic hashing protocols is that a location server can potentially be far away from both the

source and destination nodes, causing update and query operations to incur high overhead. To alleviate this problem, GHLS uses a hashing function that generates locations within a small region near the center of the whole region. We define the ratio of the side length of the scaled location server region and that of the whole area as the *scaling factor* α, and denote the scaled location server region as the α *region*. Fig. 3 shows an example of a scaled location server region.

Using a scaled location server region can potentially create service load imbalance among the nodes in the whole network, i.e., higher load in the scaled region. We evaluate and compare the load balance in all three location services in our simulation study in Section 6.

6. *Handling node failure*: Consider a node X that is currently a location server in GHLS. If X fails, queries that arrive between the time that X fails and the next updates from nodes whose locations are stored at X could potentially fail. However, caching of updates allow other nodes to answer queries despite failure of the location server. Additionally, the locations stored by X can also be piggy-backed in the beacon packets and stored at all the neighbors of X for added reliability.

7. *Routing structures*: GHLS maintains three tables to provide its distributed location service. A *neighbor table* is used to maintain the current neighbors of a node along with their locations. A *location table* holds the node's portion of the distributed location database. Each entry consists of a node's ID, its geographic location and the timeout associated with the reported location. A *location cache* similar in structure to the location table is used to store the location information learned by a node while forwarding update packets.

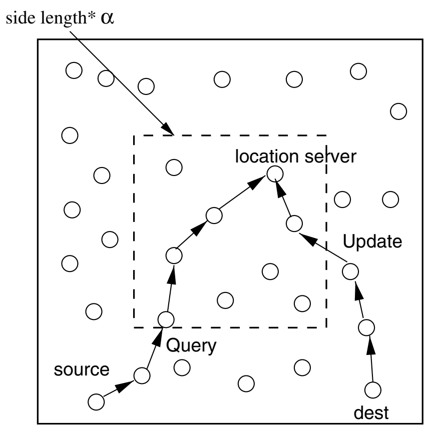

Fig. 3. Location update and query in GHLS. The location servers are inside the scaled location server region, surrounded by the dashed-line box.

4. Analysis

In this section, we analyze the performance of the three location services under study. The main results are summarized in Table 1.

4.1. Analysis for static networks

We first compare the amount of location service protocol packets initiated and transmitted in the three location services. The total number of location service protocol packets transmitted is referred to as the *LS protocol overhead*. To make the analysis tractable, we assume the network is static, the nodes are uniformly distributed in the geographic area, and location updates are not cached at forwarding nodes. To aid our analysis, we define the following parameters: Let the update frequency be f updates per second, the side length of order-1 square in GLS be l, and the height of the hierarchy be h, i.e., the whole area is an order-h square. Thus the side length of order-i square will be $2^{i-1} \cdot l$, and the side length of the whole area is $2^{h-1} \cdot l$. In GLS, l is chosen to be the transmission range of 802.11, i.e., 250 m. In the following, we use this assumption as well as the notion of order-1 squares to simplify our analysis of all three protocols, even though the notion of order-1 squares does not exist in XYLS and GHLS.

In XYLS, one update is sent at each update interval. The update is propagated along the column, which covers 2^{h-1} order-1 squares. Thus in theory, an update costs 2^{h-1} transmissions. However, since our implementation of XYLS update (or query)

Table 1
Comparison of the three location services

Metrics	XYLS	GLS	GHLS
Type	Quorum	Hierarchical hashing	Flat hashing
Updates per interval	1	$3 \cdot (2 - \frac{1}{2}^{h-2})$	*1*
Average update path length	2^{h+1}	$\frac{3}{2} \cdot (2 + \sqrt{2}) \cdot (h - 1)$	$\frac{1}{3} 2^{h-1} \sqrt{2}$
Average query path length	$\frac{2}{3} \cdot 2^{h}$	$\frac{2}{3} 2^{h-1} \sqrt{2}$	$\frac{1}{3} 2^{h-1} \sqrt{2}$
Average query rep.path length	$\frac{1}{3} \cdot 2^{h-1}$	$\frac{1}{3} 2^{h-1} \sqrt{2}$	$\frac{1}{3} 2^{h-1} \sqrt{2}$
Robustness	High	Medium	High

chooses the node closest to the starting node's x-coordinate (or y-coordinate), the average stride each hop makes is about half (the theoretically exact number is $\frac{4}{3\pi}$) of the transmission range, or half the side length of an order-1 square. Thus an update travels about 2^h hops in the north–south direction. Since a unicast and a broadcast are performed at each hop, the total number of transmissions is 2^{h+1}. For each query, two packets are sent along the two directions, only unicast is performed at each hop, and one of the two packets will not be forwarded further when it reaches a location server. The expected number of transmissions per query is $\frac{2}{3} \cdot 2^h$ (via simple integration). A query reply travels about $\frac{1}{3} \cdot 2^{h-1}$ hops on average since it goes along one direction towards the source.

In GLS, at each update interval, 3 updates are sent to the 3 location servers in order-1 sibling squares. At every two update intervals, 3 updates are sent to the 3 location servers in order-2 sibling squares, and so on. At every 2^{h-2} update intervals, 3 updates are sent to the 3 location servers in order-$(h-1)$ sibling squares. Thus the average number of update packets per interval is

$$L_{avg\_update\_pkts} = 3 \cdot 1 + \frac{1}{2} \cdot 3 + \frac{1}{4} \cdot 3 + \cdots + \frac{1}{2^{h-2}} \cdot 3 = 3 \cdot \left(2 - \frac{1}{2}^{h-2}\right). \qquad (1)$$

The number of hops traveled by the update packets to the three order-1 sibling squares are 1, 1, and $\sqrt{2}$, approximately, and the number of hops traveled between larger squares grows proportionally to the side lengths of the squares. Thus, the total number of hops traveled by an update packet sent to 3 order-k sibling squares is

$$L_{update\_hops}(k) = \left(2 + \sqrt{2}\right) \cdot 2^{k-1} \cdot \frac{3}{2}. \qquad (2)$$

The factor of $\frac{3}{2}$ for each term is explained as follows. Let the lowest-order sibling squares containing the destination (whose location is being inserted) and the location server be order-k squares X and Y, respectively. The update packet is first forwarded in a straight line towards Y while inside X. Once crossing the boundary between X and Y, it is forwarded in a sequence of steps in traversing up the hierarchy starting from an order-1 square and reaching the order-$(k-1)$ square containing the location server. Since the sequence of steps are directionless and the expected distances they travel are recursively doubled, they incur an approximate factor of 2 overhead compared to the direct distance between the starting point and finishing point inside square Y. The factor of 2 overhead while traversing inside Y and the straight line path inside X together contribute to the factor of $\frac{3}{2}$ for each term in the above hop count formula for update packets.

Note that from Eq. (2), the average path length of an update to an order-k sibling square is $\frac{1}{3}L_{update\_hops}(k)$.

From Eqs. (1) and (2), the average of number of hops traveled by all update packets is

$$\begin{aligned} L_{total\_update\_hops} &= L_{update\_hops}(1) + \frac{1}{2} \cdot L_{update\_hops}(2) \\ &\quad + \frac{1}{4} \cdot L_{update\_hops}(3) + \cdots \\ &\quad + \frac{1}{2^{h-2}} \cdot L_{update\_hops}(h-1) \\ &= \left(2+\sqrt{2}\right) \cdot \frac{3}{2} + \frac{1}{2} \cdot \left(2+\sqrt{2}\right) \cdot 2 \cdot \frac{3}{2} \\ &\quad + \frac{1}{4} \cdot \left(2+\sqrt{2}\right) \cdot 2^2 \cdot \frac{3}{2} + \cdots \\ &\quad + \frac{1}{2^{h-2}} \cdot \left(2+\sqrt{2}\right) \cdot 2^{h-2} \cdot \frac{3}{2} \\ &= \frac{3}{2} \cdot \left(2+\sqrt{2}\right) \cdot (h-1). \end{aligned} \qquad (3)$$

In GLS, the average number of hops queries travel can be approximated as twice the average distance between two random points in a square of area $2^{h-1} \times 2^{h-1}$, which is about $\frac{2}{3}2^{h-1}\sqrt{2}$. The factor of two comes from the fact that the location server has to forward the query to the destination node. Similarly, query reply takes an average $\frac{1}{3}2^{h-1}\sqrt{2}$ hops as it travels from destination to the source of the query.

In GHLS, one update is sent at each update interval. Assume the location server region is not scaled down, i.e., $\alpha = 1$. Again, the average number of hops it travels is the same as the average distance between two random points in a square of area $2^{h-1} \times 2^{h-1}$, which is about $\frac{1}{3}2^{h-1}\sqrt{2}$. Similarly, the average number of hops a random query or query reply travels is $\frac{1}{3}2^{h-1}\sqrt{2}$.

The following conclusions can be drawn from the above analysis (summarized in Table 1): (1) GLS loses to GHLS by a factor of 2 and outperforms XYLS by a factor of $\sqrt{2}$ in average query path length. (2) GLS and GHLS have comparable query reply path lengths which are a factor of $\sqrt{2}$ times that of XYLS. (3) The average update path length for GHLS and XYLS grows as $O(2^h)$ while that of

GLS is O(h). Thus, theoretically, GLS should outperform XYLS and GHLS in average update path length as the network is scaled. To estimate the network size beyond which GLS outperforms GHLS and XYLS, in Fig. 4, we plot the formulas for the average update path length for all three protocols as a function of the grid hierarchy height h and the network size N assuming an adequate density of 100 nodes/km^2. The plots show that while GLS outperforms XYLS for all network sizes, it has a longer average path length than GHLS until the height of the hierarchy reaches beyond 7, i.e., when the network size reaches 25,600. It is not clear whether ad hoc networks of such sizes can be realized or have applicability in most practical scenarios.

1. *Example analysis*: The formulas derived above can be used to estimate the overall protocol overhead of each of the three location services given a specific network scenario. We consider a network of 600 nodes and predict the number of update, query and query reply packets for each of the three protocols for a static network as shown in Table 2. The parameters used for the calculations are based on the simulation scenario used in the evaluation of GLS [21]. A network size of 600 nodes is the largest network size chosen in [21]. Although the 600 nodes are distributed in an area of 2500 m × 2500 m, GLS pads the area to 4000 m × 4000 m in order to generate a perfect grid with the height of hierarchy h being 5. The simulation duration (S) is chosen as 300 s. Since the network is static, the frequency of updates (f) is chosen to be once every 40 s. The number of queries (Q) initiated depends upon the traffic pattern used in the simulation and also the extent of local caching at the source node under each location service. We have borrowed values for Q from a simulation of the three location services with the same setup as above. Details of this simulation are outlined in Section 6. Q is different in the three protocols because they have different amount of local caching effects. For GHLS, we have chosen an α value of 1. This version is referred to as GHLS-1 in Table 2.

Table 2 compares the relative performance of the three location services. Each update interval is counted as one update event, and the number of actual update packets sent out is analyzed according to Table 1. The additional N update events for every location service protocol correspond to the one update sent by each node at startup time. "Txed" stands for the number of transmissions; one packet traversing one hop is counted as one transmission.

The following conclusions can be drawn from the comparison. In a static network of 600 nodes, GLS

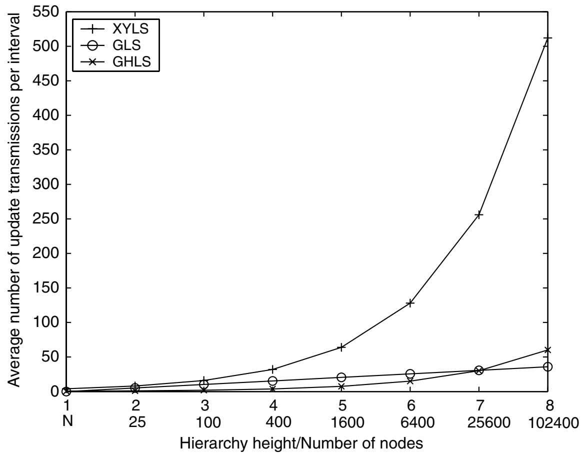

Fig. 4. Average update path length as a function of the hierarchy height.

Table 2
Overhead breakdown of location services for a static network of 600 nodes

	XYLS	GLS	GHLS-1
Updates			
Events	$E = S \cdot f \cdot N + N$ $= 300 \cdot \frac{1}{4} \cdot 600 + 600 = 4700$	$E = S \cdot f \cdot N + N$ $= 300 \cdot \frac{1}{4} \cdot 600 + 600 = 4700$	$E = S \cdot f \cdot N + N$ $= 300 \cdot \frac{1}{4} \cdot 600 + 600 = 4700$
Sent	$4 \cdot E = 18{,}800$	$E \cdot 3 \cdot (2 - \frac{1}{2}^{h-2}) = 26437$	$E = 4700$
Txed	$E \cdot 2^{h+1} = 300800$	$E \cdot \frac{3}{2} \cdot (2 + \sqrt{2}) \cdot (h-1) = 96,162$	$E \cdot \frac{1}{3} 2^{h-1} \sqrt{2} = 35,344$
Queries			
Sent	$Q = 7184$	$Q = 8055$	$Q = 8578$
Txed	$Q \cdot \frac{2}{3} \cdot 2^h = 153258$	$Q \cdot \frac{2}{3} 2^{h-1} \sqrt{2} = 121,147$	$Q \cdot \frac{1}{3} 2^{h-1} \sqrt{2} = 64,506$
Replies			
Sent	$Q = 7184$	$Q = 8055$	$Q = 8578$
Txed	$Q \cdot \frac{1}{3} \cdot 2^{h-1} = 38314$	$Q \cdot \frac{1}{3} 2^{h-1} \sqrt{2} = 60,573$	$Q \cdot \frac{1}{3} 2^{h-1} \sqrt{2} = 64,506$

outperforms XYLS in terms of overhead. However, the number of updates transmitted for GLS is 2.7 times more than the number of transmitted updates in GHLS. Also, the number of queries transmitted for GLS is larger than for GHLS as predicted by the analysis (twice as large). Thus, GLS loses to GHLS in terms of protocol overhead at a network size of 600 nodes. To verify the model, we also compare the predicted values with numbers obtained from a real simulation in Section 6.

4.2. Effect of mobility

We qualitatively compare the impact of mobility on the performance of the three location service protocols. In all three approaches, each node in its role as a destination node needs to send a new update whenever it moves the threshold distance, or when the timeout window expires first due to the change of speed. These events result in the corresponding increase or decrease in the amount of protocol packets and transmissions.

In XYLS, mobility has no additional effect on the communication complexity. In GHLS, a node in its role as a location server needs to hand off its local location database to neighboring nodes. In GLS, because the hashing is in the node ID space, there is no need for handoff. However, since the hashing is performed individually for each sibling square at each level of the hierarchy, every time a node crosses a grid boundary, it has to update some of its location servers accordingly. In the worst case, when a node crosses a boundary between two highest level grids, it has to select and update all of its $O(\log N)$ location servers. Furthermore, to deal with temporarily stale location information in high-order location servers, a node (as a location server) has to leave "forwarding pointers" via broadcasts to nodes in its previous order-1 square [21]. Such forwarding pointers are continuously propagated, via piggybacking on beacon packets, to newcomers to that order-1 square. As a result, the size of the beacon packets tends to grow.

4.3. Robustness

We compare the effects of mobility on the successful resolution of queries in the three location services. In XYLS, the successful resolution of a query relies on the intersection between the update column and the query row. If both the column and the row are thin, it is likely the servers near the intersection have moved away and a query passes through the column without reaching any servers. In our implementation, each update is propagated at each hop with both a unicast and a broadcast, and thus the update column is expected to be $\sqrt{C}$ wide, i.e., containing about $\sqrt{C}$ nodes along the east–west direction, where C is the number of nodes within the range of a single transmission. This significantly improves the robustness of XYLS in the presence of node mobility. The cost is the doubled update transmissions.

In GLS, successful resolution of a query relies on up-to-date location information of the location servers a query needs to reach successively. Because of the exponentially decaying update frequency for location servers high up in the hierarchy, the order-1 grid location information for high-order location servers are likely to be stale. GLS addresses this problem by inserting forwarding pointers in order-1 squares. For high mobility and/or high-order location servers, the location information can be so stale that a query may need to follow forwarding pointer multiple times. This can potentially affect whether a query can be successfully resolved.

In GHLS, the handling of location server movement is decoupled from destination node movement. A location server always stores the up-to-date locations of destination nodes, hands off its share of location information whenever it discovers, based on received beacon packets, neighboring nodes that are closer to the hashed locations. Since the handoffs from X to Y happens as soon as the beacon from node Y is heard by its surrounding neighbors, the window of inconsistency, in which node Z thinks Y is closer than X but X has not handed off to Y, is at most one beacon duration. Thus GHLS is highly robust to node mobility.

5. Experimental methodology

We implemented all the protocols studied as well as geographic forwarding in the Glomosim [38] simulator. Our implementation and simulation parameters of geographic forwarding are based on [15]. Glomosim is a widely used wireless network simulator with a comprehensive radio model. We use a 802.11 radio model with a nominal bit-rate of 11 Mbps and a transmission range of 250 m. For all three location service protocols, we chose a beacon period of 2 s. The mobility scenarios use the modified random way point mobility model [37]. For mobile networks, all simulations use a pause time of 0 s with nodes moving at speeds randomly chosen between 1 and 9 m/s. The simulation duration is 300 s.

The following metrics are evaluated for the location service protocols: (1) *LS protocol overhead* – The number of location service (LS) protocol packets transmitted, with each hop-wise transmission of the protocol packet counted as one transmission. Note the LS protocol overhead excludes the overhead due to beaconing. (2) *Packet delivery ratio* (PDR) – The ratio of the data packets delivered to the destinations to those generated by the CBR sources. (3) *Query success ratio* (QSR) – The ratio of the number of query replies received at the sources to the number of location queries initiated by the sources.

6. Location service performance

In this section, we first evaluate the three location service protocols in a 600-node static and a 600-node mobile network to compare their relative performance and analyze the impact of mobility on the LS protocol overhead. We then study their scalability by evaluating their performance in networks ranging from 100 to 2000 nodes.

The query pattern in this section is chosen to study the efficiency of the query and update mechanisms of all the protocols. Every node in the network initiates 15 queries to look up the location of randomly chosen (unless noted otherwise) destinations at times randomly distributed between 30 and 300 s. Thus the number of queries originated are $N \cdot 15$. Also, if a query is not successful, no retry is initiated. To measure the accuracy of the query reply, when the query reply is received, each source sends a single data packet of size 128 bytes to that destination using the replied location. The PDR obtained in this scenario is an indication of the accuracy of the destination location since for all location services, the same greedy geographic forwarding is used to route the data packets. For all protocols, the value of d (update threshold) was fixed at 200 m. For GHLS, we also vary α as 0.5 and 1. The node density in all the scenarios is kept constant at 100 nodes/km^2.

6.1. Static networks

We first compare the performance of the three protocols in a static network to verify our analysis and separate the effects of mobility on the performance of the three protocols. Table 3 shows the various components of the LS protocol overhead for a static network and a mobile network of 600 nodes. The terrain is a square of 2500 m × 2500 m. To construct a perfect grid, GLS pads the area to a virtual area of 4000 m × 4000 m, and constructs a grid hierarchy of height 5. To decouple the LS protocol overhead due to the update frequency from that due to mobility, for the static network, we assume a constant update interval of 40 s for all three protocols. This interval is based on the average time taken to cross $d = 200$ m (update threshold) in the corresponding mobile network where nodes travel with random velocities between 1 m/s and 9 m/s.

Table 3
LS protocol overhead breakdown for a 600 node network

	Static network				Mobile network			
	XYLS	GLS	GHLS-1	GHLS-0.5	XYLS	GLS	GHLS-1	GHLS-0.5
Beacon overhead	89,400	89,400	89,400	89,400	89,400	89,400	89,400	89,400
LS protocol overhead	311,181	254,504	134,643	112,317	302,771	526,327	134,962	115,046
Updates								
Sent	19,200	24,466	4800	4800	17,708	44,502	4427	4427
Txed	184,319	83,099	32,611	26,752	166,798	351,940	30,651	24,396
Queries								
Sent	14,368	8055	8578	8613	13,498	7445	8459	8464
Txed	97,232	120,218	51,639	43,829	110,452	124,534	48,139	38,092
Replies								
Sent	6893	7073	8386	8308	7191	6793	8190	8361
Txed	29,630	51,187	50,392	41,736	25,521	45,980	45,815	37,235
Handoffs	–	–	1	0	–	–	10,357	15,323
FP Updates	–	0	–	–	–	3873	–	–

For GLS, this interval corresponds to the update interval of the order-1 sibling location servers. The higher order servers are updated at exponentially lower rates (with intervals being power-of-two multiples of 40 s).

We first verify the predictions of the quantitative model with the simulation results. In XYLS, the number of updates sent (19,200) closely matches the prediction in Table 2 (18,800). Since the dimension used in the simulation (2500 m) is 62.5% of the dimension assumed in the analysis (4000 m), the number of updates transmitted (184,319) is approximately 60% of the predicted value of 308,800 in Table 2. Similarly, the number of queries transmitted (97,232) is close to 60% of the value (153,258) predicted by the analysis. Lastly, the number of query replies is 29,630, which is again close to the predicted value of 38,314 after scaling down by the dimensions ($0.625 \cdot 38{,}314 = 23{,}754$).

In GLS, the number of update events is the same as in XYLS, i.e., 4700. The number of updates sent shown in Table 3 is close to the value analytically computed. The results in the simulation (Table 3) show that the actual number of updates transmitted is 83,099. The analytically calculated value for the same shown in Table 2 comes out to be 96,162. This discrepancy exists due to the difference between the actual simulation area and the virtual grid that GLS uses. GLS operates using a virtual grid of 4000×4000 m even if the actual simulation area is 2500×2500 m. As a consequence, the path length for the highest level hierarchy ($h = 5$) is overestimated by the analytical model which assumes the virtual grid and physical area to be of the same size. Thus, the path length for updating location servers in the highest order square is smaller than predicted by the analysis. In fact, we measured the average hop lengths for updates to the different levels of the hierarchy. These are shown in Table 4 along with a comparison to their analytical estimates ($\frac{1}{3}L_{update\_hops}(k)$) from Section 4.1. As expected, the measured average path lengths match well with those predicted except for the highest level hierarchy ($h = 5$). If we plug in the measured path lengths from Table 4 and Eq. (3), instead of the analytically computed path lengths from Eq. (2), we obtain the average number of transmissions per update event as 19.76, and the total number of updates transmitted as $4700 \cdot 19.76 = 92{,}872$. This matches well with the measured simulation results. Note that the measured update path length at $h = 5$ is close to 60% of the predicted path length at $h = 5$ because of dimension scaling.

Table 4
Effect of h on GLS average update path length

Hierarchy height	$h = 2$	$h = 3$	$h = 4$	$h = 5$
Path length (analysis)	1.71	3.42	6.84	13.68
Path length (measured)	1.92	3.5	7.28	8.94

The number of queries initiated for GLS is 8055 and the analysis closely predicts the query transmissions of 120,218 packets. Additionally, if we assume that all queries are answered, the analysis predicts the number of query replies transmitted to be 60,573. However, some queries are lost and consequently the overhead for query replies is 51,187.

Finally, in GHLS with $\alpha = 1$, the number of updates sent is close to the predicted value. The numbers of update transmissions, query transmissions, and query reply transmissions, are about 60% of those predicted in Table 2. This is again explained by the fact that the actual dimension used in the simulation (2500 m) is 62.5% of the dimension assumed in the analysis (4000 m).

In summary, the numbers predicted by the analysis for the three categories of protocol operations closely match the values from the simulation.

We now compare the location services based on the simulation results. Table 3 shows that in a static network, XYLS has the highest overhead among all three protocols. This is primarily due to the larger number of updates that are propagated up and down the column. GLS has a lower number of updates than XYLS. However, as predicted by the analysis, it has almost 5 times more updates than GHLS because it has to update multiple location servers at multiple levels. As a result, it has 2.55 and 3.11 times more update transmissions than GHLS-1 and GHLS-0.5. The number of query transmissions by GLS is the highest among all three protocols since the query has to reach the actual node and takes detours along the way.

In summary, for static networks, GHLS has the lowest overhead due to its lightweight update mechanism. Although as expected GHLS-1 has slightly higher query and query reply overhead than GHLS-0.5, both protocols have lower query overhead than GLS and XYLS. However, the effect of mobility on these protocols is yet to be characterized.

6.2. Mobile networks

To measure the impact of node mobility on the three location service protocols, we introduced mobility into the same 600-node scenario. Nodes now move according to a random waypoint model with velocities chosen randomly between 1 m/s and 9 m/s and with a pause time of 0 s. Table 3 shows a detailed breakdown of the LS protocol overhead for all three protocols.

1. *Comparison with static networks*: We first compare the overhead for each location service protocol in the mobile network relative to in the static network. The following observations can be made by comparing the columns in Table 3.

First, the number of updates sent for GLS increases by about 81.89% while the numbers for XYLS and GHLS both decrease by about 7.77%. The reason for the slight decrease in updates sent for XYLS and GHLS is as follows. In the mobile network, the nodes have different velocities randomly distributed between 1 m/s and 9 m/s. Thus for the same update distance (200 m), different nodes send different numbers of updates. In contrast, in a static network, every node sends an update every 40 s.

We investigated the reason for the 80% increase in the number of updates in GLS by separating the causes for different updates initiated. Table 5 shows the breakdown of GLS updates by their causes. A "Distance" update is caused when a node travels for distance *d* (the update threshold distance). A "Prediction" update is caused when the timeout advertised by a node in its previous update expires before the node has moved *d*. This can happen if the node slows down from the time it sent its last update. A "Distance + Prediction" update occurs when a node travels distance *d* and the last timeout window advertised also expires, i.e., the predicted timeout was correct. Finally, a "Grid" update is triggered when a node crosses a grid boundary.

Table 5 shows that grid crossings account for approximately 50% (21,000) of the total updates sent. Thus, the 80% increase in the number of GLS updates in the mobile scenario is attributed to frequent grid boundary crossing by the mobile nodes. In GLS, whenever a node crosses grid boundaries, it needs to update its old (high-order) location servers as well as select new (low-order) servers since its position in the hierarchy has changed. This causes an increased number of updates to be initiated in a mobile network for GLS. These updates do not occur in a static network and occur less frequently when mobility is lower.

Table 5
Breakdown of GLS updates by their cause

	Distance	Prediction	Distance + Prediction	Grid
GLS	15,724	801	6975	21,002

Second, in moving from the static network to the mobile network, the average number of transmissions per update in GLS grows from 3.39 to 7.91, whereas the same metric stays roughly the same in XYLS and GHLS. The reason for this is that in GLS, node mobility leads to temporal node state inconsistencies which result in detours in routing update packets towards the location servers.

Third, the number of queries sent for all three protocols are reduced by on average 2–6% in the mobile network. This is because mobility helps to improve the connectivity and the effectiveness of caching over time, resulting in more location queries being resolved using the cache at the source. However, the average number of transmissions per query for GLS increases by 12%. This shows the susceptibility of the GLS query process to node mobility. In particular, when location servers in GLS move, forwarding pointers may have to be followed to resolve a query thereby increasing the average number of transmissions per query. Similarly, in XYLS, the average number of transmissions per query increases by 21%. While searching for a next hop that preserves the east–west direction of propagation, stale neighbor location information due to mobility hampers the accurate choice of the next hop. (The effect is less severe in location update as the neighbor location information is less stale due to the broadcast used in location update in XYLS.) Thus, at every hop, there is a possibility of choosing a less optimum next hop, causing packets to take extra hops to reach the column. In contrast, the average number of transmissions per query is slightly reduced for both version of GHLS in the mobile scenario as compared to the static scenario.

Last, since all protocols use geographic forwarding for query replies, the reply overhead remains largely unchanged in the mobile scenario. In addition, mobility introduces the overhead of handoff packets in GHLS. Similarly, in GLS, when a node crosses grid boundaries, it broadcasts a one-hop forwarding pointer information packet (FP Update) to the nodes in its previous grid.

In summary, the significant increase in the average number of updates and in the average number of transmissions per update, and slight increase in the average number of transmissions per query in GLS in the mobile network compared to in the static network suggests that GLS is the most susceptible to node mobility.

2. *Comparison among the three protocols*: Table 3 shows that while the protocol overhead of GLS is 1.89 times that of GHLS-1.0 and 0.81 times that of XYLS in the static network, these ratios increase to 3.89 and 1.74 in the mobile network. In particular, while the number of update transmissions of GLS is 2.54 times that of GHLS-1 and 0.45 times that of XYLS in the static network, these ratios increase to 11.48 and 2.11 in the mobile network. Thus GLS is significantly more adversely affected by mobility than XYLS and GHLS. This suggests that if for static networks, the network size has to be 25,000 nodes for GLS to tie GHLS in terms of update transmissions, then for mobile networks, the network size has to be much larger for GLS to catch up with GHLS in terms of update transmissions. In moving from the static network to the mobile network, the protocol overhead of XYLS stays roughly the same while that of GLS is more than doubled. As a result, XYLS has lower protocol overhead than GLS in the mobile scenario although it has higher protocol overhead than GLS in the static network.

We further compare the query performance of GHLS, GLS and XYLS by measuring the average hop lengths of resolved queries and of their replies (depicted in Table 6). Note that for XYLS, the query length refers to the number of hops traveled by the one query (out of the two sent to opposite directions) that reaches a location server. Since in GLS, queries have to travel from the location server to the destination, whereas for the other two protocols, they just need to travel to the location servers, the average query length in GLS is twice that in XYLS or GHLS. In GLS, the query reply travels back from the destination node to the source node, and thus the query reply length is about half of the query length. In GHLS, a query travels the same number of hops as the query reply and as expected, both the query length and query reply length for GHLS-0.5 are smaller compared to GHLS-1. In XYLS, a query reply travels using geographic routing, making maximum progress to its destination at each hop, and thus is shorter than the corresponding query in XYLS which is forwarded along the east–west direction, not necessarily making maximum progress towards its destination at each hop. For the same reason, the query length in XYLS is also larger than that in GHLS-0.5.

Finally, Fig. 5 shows the CDF of the query resolution length, which is the sum of the query length and the query reply length. Again, for XYLS we only consider one of the two queries sent in opposite directions that reaches a location server. The figure shows that most of the queries in GHLS and XYLS are resolved within 15 hops whereas those in GLS take larger numbers of hops to reach back to the sources.

An interesting observation is that GLS has the highest Query Reply Length despite the fact that GLS query replies travel back just from the destination to the source and are thus significantly lower overhead than the GLS query. This occurs because: (1) Query replies in GHLS-0.5 and XYLS are expected to travel a shorter distance since they are restricted in scope (to a nearby column or to a smaller hashed area) compared to GLS in which case the source and destination are essentially random points. (2) GHLS-1, GHLS-0.5 and XYLS cache location updates with a timeout, thus the query replies in some cases may come back before reaching the destination, thus reducing the query reply hop length compared to GLS which requires query replies to only come from the destination.

Finally, note that the evaluation chooses a pause time of 0 s in order to stress the protocols. We did run simulations for a wide range of pause times and observed the trends to hold true across all pause times. As expected, as the network became less

Table 6
Average query and query reply hop lengths

Protocol	XYLS	GLS	GHLS-1	GHLS-0.5
Query length	4.53	10.42	5.05	4.03
Query reply length	3.34	5.91	5.05	4.04

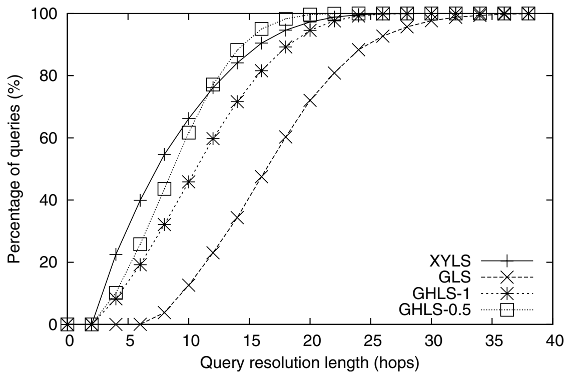

Fig. 5. CDF of query resolution length (query length + query reply length).

mobile the protocol performance became closer to the static scenario.

6.3. Load balance

When GHLS is used with $\alpha < 1$, nodes in the α region potentially bear more update and query traffic than a node outside. To measure this effect, we repeated the same 600-node experiment in the previous section (with mobility), e.g., keeping the same query rate, except the simulation is extended to 1500 s so that the results better reflect long term mobility. For all protocols, we logged the LS protocol packet transmissions inside and outside the α region where α is chosen to be 0.5. The measured results are shown in Table 7.

In XYLS, the *overhead ratio*, i.e., the ratio of the overhead outside to the overhead inside the α region, is the same as the ratio of their areas, 3:1. XYLS exhibits near perfect load balance because independent of where the nodes are, their updates and queries are sent across the network along the x- and y-dimensions. For GLS, the overhead ratio is 1.7:1, suggesting an imbalance and higher load in the α region. This occurs because in GLS, some updates are sent to faraway location servers in diagonal grids resulting in more load in the center of the network. In GHLS-1, the overhead ratio is approximately 1:1 also suggesting an imbalance. This occurs because of the fact that lines connecting points chosen at random within a square are likely to cross the center of the square. Compared to GHLS-1, GHLS-0.5 increases the overhead inside the α region by 25%, but reduces the overhead outside the region by 60%. Effectively, the choice of α decides a tradeoff between the total overhead incurred in the whole network and the load inside the α region.

The most important observation made from Table 7 is that despite the load in GHLS are more imbalanced than in XYLS and GLS, GHLS has significantly lower LS protocol overhead either outside (compared to XYLS) or both inside and outside (compared to GLS) the α region.

Table 7
LS overhead inside and outside the $\alpha = 0.5$ region

	LS protocol overhead	
	Inside	Outside
XYLS	334,605	1,044,216
GLS	729,951	1,308,840
GHLS-1	331,971	299,009
GHLS-0.5	416,635	119,828

In GHLS-0.5, although the average storage load on each node in the α region is four location entries, while nodes outside store no entries, each stored entry is of size 20 bytes, making the extra storage load on the nodes inside rather insignificant. In contrast, GLS stores each entry on $O(3 \cdot \log_4 N)$ servers, and thus the average number of entries stored at any node in the network can easily exceed 4 for medium to large networks.

6.4. Impact of locality in traffic patterns

In this section, we investigate the performance of the protocols under traffic patterns with locality. Such patterns are in favor of protocols such as GLS and XYLS which exhibit locality during query lookups. The experiment uses the same 600-node static network as before. A static network was used since it is difficult to instrument locality of traffic as nodes move; to preserve locality of traffic as nodes move, the traffic pattern would have to be continuously changed based on the new positions of the nodes. We consider three traffic patterns with increased locality. The first pattern is the random traffic pattern used in the previous experiments. In the second pattern, denoted as *10/15 Local*, 10 out of the 15 queries each source initiates go to nearby nodes ($\leqslant$750 m away) while the remaining 5 go to random nodes. In the third pattern, denoted as *15/15 Local*, all 15 queries go to nearby nodes ($\leqslant$750 m away).

Table 8 shows the overhead breakdown for the three protocols under the three traffic patterns. As expected, due to their protocol features, the query overhead of GLS and XYLS decreases as the locality in the traffic increases. This is because in GLS, the queries and replies are restricted to the largest square containing the source and the destination and therefore incur lower overhead when traffic is local. In XYLS, the query and reply distances and consequently overhead are reduced since the columns of the destinations are nearby. Although GHLS has no specific feature to exploit locality apart from scaling the α region, its query overhead is also reduced due to increased effectiveness of caching. This occurs because as the locality in the traffic increases, more nearby nodes look up locations of a similar set of destinations. Note the increased locality in the traffic also improves the effectiveness of caching in both XYLS and GLS.

Overall, with extreme locality in the traffic (15/15 Local), GLS incurs lower query overhead than

Table 8
Effect of locality in traffic patterns

	Queries	Replies	Updates	Total
Random				
XYLS	97,232	29,630	184,319	311,181
GLS	120,218	51,187	83,099	254,504
GHLS-1	51,639	50,392	32,611	134,643
GHLS-0.5	43,829	41,736	26,752	112,317
10/15 Local				
XYLS	73,088	16,988	184,319	274,395
GLS	60,756	30,675	83,099	174,530
GHLS-1	41,325	41,383	32,611	115,319
GHLS-0.5	34,541	34,446	26,752	95,739
15/15 Local				
XYLS	41,932	4069	184,319	230,320
GLS	13,113	4702	83,099	100,914
GHLS-1	22,003	21,332	32,611	75,946
GHLS-0.5	18,762	18,481	26,752	63,995

GHLS, which still has lower overhead than XYLS. The update overhead of GLS and XYLS, however, remain higher than that of GHLS since it is independent of the locality in the traffic. Because of the unchanged large gap between update overheads, GHLS has lower overall protocol overhead than GLS and XYLS. Note this result was observed for very high query rate; on average every node issues a query every 20 s, which favors GLS and XYLS.

6.5. Flooding versus rendezvous

In this section, we compare the performance of rendezvous-based location services and flooding-based ones. This allows us to understand the performance benefits and quantify the scalability of rendezvous-based location services. We choose the mobile network setup of 600 nodes used in previous sections for this evaluation.

We first compare the performance of a basic Proactive Location Service (PLS) and a basic Reactive Location Service (RLS) with the rendezvous-based location services. PLS requires each node to flood its location in the network whenever it moves *d*m, while RLS requires a node to flood the network in search for a node's location whenever it needs to route a packet to that node. RLS is similar to the reactive location service proposed in [18]. As seen in the results in Table 9, native flooding, either reactive (RLS) or proactive (PLS), is very expensive compared to the other location services. This is because inherently each flooding results in N transmissions.

Several techniques have been proposed to optimize this cost of the basic flooding mechanism and these proposals can be leveraged to reduce the cost of flooding-based location services. In particular, an approach that requires knowledge of nearby node locations is shown to perform the best across a variety of topologies and densities in [23]. For both RLS and PLS, we developed optimized versions using this technique, and the corresponding versions are referred to as OPT-RLS and OPT-PLS.

However, the results again show that even with optimizations to flooding, the lowest overhead version OPT-PLS still has an overhead of 1.7 million packets. In fact, to be comparable with GHLS, OPT-PLS would have to ensure network wide

Table 9
LS protocol overhead breakdown for a 600 node mobile network

	Rendezvous-based				Flooding-based			
	XYLS	GLS	GHLS-1	GHLS-0.5	RLS	PLS	OPT-RLS	OPT-PLS
Beacon overhead	89,400	89,400	89,400	89,400	89,400	89,400	89,400	89,400
LS protocol overhead	302,771	526,327	134,962	115,046	5,234,229	2,880,000	3,076,004	1,718,400
Updates								
Sent	17,708	44,502	4427	4427	–	4800	–	4800
Txed	166,798	351,940	30,651	24,396	–	2,880,000	–	1,718,400
Queries								
Sent	13,498	7445	8459	8464	8469	–	8469	–
Txed	110,452	124,534	48,139	38,092	5,189,400	–	3,031,902	–
Replies								
Sent	7191	6793	8190	8361	8462	–	8421	–
Txed	25,521	45,980	45,815	37,235	44,829	–	44,102	–
Handoffs	–	–	10,357	15,323	–	–	–	–
FP Updates	–	3873	–	–	–	–	–	–

Comparison with flooding.

dissemination of the location using approximately only 25 nodes out of 600 which is improbable in a random distribution of nodes. This illustrates the power of rendezvous-based approach versus flooding-based approach. While flooding essentially searches/disseminates locations in the entire space, rendezvous-based location services allow a low cost method for mutual exchange of information. Essentially, rendezvous-based location services provide a structured way to route and search for data similar to Distributed Hash Tables [27,30,29,39] in the Internet which have lower overhead and are better at performing resource discovery than unstructured search based schemes such as Gnutella.

6.6. Scalability

In this section, we compare the scalability of the three location service protocols in terms of the network size. Fig. 6 shows the performance of the three location services as the network size is scaled up from 100 to 2000 nodes. The same random waypoint model and the same query traffic pattern as in Section 6.2 are used. A pause time of 0 s is chosen.

Fig. 6a depicts the query success ratio of location queries as a function of the network size for all the location services. Similar to the study in [21], queries are not retransmitted, so a success indicates a success on the first try. The results show that for a small network size of 100 nodes, all protocols have identical query success ratios of close to 100%. As the network size increases, the QSR of GLS drops quickly, reaching 70% at 2000 nodes, whereas GHLS-0.5 achieves a consistently high QSR (above 98%). XYLS performs similarly to both versions of GHLS for small to medium networks but has a lower QSR than GHLS-0.5 for larger networks.

GLS's lower query success ratio is due to the difficulty of maintaining consistency by hashing in the namespace. This approach fundamentally is affected by mobility since query routing depends on consistent information being stored at nodes in all levels of the hierarchy. In contrast, both XYLS and GHLS only perform simple local operations when faced with mobility.

To measure the qualities of the location retrieved, each location query is triggered with a single data packet, and the packet delivery ratio of the data packets is measured. Note that we do not show

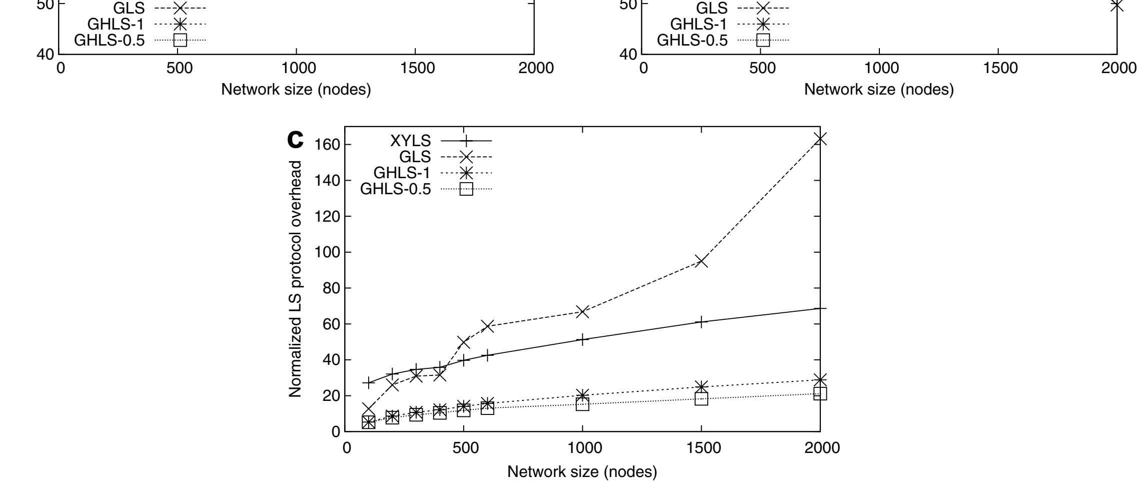

Fig. 6. Query success ratio, PDR and LS protocol overhead for a pause time of 0 s. (a) Query success ratio; (b) PDR; and (c) LS protocol overhead.

the geographical deviation of the location received in the query reply from the actual location of the destination since geographic forwarding is quite robust against slight inaccuracies in the location of a destination. Fig. 6b plots the PDRs under the three location services. It shows for all location services, the PDR curve closely resembles the corresponding QSR curve, suggesting almost all locations retrieved are accurate enough for successful data packet delivery.

Fig. 6c depicts the protocol overhead of all three location services. It shows that although the location services have similar overhead for a small network of 100 nodes, the overhead of both versions of GHLS is significantly lower than those of GLS and XYLS in larger networks, and the gap between GHLS and GLS widens as the network size is increased. More importantly, GHLS not only incurs lower protocol overhead, it also achieves higher QSR and PDR than GLS and XYLS. XYLS and GLS have comparable overhead in these large networks, although XYLS achieves higher QSR and PDR than GLS.

7. Geographic routing performance

In this section, we combine each location service with geographic forwarding to provide a data routing protocol. We then evaluate the data delivery performance of these protocols across a wide range of networks sizes. The number of nodes in the network is increased from 100 to 2000 nodes while keeping the node density constant at 100 nodes/km^2. A pause time of 0 s is chosen. Half of the nodes in the network are sources of data traffic, each originating one connection to a random destination and sending 128-byte packets at a rate of 4 packets/s for a duration of 20 s. Similarly as in [21], to reduce the effects of caching and fully stress the routing protocols, we restrict the number of times a node can be chosen as a destination to 3. Unlike in the previous section, an exponential backoff of location queries is done by each protocol whenever replies are not received. Additionally, all protocols maintain a table of live connections so that once a CBR flow is initiated, the destination and source can coordinate with each other, avoiding location queries for the duration of the flow, same as in [21]. GHLS uses an α of 0.5.

Our aim was to measure the performance of location-based protocols by taking into account location service overhead as well as geographic beaconing overhead. This allows us to understand the applicability of location-based protocols to different network configurations. We used two representative non-location-based routing protocols, AODV and DSR, in addition to three routing protocols based on the three location services studied in this paper. Note that although this comparison is unfair to the non-location-based protocols, it gives us an idea as to till what network sizes, location-based routing protocols can provide a large enough gain to justify the costs of including positioning systems on the mobile nodes. The AODV implementation provided by the authors of AODV follows draft version 13 of the protocol and incorporates local repair and query localization to optimize its performance in large networks. We augmented DSR with a graph-based cache [10] to reduce its overhead so as to allow it to scale further than the results presented in [21].

Fig. 7 shows the simulation results. The most important observations from the results are as follows. First, the AODV routing protocol with

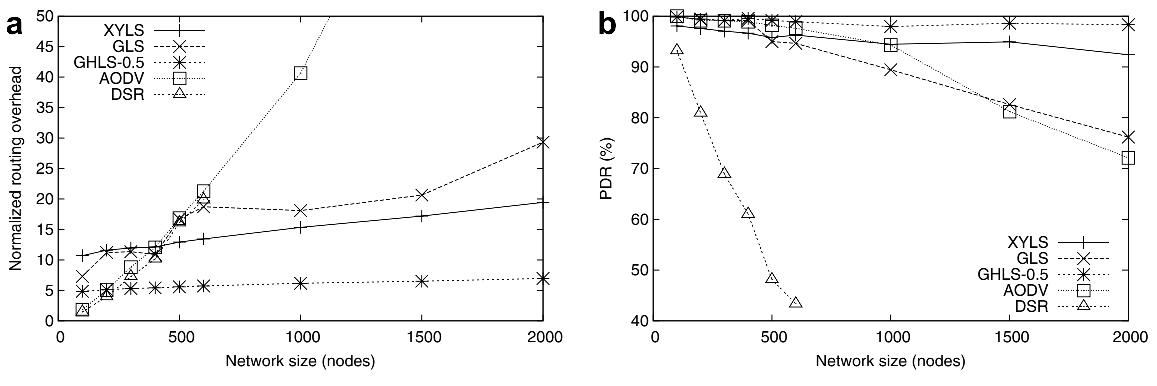

Fig. 7. Routing overhead and PDR for a pause time of 0 s. The routing overhead includes transmissions of all non-data packets. (a) Normalized routing overhead and (b) PDR (%).

scalability extensions performs well when compared to location-based protocols. For network sizes up to 400 nodes, AODV has lower overhead than GLS and XYLS while delivering a similar amount of data packets. This occurs due to the reactive design of AODV in contrast to the proactive location update mechanism used in the location-based protocols. In addition, optimizations like query localization and local repair reduce the routing overhead of AODV in the presence of mobility, especially in large networks. The results for DSR show that although it performs better when augmented with a graph cache than the results presented in [21] which used a path cache, the overhead of source routing in large networks and the absence of local repair and query localization in contrast to AODV severely hamper its scalability.

Second, GHLS has the lowest routing overhead among all protocols for network sizes larger than 300 nodes, while achieving the highest packet delivery ratio. Importantly, GHLS is the only location-based protocol that has comparable overhead to AODV and DSR for small networks. Thus, the nature of location service used in geographic routing impacts the scale of network at which geographic routing becomes useful compared to topology based routing protcols.

8. Related work

Several different classifications of location services have been proposed. In [22], location services are classified according to the number of servers and the extent of each servers usage. In [3], they are classified as being reactive or proactive. A survey of location services appears in [33].

While our work is the first (to the best of our knowledge) to evaluate the performance tradeoffs between rendezvous-based location services, several previous studies have either quantitatively compared flooding-based and rendezvous-based location services, or evaluated flooding-based services via simulations. Studies of individual protocols such as GLS have also been performed. In [22], the authors discuss the asymptotic complexities of protocols built using flooding-based and rendezvous-based location services such as greedy forwarding, DREAM [1], LAR [17], Terminodes [13], and Grid [21]. In [4], the authors compare three flooding-based location services (DREAM, Simple Location Service (SLS), and Reactive Location Service (RLS)). SLS relies on each node to periodically exchange location information with its neighbors whereas RLS floods the entire network with location queries on-demand. Another work [11] proposes a proactive protocol LEAP and compares it with SLS, RLS, and GLS for small networks. In [16], the authors propose a reactive location service similar to RLS in [4] and compare it to DSR and GLS running over GPSR. In [20], a proactive location service with a prediction model utilizing the distance effect is proposed and compared with other proactive approached such as SLS, DREAM and GRSS [9].

The GHLS protocol proposed in this paper shares the nature of geographic hashing with GHT [28]. However, GHT is proposed to support data storage in dense sensor networks which are typically static. Additionally, storage and replication strategies are fundamentally different for GHT since it stores sensed data where reliability and storage costs are a bigger concern.

Ad hoc positioning systems such as [26] have been proposed which can route packets geographically without using GPS hardware. Such systems can provide a node with an estimate of its virtual coordinates. However, a location service is still required to look up another node's virtual coordinates. An evaluation of such positioning systems is outside the scope of this paper.

9. Conclusions

In this paper, we presented a performance comparison of three scalable location services for geographic routing. The primary insight from our study is that when practical MANET sizes are considered, i.e., up to a few thousand nodes, the constants matter more than the asymptotic costs of location service protocols. In particular, we have shown that although asymptotically GLS is more scalable than both GHLS and XYLS in terms of protocol overhead, GHLS transmits fewer packets than GLS in networks of up to 25,000 nodes. Additionally, in contrast to the complex GLS design, the simplicity of GHLS provides significant resilience to performance degradation from mobility. Finally, although XYLS has a comparable packet delivery ratio to GHLS, it achieves this ratio with a higher overhead.

Hierarchical hashing-based protocols try to reduce their overhead by taking advantage of localized mobility. However, if communication is not local, the query has to travel high up in the hierarchy, and thus such schemes have to deal with inconsistencies at faraway location servers which degrade performance in mobile networks. The primary reasons for GLS losing to the flat hashing scheme GHLS are: (1) Maintaining multiple location servers at each level of the hierarchy increases the number of updates issued by each node by a factor of 3 every one interval, a factor of 6 every two intervals, a factor of 9 every four intervals, and so on. The average number of updates per interval approaches 6 quickly as the height of the hierarchy increases. (2) Hashing in the node ID space (for both updates and queries) using geographic forwarding unavoidably run into complicated consistency handling, for example when nodes cross-grid boundaries. In contrast, geographic hashing uses a simple handoff mechanism to deal with node mobility, avoiding complicated consistency handling. (3) While queries for nearby nodes can benefit from the locality inherit in the hierarchy, they can also suffer the "gridding" effect: source and destination nodes near the boundaries of two high-order sibling squares may have to travel many steps up the common sub-hierarchy containing both nodes, which defeats the purpose of using hierarchy for query locality. We have also shown that even for traffic patterns with high locality, GHLS incurs lower total protocol overhead than GLS due to its low cost location updates as well as effective caching in query resolution which benefits all three location services.

We have shown that reactive routing protocols like AODV provide an efficient alternative to geographic routing for small to medium sized networks. In fact, in such networks, the beaconing overhead of geographic routing and the location service update and lookup cost of GLS and XYLS make them less attractive than a cheaper, non-location-based solution. In contrast, geographic forwarding equipped with GHLS provides a routing protocol that is efficient in networks of a wide variety of sizes including small networks.

Acknowledgments

This work was supported in part by NSF Grant ANI-0338856.

References

[1] S. Basagni. et al., A distance routing effect algorithm for mobility (DREAM), in: Proc. ACM MobiCom, October 1998.
[2] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, in: Proc. ACM DialM Workshop, August 1999.
[3] T. Camp, Location information services in mobile ad hoc networks. Technical Report MCS-03-15, The Colorado School of Mines, 2003.
[4] T. Camp, J. Boleng, L. Wilcox, Location information services in mobile ad hoc networks, in: Proc. IEEE ICC, May 2002.
[5] C. Cheng, H. Lemberg, S. Philip, E.V. den Berg, T. Zhang, SLALoM: A scalable location management scheme for large mobile ad hoc networks, in: Proc. IEEE WCNC, March 2002.
[6] S. Giordano, M. Hami, Mobility management: The virtual home region. Technical Report SSC/037, EPFL, October 1999.
[7] Z.J. Haas, B. Liang, Ad hoc mobility management with uniform quorum systems, IEEE/ACM Trans. Netw. 7 (2) (1999) 228–240.
[8] Z.J. Haas, M.R. Pearlman, The performance of query control schemes for the zone routing protocol, in: Proc. ACM SIGCOMM, August 1998.
[9] P. Hsiao, Geographical region summary service for geographical routing, ACM MC2R 5 (4) (2002) 25–39.
[10] Y.-C. Hu, D.B. Johnson, Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks, in: Proc. ACM MobiCom, August 2000.
[11] X. Jiang, T. Camp, An efficient location server for an ad hoc network. Technical Report MCS-03-06, The Colorado School of Mines, May 2003.
[12] D.B. Johnson, D.A. Maltz, Dynamic Source Routing in Ad hoc Wireless Networks, Kluwer Academic Publishers, Dordrecht, 1996.
[13] J.P. Hubaux et al., Towards self-organizing mobile ad hoc networks: the terminodes project, IEEE Commun. Mag. 39 (1) (2001) 118–124.
[14] D. Karger et al., Consistent Hashing and Random Trees: Tools for Relieving Hot Spots on the World Wide Web, in: Proc. ACM STOC, May 1997.
[15] B. Karp, H. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in: Proc. ACM MobiCom, August 2000.
[16] M. Kasemann, H. Fuler, H. Hartenstein, M. Mauve, A reactive location service for mobile ad hoc networks. Technical Report CS TR-14, University of Mannheim, November 2002.
[17] Y.-B. Ko, N.H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in: Proc. ACM MobiCom, October 1998.
[18] M. Ksemann, H. Fler, H. Hartenstein, M. Mauve, A reactive location service for mobile ad hoc networks. Technical report, TR-02-014, Department of Computer Science, University of Mannheim, November 2002.
[19] F. Kuhn, R. Wattenhofer, A. Zollinger, Worst-case optimal and average-case efficient geometric ad hoc routing, in: Proc. ACM MobiHoc, June 2003.
[20] V. Kumar, S.R. Das, Performance of dead reckoning-based location service for mobile ad hoc networks, Wireless Commun. Mobile Comput. J. (2003), December.
[21] J. Li, J. Jannotti, D.S.J.D. Couto, D.R. Karger, R. Morris, A scalable location service for geographic ad hoc routing, in: Proc. ACM MobiCom, August 2000.

[22] M. Mauve, J. Widmer, H. Hartenstein, A survey on position-based routing in mobile ad hoc networks, IEEE Network (1999) 30–39, November/December.
[23] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, J.-P. Sheu, The Broadcast Storm Problem in a Mobile Ad Hoc Network, in: Proc. ACM MobiCom, August 1999.
[24] C.E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in: Proc. ACM SIGCOMM, August 1994.
[25] C.E. Perkins, E.M. Royer, Ad hoc on-demand distance vector routing, in: Proc. IEEE WMCSA, February 1999.
[26] A. Rao, C. Papadimitriou, S. Shenker, I. Stoica, Geographic routing without location information, in: Proc. ACM MobiCom, September 2003.
[27] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, A scalable content-addressable network, in: Proc. ACM SIGCOMM, August 2001.
[28] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, GHT: A geographic hash table for data-centric storage in sensornets, in: Proc. ACM WSNA, September 2002.
[29] A. Rowstron, P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, in: Middleware, November 2001.
[30] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: Proc. ACM SIGCOMM, August 2001.
[31] I. Stojmenovic, Home region based location updates and destination search schemes in ad hoc wireless networks. Technical Report TR-99-10, SITE, University of Ottawa, September 1999.
[32] I. Stojmenovic, A routing strategy and quorum based location update scheme for ad hoc wireless networks. Technical Report TR-99-09, SITE, University of Ottawa, September 1999.
[33] I. Stojmenovic, Location Updates for Efficient Routing in Ad hoc Wireless Networks, Wiley, New York, 2002.
[34] USCG Navigation Conter GPS page. http://www.navcen.uscg.nil/gps/default.html, January 2000.
[35] S.-C.M. Woo, S. Singh, Scalable routing protocol for ad hoc networks, Wirel. Netw. 7 (5) (2001) 513–529.
[36] Y. Xue, B. Li, K. Nahrstedt, A scalable location management scheme in mobile ad hoc networks, in: Proc. IEEE LCN, November 2001.
[37] J. Yoon, M. Liu, B. Noble, Random waypoint considered harmful, in: Proc. IEEE INFOCOM, April 2003.
[38] X. Zeng, R. Bagrodia, M. Gerla, Glomosim: A library for parallel simulation of large-scale wireless networks, in: Proc. PADS Workshop, May 1998.
[39] B.Y. Zhao, J.D. Kubiatowicz, A.D. Joseph, Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U.C. Berkeley, April 2001.

Saumitra M. Das is currently a Ph.D. candidate in the School of Electrical and Computer Engineering at Purdue University, USA. Previously, he received a MS degree from Carnegie Mellon University, USA and a B.Engg. degree from the University of Bombay, India. His research interests include cross-layer system design for multi-hop wireless networks, scalable routing strategies in wireless ad hoc networks, and mobile robotics.

Himabindu Pucha is currently a Ph.D. candidate in the School of Electrical and Computer Engineering at Purdue University, USA. Previously she received a MSEE degree from Purdue University, USA and a B.Engg. degree from the University of Bombay, India. Her research interests include Internet routing and overlay networks, peer-to-peer systems and applications, and mobile computing.

Y. Charlie Hu is an Associate Professor of Electrical and Computer Engineering and Computer Science at Purdue University. He received his M.S. and M.Phil. degrees from Yale University in 1992 and his Ph.D. degree in Computer Science from Harvard University in 1997. From 1997 to 2001, he was a research scientist at Rice University. His research interests include operating systems, distributed systems, networking, and parallel computing. He has published over 100 papers in these areas. He received the NSF CAREER Award in 2003. He served as a TPC vice chair for the 2004 International Conference on Parallel Processing, and a co-founder and TPC co-chair for the International Workshop on Mobile Peer-to-Peer Computing. He is a member of USENIX, ACM, and IEEE.