Journal of Network and Computer Applications 83 (2017) 1–11

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

PETAL: A fully distributed location service for wireless ad hoc networks

Amir Rahimzadeh Ilkhechi[a,*], Ibrahim Korpeoglu[b], Uğur Güdükbay[b], Özgür Ulusoy[b]

[a] Duke University, Computer Science Department, North Building, Office #N303A, 304 Research Drive, Durham, NC 27708, USA
[b] Bilkent University, Computer Engineering Department, Turkey

A R T I C L E I N F O

Keywords:
Location service
Mobile ad hoc network
Geographical routing
Swarm mobility

A B S T R A C T

Location service is an essential prerequisite for mobile wireless ad hoc networks (MANETs) in which the underlying routing protocol leverages physical location information of sender and receiver nodes. Fulfillment of this requirement is challenging partly due to the mobility and unpredictability of nodes in MANETs. Moreover, scalability and location information availability under various circumstances are also substantial factors in designing an effective location service paradigm. By and large, utilizing centralized or distributed location servers responsible for storing the location information of all, or a subset of participant mobile devices, is a method employed in a significant portion of location service schemes. However, from the fairness point of view, it is more suitable to employ a location service scheme that treats participant nodes fairly, without mandating an unlucky subset to undertake the responsibility of serving as location server(s). In this work, we propose a scalable and fully decentralized location service scheme (PETAL) in which the burden of location update and inquiry tasks is almost evenly distributed among the nodes, resulting in an improvement in resilience against individual node failures. PETAL does not require hashing which results in more complexity, it is resilient against swarm mobility pattern, it requires minimal periodic location update messages when nodes do not move, and finally it does not require too many parameter configurations on all nodes. Our simulation results reveal that PETAL performs efficiently, particularly in environments densely populated by wireless devices.

1. Introduction

Statelessness is often used as a differentiating characteristic to categorize wireless network routing protocols. State-based routing protocols require that the participating nodes store information about the current topology and other universal properties of the whole network. DVRP (Perkins and Royer, 2003) and Link State (Clausen et al., 2003) routing algorithms are two examples of protocols that require per-node states. In contrast, stateless routing protocols work with a different mechanism that does not require participant nodes to store expensive states in terms of memory and/or updates. For example, greedy location-based routing protocols like GPSR (Karp and Kung, 2000) and GOAFR (Hwang et al., 2009) require each node to store just a small state containing information about their neighboring nodes. Some routing algorithms are considered hybrid which combine these two approaches (Kang et al., 2012; Bok et al., 2016; Arora and Jangra, 2012). Alotaibi and Mukherjee (2012) categorize the wireless routing protocols more precisely. The literature reviews of Cadger et al. (2013), Mauve and Widmer (2001), Liu et al. (2016), Mahmood and Manivannan (2015) and Jain and Sahu (2012) are literature reviews that provide description and comparison of various position-based and geographical routing algorithms and protocols.

While statelessness is a desired property, all location-based routing protocols are dependent upon the physical locations of senders and receivers. Therefore, in order to function correctly, a distributed and scalable location service is a requirement for mobile ad hoc networks whose underlying routing protocol is based on geographical routing.

State-of-the-art location service schemes, regardless of their level of centrality, are mostly based on the client—server model in which a subset of participant nodes are required to serve as location servers. The idea of employing one or multiple dedicated location servers is considered naive since it opposes the infrastructure-free and ad hoc nature of the network. Furthermore, since the location servers in a MANET are ordinary wireless nodes, their mobility should not be restrained. Otherwise, the non-server wireless nodes would need to inquire the location of the location servers which cannot be implemented efficiently (Mauve and Widmer, 2001) (better known as egg and chicken paradox). Although deploying non-dedicated location servers alleviates the mentioned drawback, still several undesired properties might be associated with such a design that we highlight

* Corresponding author.
E-mail addresses: ilkhechi@cs.duke.edu (A.R. Ilkhechi), korpe@cs.bilkent.edu.tr (I. Korpeoglu), gudukbay@cs.bilkent.edu.tr (U. Güdükbay), oulusoy@cs.bilkent.edu.tr (Ö. Ulusoy).

http://dx.doi.org/10.1016/j.jnca.2017.01.021
Received 9 October 2016; Received in revised form 22 December 2016; Accepted 20 January 2017
Available online 25 January 2017

as following:

1. A location server might be a single point of failure or a bottleneck if it is intended to store location information of some intensely queried nodes: the idea of choosing a larger subset instead of one node as location server mitigates the issues of too many nodes being assigned to the same location server. However, it is still possible that the location information of some nodes is queried more intensely than the others. If a server receives a massive number of requests in a short window of time, it will introduce some delay in the response which can degenerate the response time and even cause the client side to time out.
2. There should be a boundary that decides what change in a node's location is important enough to be quickly reported to the location servers: location updates can be bandwidth consuming if all of the servers are strictly required to store records of precise location information corresponding to the nodes that are assigned to those servers. As a result, location update packets might congest the network.
3. If an ad hoc network is mobile as a swarm (Li et al., 2012) such that the nodes seem still relative to each other, since the nodes are required to report their absolute location to the servers, the network might be congested due to excessive location updates: as an example suppose that a large group of mobile agents moves towards a target. While the absolute location of a large subset of nodes changes, the location service should be resilient enough to cancel out the movements of nodes relative to each other.
4. Node ID translation is quite strongly coupled with the idea of using location servers: the selection of server(s) responsible for tracking a given node and the process of querying the servers requires a means of converting unique IDs such as IP, MAC, or host names into names more suitable for the protocol. Hashing methods are widely employed for address conversion, and a consistent hashing algorithm is applied to decide which node(s) must take the responsibility of serving as location server(s). While address translation and hashing requirements are not considered drawbacks by themselves, they add complexity to the design and cause the behavior of a protocol be dependent on qualities of the utilized address translation and hashing algorithms such as uniformity and consistency.
5. Dividing the area into grids adds another level of complexity into some of the location service schemes. There must be an initial agreement on the size and other specifications of the grids such that all of the nodes would be able to follow the right steps when attempting to update their location information on a server or query a target node's location. Moreover, deciding the size of a grid is often dependent on the physical attributes of the nodes such as transmission range. These approaches implicitly accept that nodes are homogeneous while in a modern mobile ad hoc network a wide spectrum of different devices with varying transmission ranges might be present.

In this work, we introduce PETAL, a location service scheme that routes location queries to the target nodes without deploying location servers but instead by using the collective abstract location information stored in the other nodes as the guide to route location query packets to their destination. We define the abstract location information as the relative location of the target in terms of the quadrant (*northwest or NW, northeast or NE, southwest or SW, and southeast or SE*) in which it is located. In a special case, if a target node is located exactly in the border of two quadrants (e.g., at the intersection of *NE* and *NW*), we count it as if it is located on both of these quadrant. PETAL has no single points of failure; it ignores minor movements that have no effect on routing location query packets; it is resilient against swarm mobility of the network (Li et al., 2012); no hashing and/or name translation is required by PETAL; and finally, it is not grid-based, so there is no need for grid size configuration on the nodes. Below, we describe the notion of abstract location information in more detail:

Consider a MANET without location servers. We can think of two extreme cases: one of them is to enforce every node to store information about the physical location of all the other nodes. In this case, the necessary location update procedures, as a result of node movements, can bound network scalability. Another extreme case is to store no location information and allow the nodes to flood the location inquiry requests throughout the network to be received and replied by the corresponding destination nodes. This approach may result in a dramatic degradation in the throughput. A moderate approach may require the nodes to store abstract location information (as opposed to coordinates) about other participants. Location information is generally stored as a coordinates pair which is susceptible to be invalidated by slight movements. The validity of abstract location information (e.g., the quadrant in which the destination node is located), however, may not be affected by insignificant relocations. Based on that intuition, we propose a fully decentralized location service scheme (called PETAL) in which a minimal number of location update messages are needed to be sent to distant nodes. In other words, the location information stored in a node is abstract enough (e.g., something other than the absolute location information) that is not affected by movements of the other, relatively far away nodes. We demonstrate that the abstract location information is sufficient to route requests directly to the destination node possibly from an acceptably short path.

Our approach is to divide the surrounding environment of a node into four regions (northeast, northwest, southeast and southwest) in a two-dimensional context. Every node stores the location information as four lists of nodes that are located in each region. The nodes that reside in a given region may be either one hop or multi-hop neighbors. In our scheme, the query originated from a source is directly routed to the destination based on the abstract location information stored in forwarding nodes. The exact location information is then sent back from the destination itself.

The rest of the paper is organized as follows: Section 2 briefly explains the ideas behind some of the location service schemes. Section 3 presents the proposed scheme in detail. Section 4 compares our approach with four other location service schemes. Finally, Section 6 concludes the paper and proposes potential future work.

2. Related work

In DREAM framework (Basagni et al., 1998), every node stores position information about any other node that is a part of the network. From this point of view, it is similar to our approach. Any node sends update messages to the others based on their distance. The more remote the nodes are, the less frequent the update messages are sent. However, the information stored even in the distant nodes may be up to date, and as a result, sending update messages may not be necessary. PETAL is more efficient since neighbors of a moving node trigger an update procedure only if the update is necessary for future routings.

A quorum-based location service is proposed in Haas and Liang (1999). The concept of quorum systems is widely used in databases and distributed systems for information replication. In the mentioned framework, a similar approach is used where a backbone of location servers is formed in the mobile ad hoc network. Any location inquiry is sent as a request to the closest location server in the backbone. If the recipient server has no updated information, it circulates the request among the other backbone servers until it is received by any location server that has up-to-date information. Using the backbone approach implies discrimination between the nodes, which is an undesirable aspect of quorum-based location service. On the other hand, managing the backbone itself is difficult because of backbone nodes' movements. Similarly, Stojmenovic et al. (2008) propose a location service based on the concept of quorums for sensor networks. PETAL has no backbone and therefore the burden of location service is distributed more evenly among the participating nodes.

In Giordano and Hamdi (1999), Stojmenovic (1999), and Woo and Singh (2001) the concept of virtual home zone is used. A home zone is where the position information for a node is stored. The position of the home zone for a node can be derived by applying a well-known hash function to the node identifier. All of the nodes within a disk with radius R centered at C (home zone position) have to maintain position information for the node. If a home zone is sparsely populated, R may have to be increased, resulting in multiple trials for updates as well as queries. An important drawback of this approach is that a location server can potentially be far away from both the source and destination nodes, resulting in inefficient location update and query operations. In PETAL location updates that nodes send to each other are independent of their current and future locations. The potential problem of long distance between the location server and the querying node is therefore solved in PETAL.

The Grid Location Service (GLS) (Li et al., 2000) divides the area containing the ad hoc network into a hierarchy of squares forming a quad-tree. Each node selects one node in each element of the quad-tree as a location server such that the density of the location servers for a given node is high in the nearby areas and becomes exponentially sparser as the distance to the node increases. The update and request mechanisms of GLS require that a chain of nodes based on node IDs is found and traversed to reach an actual location server for a given node. The chain leads from the updating or requesting node via some arbitrary nodes to a location server. Traversing the chain of mobile nodes can result in a significant update and lookup failures if node mobility is high (Käsemann and Hartenstein, 2002): as soon as one of the dedicated nodes in the chain cannot be reached, the update or request message is lost.

In the location service scheme named GPLS (Zhou et al., 2013), the network is partitioned into grids and they are divided into groups by using a HASH function, which guarantees the uniform distribution of responsible location servers of any node. Out of each grid, the node with the highest power becomes the location server. Any node has a home grid. The location update messages are sent to the nearest grid that is in the same grid group with the moving node. The servers of the same grid group communicate with each other on a timely basis to update their information. One glaring shortcoming in this scheme is to implicitly assume that the servers in the grids do not move. If the servers start to move to other grids, the process of recovery can be complicated and bandwidth consuming.

Flat-based Some-for-some Location Service (FSLS) is proposed in Derhab and Badache (2008) for ad hoc mobile networks. The mentioned location service is based on the hash-based sets system. It divides the network area into non-overlapping zones and node identifiers are then mapped to a set of home zones. Each home zone has a unique location server, therefore this approach can be categorized as some-for-some.

As described in Section 1, PETAL avoids using hashing and grid-based methods. Therefore, it is less complex compared to GLS and GPLS. Nevertheless, our simulation results reveal that it is more efficient. Ahmed et al. (2009) proposes a grid-based location service that is more resilient against non-uniform node distributions compared to the past approaches such as GLS (Li et al., 2000). A prediction-based mechanism is proposed in Cheng and Huang (2012) which uses mobility information to predict the current location of a target node. In this scheme, every node examines its current location and also the mobility information it has previously sent. Then it decides whether sending out location-update packets is necessary or not. Correctness of PETAL is not dependent on the distribution of the nodes. However, if the distribution is denser and more balanced, it may perform more efficiently.

Some other past works are less relevant but also important: (Marwane et al., 2014; Saleet et al., 2010; Wu et al., 2014) focus on designing a location service and routing protocol for vehicular ad hoc networks (VANETs). The authors argue that high mobility of nodes is a distinguishing factor for VANETs and results in more frequent topology changes and link disconnections. Shen and Zhao (2013), El Defrawy and Tsudik (2011) study the location privacy in location-based routing. There are other works that focus on the energy efficiency aspects of the location service such as Wang et al. (2013).

3. The proposed location service scheme

Every location service scheme should specify the type of information that is stored in any node and also address two major requirements: location update and location query. The location update procedure used in our proposed scheme consists of two major phases: (1) *Table Initialization phase* and (2) *Dynamic phase*. Apart from that, the location queries are routed directly to the destination using a simple routing algorithm. Our routing algorithm guarantees that the destination itself will receive the location query. As an improvement, the connection request can be accompanied with the location query to decrease the connection delay in case of using a connection-oriented transmission layer protocol. The type of information stored in the nodes and the details of location update and location inquiry are provided in the following subsections.

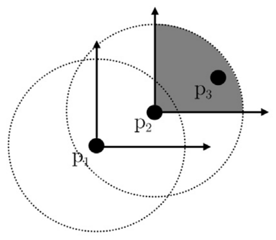
3.1. Forming the location information in the nodes

In our scheme, every node stores a table in which any record belongs to a particular node other than itself. The only information stored in these records is the quadrant in which the target node is located and the set of nodes that report the target node. For example, for the node p_1 shown in Fig. 1(a), the corresponding table is provided in Fig. 1(b). The p_3 cannot be discovered by p_1. Instead, p_2 serves as a reporter for p_3.

We take advantage of a relation called *dominance* (Preparata and Ian Shamos, 1985): A point p is said to be dominated by point p' if both x and y coordinate values of p' are greater than that of p. For example, in Fig. 2, all of the points are dominated by point d. We can define a similar and more general relation as:

R-dominance: A point p *R-dominates* a set of points P, if and only if $\forall\ p' \in P$, p is at the R quadrant of p' where $R \in$ {Northwest (*NW*), Northeast (*NE*), Southwest (*SW*), Southeast (*SE*)} (e.g., consider the set of points in Fig. 2 again. The point d' *SW-dominates* the set of all the other points in the figure.)

Suppose that nodes A and B are one-hop neighbors and likewise are B and C, but A and C are not immediate neighbors of each other. If point A *R-dominates* point B and point B itself *R-dominates* point C, it means that C is *R-dominated* by A as well. In other words, if C is moving in the region that is *R-dominated* by B, there is no necessity to update the corresponding node's state in A. Consider Fig. 1(a), where p_1 *SW-dominates* p_2 and p_2 itself *SW-dominates* p_3. The shaded region shows the area in which p_3 can freely move without any need for updating its corresponding state in p_1.



(a)

Node ID	Region	Reporters
p_2	*NE*	p_2
p_3	*NE*	p_2

(b)

Fig. 1. Part (a) shows three wireless nodes. Part (b) shows the table stored in node p_1. The first column is the ID of the node to which the row belongs. The second column indicates the region in which it resides. The last column shows which nodes have reported the node corresponding to this row so far. The number of reporters can be more than one.

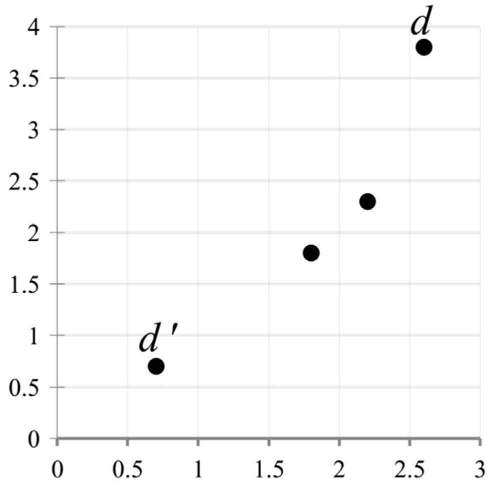


Fig. 2. Location information of four mobile devices.

In wireless ad hoc networks whose participating devices are equipped with positioning systems like GPS, it is possible for the nodes to find out if they *R-dominate* the one-hop neighbors by just comparing the location information of their immediate neighbors with their own. Multi-hop neighbors that are *R-dominated* can be reported by the single-hop neighbors.

3.2. Table Initialization phase

In this phase, the nodes collect information about their surroundings by simply reporting the nodes that they *R-dominate* to the nodes that they are *R-dominated* by. This phase might require sending a large number of messages. Once the initial states are formed in the nodes (equilibrium is attained), the dynamic phase (which is discussed in the next subsection), requiring less number of messages to be exchanged begins.

After completion of this procedure, any node will acquire a partial knowledge about petal shaped areas of the network in different regions as shown in Fig. 3. At that point, the majority of the participants will be most likely discovered depending on the density and distribution of the nodes. The following subsections discuss the possible problems and their solutions when discovering the petals.

3.2.1. Dealing with disguised nodes

The holes in the petal shaped areas can disguise some of the nodes, which is a significant issue. A hole is a void area formed because of the lack of coverage or the presence of a physical block in the network. In the network graph, a hole usually appears as a convex polygon. To understand the type of problems that holes cause, consider Fig. 4. In this figure, the node D cannot be discovered in the petal, because it has no neighbors at its *NW* quadrant. This subgraph can be considered as a part of the *SE* petal for node S. To tackle that problem, we define *virtual R-neighbors*:

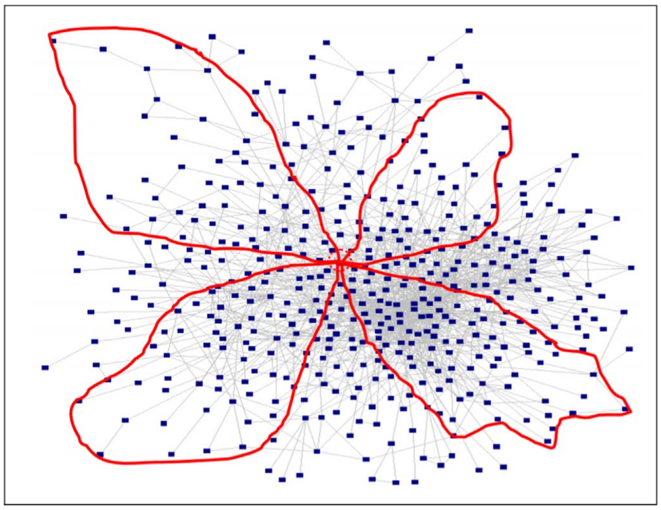


Fig. 3. The petal shaped areas that are discovered by the central node.

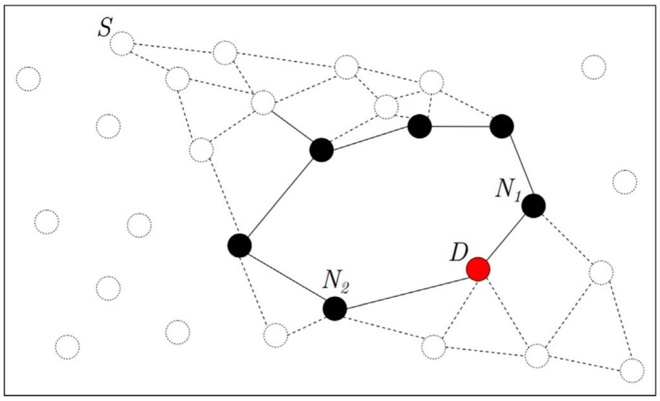


Fig. 4. An example of a disguised node.

Any node that has no neighbors at its R region, will take one (or possibly two) of its neighbors from the closest regions as its virtual neighbors. Suppose that the neighbors of any given node x are lexicographically sorted according to their polar angle with respect to x as origin and their distance from x: the virtual neighbor at the clockwise neighbor region is the one that is the leftmost (i.e., it is the closest node to the boundary of the clockwise region). Likewise, the virtual neighbor at the counterclockwise neighbor region is the one that is the rightmost.

In our example in Fig. 4, node D takes N_1 and N_2 as its virtual neighbors. N_1 is the leftmost neighbor at the adjacent clockwise neighbor region, and likewise N_2 is the rightmost neighbor at the adjacent counter-clockwise region. Both D and its virtual neighbors treat each other as real neighbors, that is, D assumes that N_1 is located both in *NE* and *NW* and N_1 on the other hand assumes that D is located both at *SW* and *SE* regions. A similar procedure is followed by any other node (located in the vertices of the polygon) that is void of nodes in any of their regions. If more than one region is empty, then it will be virtually filled by nodes in the closest regions as described above. Virtual neighbors will guarantee that no node in a given petal will be disguised.

3.2.2. Network boundaries versus hole boundaries

The idea of *virtual neighbors* treats the network and hole boundary nodes in the same manner. Network boundary nodes are the boundaries of the network graph. Hole boundary nodes are the ones that are inside the network span but lack a neighbor in at least one region. For example, in Fig. 4, node D is a hole boundary node. If we use the idea that is proposed in the previous subsection, network boundary nodes will look for virtually dominated nodes since a hole and a network boundary node look locally the same (i.e., both lack one-hop neighbors in at least one quadrant). This can cause an infinite loop when routing a location request because as we describe in the rest of the paper, network boundary nodes do not forward a request. To avoid that, every node that lacks a neighbor in a specific region, periodically propagates a message in the whole network that contains the ID of the sender and the void region. Any node that geographically resides in that region, will respond to the sender. If no response is returned from any other node, the sender can be sure that it is a network boundary node and does not need to discover virtually dominated neighbors.

3.3. Dynamic phase

The movements of the nodes must trigger reactions in the neighbor nodes in order to keep the stored states valid. When a node starts to move, it compares its own location with those of its neighbors. When the relative location between any pair is disturbed, a chain reaction begins. Suppose that node *B* is located at the *NE* region of node *A*. Suddenly, node *C* enters to the *NE* region of *A*. Because of beacon messages, *A* will be notified about the presence of *C*. Now, node *A* has to report the entrance of *C* to its own *SW* one hop neighbors. The neighbors at *SW* region either know that *C* is at their *NE* or they do not. If they have *C* in their records with value *NE* for the region, then they will simply add *A* as a new reporter for *C*. Otherwise, they will create a new record for *C*. In the second case, the *SW* neighbors have to report *C* to their own *SW* neighbors. This process continues until the first case happens.

A will add *C* as the reporter for the reported nodes or add a new record as described above. From *C*'s point of view, *A* is a new *SW* neighbor. In a similar way, *C* reports *A* as its *SW* neighbor to the *NE* neighbors.

Algorithms 1 and 2 show the details of procedures that are triggered in the discoverer node and all of the nodes that receive the updates, respectively.

Algorithm 1. Reaction triggered when discovering new single hop nodes in region *R* (Procedure *ReportDiscoveredOneHops*(*R*)).

```
1:  OneHopNodeList ← List of one-hop nodes discovered in R
2:  NodeList ← empty list
3:  for any node N in OneHopNodeList do
4:     add N and R′-dominated nodes of N to NodeList without
       duplicating (R′ is the diagonally neighbor region of R)
5:  end for
6:  ToBeAdded ← empty list
7:  for any node N in NodeList do
8:     if there is info about N in the table then
9:        update the reporter list for N
10: else
11:       add a new row for N
12:       add N to ToBeAdded
13: end if
14: end for
15: send ToBeAdded to the diagonally neighbor region of region R
    (triggers their ForwardToBeAdded)
```

Algorithm 2. Forwarding the *ToBeAdded* list received from node *RN* (Procedure *ForwardToBeAdded* (*R*, *ToBeAdded*, *RN*)).

```
1:  for any node N in ToBeAdded do
2:     if there is info about N in the table then
3:        add RN to the reporter list of N
4:        delete N from ToBeAdded
5:     else
6:        add a new row for N
7:     end if
8:  end for
9:  if ToBeAdded is not empty then
10: send ToBeAdded to any node residing at diagonal neighbor
    region of region R (triggers their ForwardToBeAdded proce-
    dure)
11: end if
```

Now, consider a different scenario in which *B* travels and goes out of *A*'s *NE* region. The reporting procedure has to be properly performed because if *A* immediately reports that *B* is gone, it might affect so many other nodes in *SW* region while the state change is transient.

According to Fig. 5, suppose that *C* after exiting from *A*'s *NE* region,

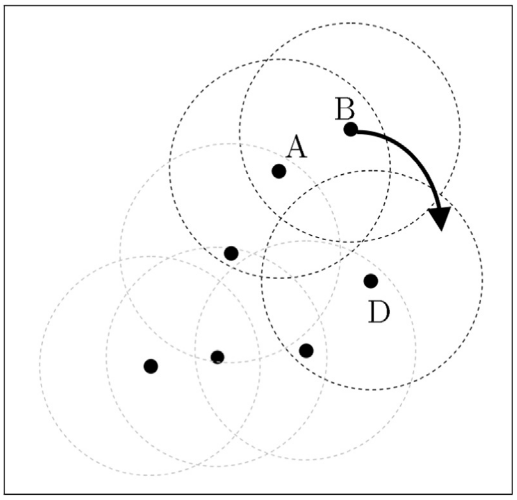


Fig. 5. Node *B* travels from *NW* quadrant of *A* and enters to *D*'s *NW* quadrant.

enters the *NE* region of D. *A* will report to the *SW* neighbors that *C* is lost and *D* will report that *C* is discovered. If *A* sends its message before *D*, then as a chain process, every *SW* neighbor of the *SW* neighbors might erase *C* from their tables and when it enters *D*, a significant portion of them must register it again. To avoid that, we force *A* to wait for a constant period time *T* before reporting the disappearance of *C* from its *NE* region.

Algorithm 3 shows the details of the procedure that is executed when a node loses a one-hop neighbor in region *R*. Similarly, Algorithm 4 provides the details of the procedure that is executed when some particular multiple-hop neighbors of lost node receive the update.

Algorithm 3. Reaction triggered when losing old single hop neighbor nodes in region *R* (Procedure *ReportLostOneHops*(*R*)).

```
1:  NodeList ← List of nodes lost in region R
2:  ToBeDel ← empty list
3:  for any node N in NodeList do
4:     remove N from reporter list of any node in region R and also
    remove the row for node N.
5:     add N to ToBeDel
6:  end for
7:  wait for t units of time
8:  if there is a row lacking any reporter then
9:     add the corresponding node to ToBeDel
10: end if
11: send ToBeDel to any node at the diagonal neighbor region of
    region R (triggers their ForwardToBeDelList procedure)
```

Algorithm 4. Forwarding the *ToBeDel* list received from node *RN* (*ForwardToBeDel*(*R*, *ToBeDel*, *RN*)).

```
1:  for any node N in ToBeDel do
2:     if there is info about N in the table then
3:        update the reporter list for N (remove RN)
4:        if row of N lacks reporters then
5:           remove N's row
6:        else
7:           remove N from ToBeDel
8:        end if
9:     else
10:       remove N from ToBeDel
11:    end if
12: end for
13: if ToBeDel is not empty then
14:    send ToBeDel to any node at the diagonal neighbor region of
    region R (triggers their ForwardToBeDel)
```

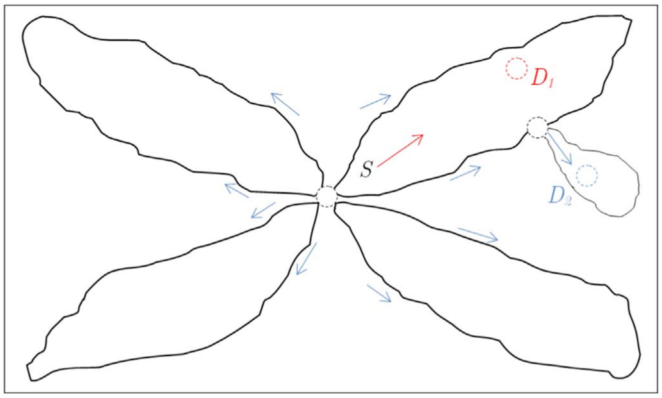


Fig. 6. Queries originated in two different cases. Red arrow shows the traveling direction of a single query. Blue arrows show the traveling directions of multiple queries. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

```
15:  end if
```

3.4. Location query

After the table initialization phase, any node will have a partial knowledge about the world. Any node that falls into any of the four petal shaped areas of a sender can be accessed without the need for flooding the request. The only required action is to follow the middle nodes according to their state using a simple heuristic and lead the packet to the destination. If the destination is not included in one of the petal shaped areas, it means that there is no information about it in the sender's table. In this situation, the destination cannot be directly accessed by following the states stored in the nodes.

The technique that we use in order to forward the location inquiry messages to the destination nodes that are not directly accessible, is to send location request messages on the edges of the petals. For a node with four petals, eight location requests will be enough. Any node on the edge of the petal has its own petal shaped regions. If the sought-after destination is reachable, then the request will be routed to it. For example, in Fig. 6, node *S* attempting to enquire the location of D_1, simply initiates a location inquiry message and sends it to another one-hop neighbor in the *NE* petal, because *S* knows that D_1 is located at the *NE* region. For D_2, however, *S* has no clue about its location. Eight location inquiry messages are initiated and sent along the edges of the petals until one of them reaches a node that has knowledge about the relative location of the destination. Note that at least one out of those eight requests will be received by the destination and then replied. The edges of the petal cannot be decided at *S*, but rather it is a conceptual area formed by the collective knowledge of the nodes as a whole system. Node *S* sends the request to the rightmost and leftmost nodes in each region and these nodes forward the request to their own rightmost or leftmost node in the given region according to the information stored in a location inquiry message. A typical location inquiry message consists of entries including the sender's ID, the destination node's ID, current location of the sender, mode variable that can be single or multiple depending on the reachability of the receiver at the sender node's table, region variable which can take *NE*, *NW*, *SE*, or *SW* values, and *RightOrLeft* variable. The two latter entries are ignored when *mode* is set to *single*. If the target node is located in one of the sender's petals, it simply selects one of the nodes that are reporters for the destination node. Suppose that the neighbors located in any region of a node are sorted in polar order. The reporter that is angularly closest to the bisector of the target node's region, is chosen as the forwarder. This heuristic works better than randomly selecting the forwarder node. If the target node moves at extremely fast speeds, the request packet can still follow it using the location information stored in the nodes as clue. Algorithm 5 provides the details of initiating a location request and Algorithm 6 shows the steps that are taken to initiate multiple location inquiry messages.

Algorithm 5. Initiating location inquiry for destination node *N* (Procedure *InitLocInq* (*N*)).

```
1:  LocReq ← Location inquiry request: {senderID=current node's
    ID, receiverID=N, SenderLocation=current node's location,
    mode=single}
2:  if N exists in the table then
3:    R=region where N resides
4:    ReporterList ← List of all of N's reporters
5:    NextNode=The node in ReporterList which is closest to the
    bisector of region R of the current node
6:    Send the ConReq to NextNode
7:  else
8:    InitMultLocInq(N)
9:  end if
```

Algorithm 6. Initiating multiple location inquiry requests for destination node *N* (Procedure *InitMultLocInq* (N)).

```
1:   LocReq[1..8] ← Location inquiry requests:{senderID=current
     node's ID, receiverID=N, SenderLocation=current node's lo-
     cation, Mode=multiple, region=undecided,
     RightOrLeft=undecided}
2:   NEL=leftmost one-hop neighbor in NE
3:   NER=rightmost one-hop neighbor in NE
4:   …
5:   SWL=leftmost one-hop node in SW
6:   SWR=rightmost one-hop node in SW
7:   for all LR in LoqReq do
8:     initialize LR // Simply fill its undecided entries
9:   end for
10:  Send the location requests to NEL,…, SWR (triggers their
     ForwardLocInq procedure) according to the value of Mode
     and RightOrLeft stored in the entries of LocReq[1..8] array
```

After initiating a single or a multiple-path location inquiry, the procedure that is provided in Algorithm 7 will start for any forwarder node to decide the next hop.

Algorithm 7. Forwarding location inquiry (Procedure *ForwardLocInq (LocReq)*)

```
1:   N=LocReq.receiverID
2:   if N=the current node's ID then
3:     receive and process LocReq //target found
4:   else
5:     if Current node is a network boundary node then
6:       send a failure message to node LoqReq.receiverID and
     discard LocReq
7:     else
8:       if LocReq.Mode==single then
9:         N← LocReq.receiverID
10:        if N exists in the table then
11:          R=region where N resides
12:          ReporterList ← List of all of N's reporters
13:          NextNode=The node in ReporterList which is closest
     to the bisector of region R of the current node
14:          Send the ConReq to NextNode
15:        else
16:          RestartLocationInquiry(LocReq) /* This procedure
     simply sends multiple requests if the current node cannot
     decide where to forward the location inquiry*/
```

```
17:             end if
18:         else
19:             if there is a row for $N$ in the table then
20:                 forward the query like the procedure InitLocInq to
    the destination // Means the destination node is accessible
21:             else
22:     forward $LocReq$ according to the values stored in
    $LocReq.region$ and $LocReq.RightOrLeft$
23:             end if
24:         end if
25:     end if
26: end if
```

4. Simulation experiments

In this section, we provide a comparison between PETAL, Grid-based Predictive Location Service (GPLS) (Zhou et al., 2013), Uniform Quorum System (UQS) (Haas and Liang, 1999), and Greed-based Location Service (GLS) (Li et al., 2000) schemes in MANETs.

We implemented the protocols in NS3 (Riley and Henderson, 2010). Various numbers of nodes (500, 1000, 1200, 1500) have been used to compare the performances in different scenarios including low and high density cases. The nodes move according to three different movement patterns that are discussed in Gorawski and Grochla (2014) including Random Waypoint Model which is memoryless (Lee et al., 2000), Gauss–Markov Mobility Model, and Human Mobility Obstacle Model (HUMO) (Roy, 2010) which are not memoryless. The moving nodes have a varying speed of 10, 15, 20, 30, 40 or 50 m/s. The nodes are bounded in square shaped areas with different dimensions (2500 m×2500 m, 5000 m×5000 m, and 10000 m×10000 m), and the complete simulation takes 500, 1000, and 2000 units of time. We also experiment with different radio transmission ranges (20 m, 50 m, and 250 m).

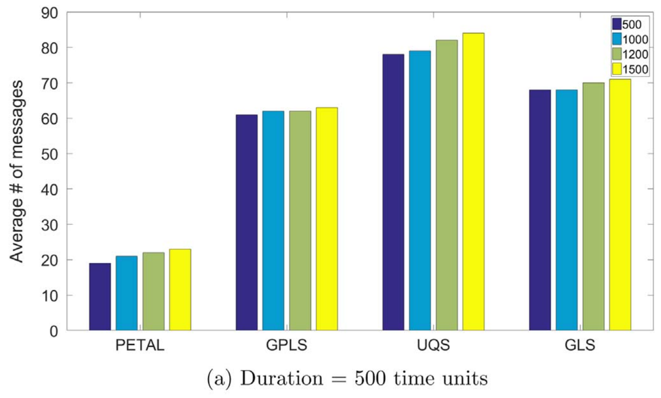
4.1. The number of messages per node

First, we compare the average number of location-related messages sent (initiated or forwarded) per node in PETAL, GPLS, UQS, and GLS. Then we compare the standard deviation of the number of location-related messages to show that PETAL treats nodes more fairly and the load of work is distributed among them more evenly.

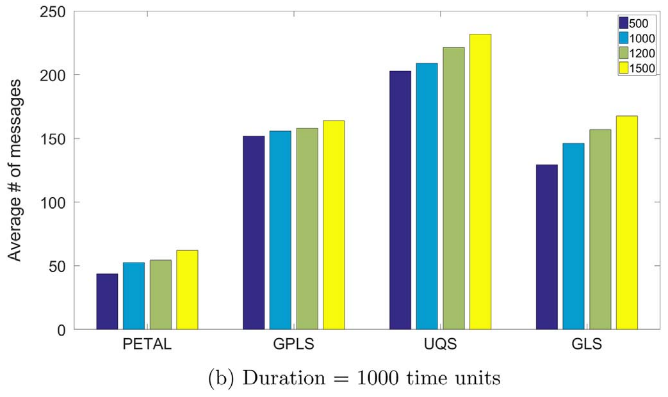
Figs. 7, 8, and 9 show a comparison of the four schemes in terms of the average number of location update messages sent (initiated or forwarded) by nodes using different mobility models and settings. Fig. 10 represents the map that is used for simulating Human Mobility Obstacle Model corresponding to Fig. 9. Note that we do not count the number of messages sent in the initialization phase, since it happens only once in our proposed location service for the whole network lifetime and can be considered as negligible for sufficiently large simulation durations (for example, the setup time is less than 60 time units for 1500 nodes in a 2500 m×2500 m field as illustrated in Fig. 12).

Apart from the average number of messages sent, we compare the standard deviation of the number of messages sent per node to decide which scheme treats individual nodes more fairly. As is illustrated in Fig. 11, in all of the schemes except PETAL, as the number of nodes increases in the MANET, the standard deviation of the number of messages sent by nodes also increases because of the inherent centrality in these location services especially in UQS.

When the area is densely populated, the location servers face a flood of requests while other nodes do not play a significant role in replying to the requests. However, in our approach, there is zero centrality and the location requests are collaboratively processed as the requests are directly routed to the destination. As a result, when the number of nodes grows in the network, the work is more evenly distributed among

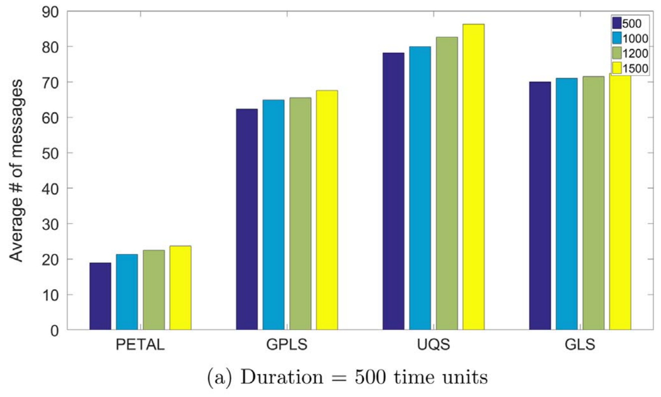


(a) Duration = 500 time units

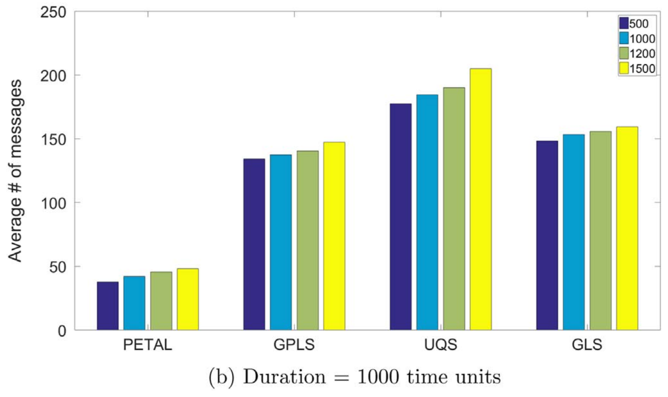


(b) Duration = 1000 time units

Fig. 7. Average number of location update messages sent per node for different number of nodes (2500 m×2500 m field, random waypoint mobility, transmission range=250 m).

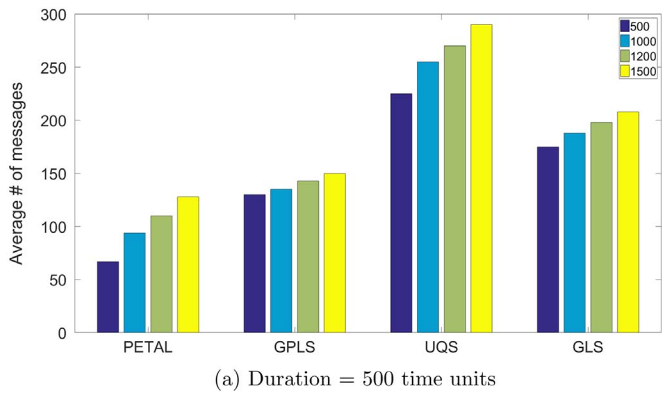


(a) Duration = 500 time units



(b) Duration = 1000 time units

Fig. 8. Average number of location update messages sent per node for different number of nodes (2500 m×2500 m field, Gauss-Markov mobility, transmission range=250 m).



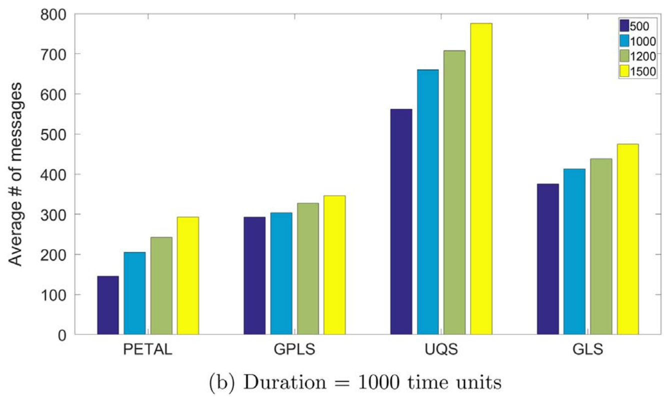


Fig. 9. Average number of location update messages sent per node for different number of nodes that are scattered with a bivariate normal distribution (10,000 m×10,000 m field, Human Mobility Obstacle Model, transmission range=50 m).

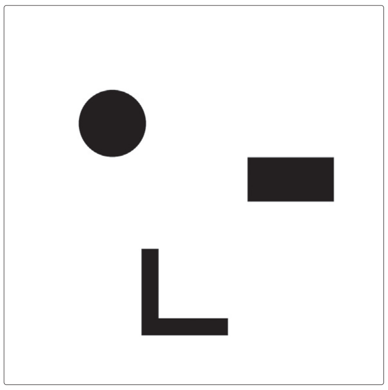
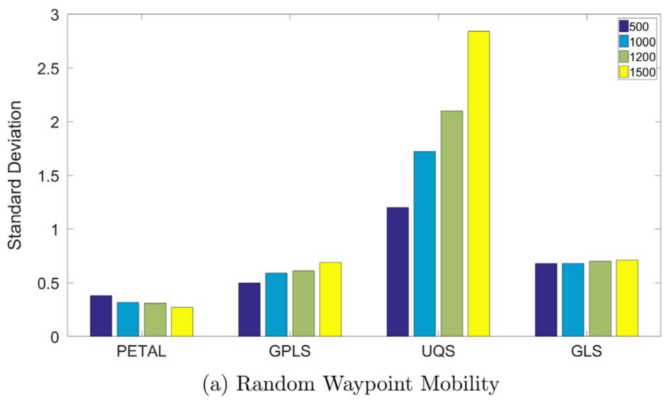


Fig. 10. The 10,000 m×10,000 m square field containing physical blocks that is used for Human Mobility Obstacle Model. The radius of the circle shaped obstacle is 70 m, the dimensions of the rectangular shaped obstacle are 60 m×120 m, and the L-shaped obstacle consists of two overlapping rectangles with dimensions 20 m×120 m.



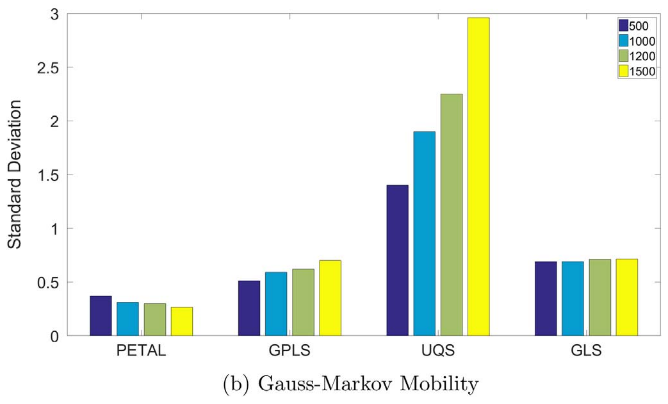


Fig. 11. The standard deviation of the number of location update messages sent per node for different number of nodes (2500 m×2500 m field, transmission range=250 m, time=500 units).

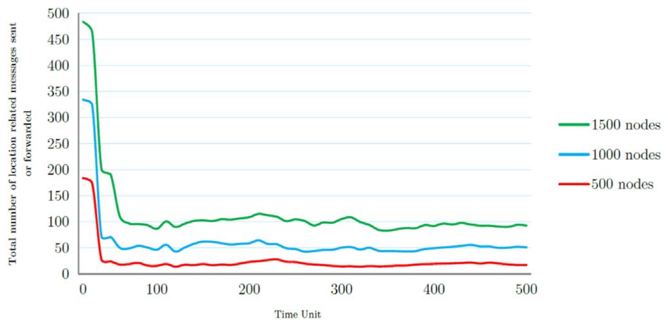


Fig. 12. The number of total location related messages that are initiated or forwarded in any given time unit during the simulation (2500 m×2500 m field, Random Waypoint Mobility, transmission range=250 m).

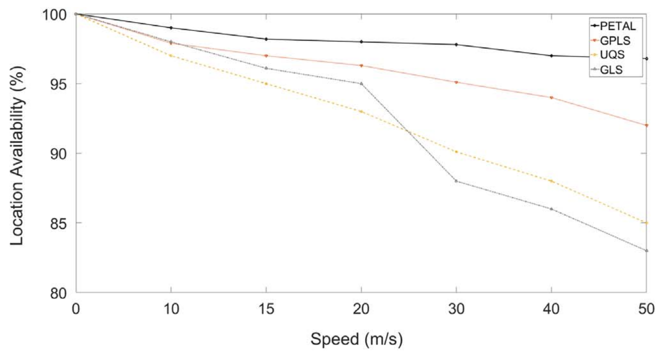
the nodes.

The number of location update messages that are sent in the network in any given time unit during a simulation run is illustrated in Fig. 12. Different curves are provided for different number of participating nodes. The simulation time was 500 time units. As the figure shows, the number of location update messages that are sent is very high when the nodes start to discover their surroundings in the initializalition phase. The initialization phase ends when the nodes do not need to send multi-hop location update messages provided that their relative locations do not change. In other words, if all of the mobile devices stay relatively still, there will be no need to send location update messages, except for the single-hop neighbors for which the update messages must be sent periodically.

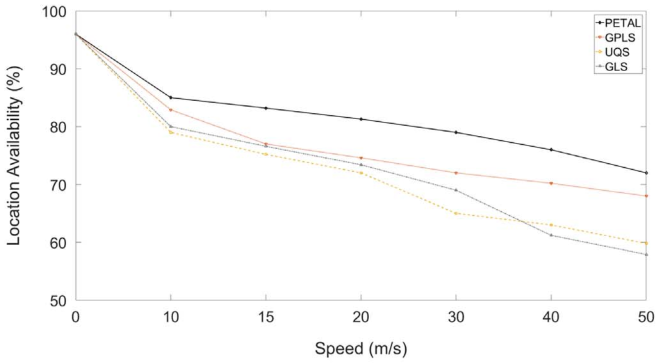
4.2. Location availability

Location availability is another metric that we consider when evaluating our proposed location service scheme. In PETAL, UQS, and GLS, the ratio of the queries (or connection requests) that reach the destination node or a location server and then replied by sending the correct location information of the destination is regarded as location availability. For GPLS, however, the ratio of the queries that are replied correctly is used as the location availability metric. A correct reply in GPLS is the one that returns the grid cell containing the destination node before a sender attempts to communicate with it. If the destination node starts to move and exits its current grid cell before the communication begins, then the corresponding location query is considered unsuccessful.

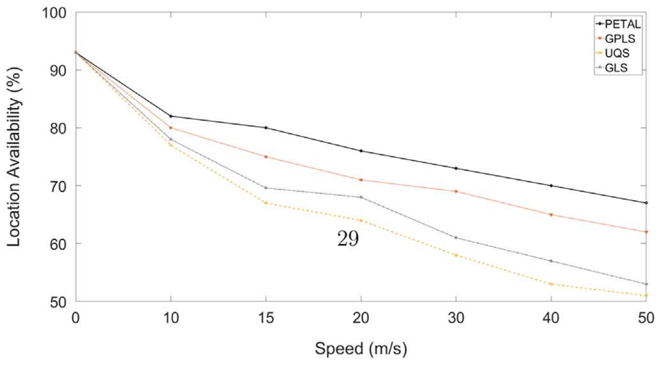
Fig. 13 illustrates a comparison between PETAL, GPLS, UQS, and GLS in terms of location availability. The comparison is made for various values of queried destination speeds, mobility models and



(a) 2500m×2500m field, random waypoint mobility, transmission range = 250m, time = 500 units



(b) 5000m×5000m field, Gauss-Markov mobility, transmission range = 20m, time= 1000 units



(c) 10000m×10000m field, Human Mobility Obstacle Model, transmission range= 50m, time = 2000 units

Fig. 13. Percentage of successful queries destined for target nodes with different moving speeds.

different device ranges. In our scheme, the location update procedure is triggered in the neighbors of a moving node when the relative location is disturbed. Besides, as the query travels towards the destination, the states in the forwarding nodes may also be updated according to the movements of the destination. According to Algorithm 7, the query will follow the moving node using the states stored in the intermediate nodes as the clue, and finally, reach the destination. That is why the ratio of successfully received location queries is higher for PETAL.

4.3. Number of hops traveled for queries

Another metric that we consider in comparing the location service schemes is the average number of hops that a query travels before reaching a location server plus the number of hops that a connection request travels before a connection (e.g., before a TCP connection or data transmission) is established between a source and a destination. The connection requests in PETAL can be directly forwarded to the destination using the collective knowledge of all nodes in the path without initiating a query and sending it to a location server (i.e., without the need for sending the request to a node other than the destination itself). Therefore, for PETAL the number of hops that a query travels until reaching a location server is zero.

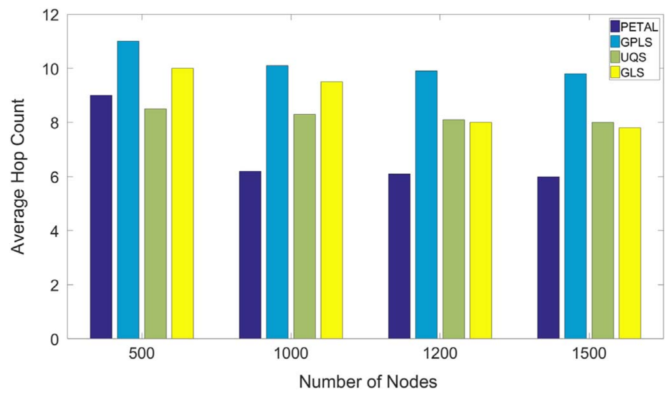


Fig. 14. Average number of hops that a query travels until it reaches the location server (in GPLS and UQS only) plus the number of hops it travels to reach the target node.

According to Fig. 14, the number of hops that it takes for any sender to start a connection with a destination is smaller in PETAL. When the number of nodes is 500 (indicating a relatively sparse network), some of the destinations are not directly accessible and the sender has to initiate eight requests to reach the destination. These requests take longer paths to reach the destination than that of a normal request. As the number of nodes increases, the average number of records in any sender's list also increases and the necessity of initiating multiple requests diminishes.

4.4. Swarm mobility

We repeat the same experiment as in 4.1 this time with swarm mobility. The scenario is as follows: all of the nodes while moving in the same pattern with respect to each other, move towards a virtual target located in the north direction as a swarm with constant speed. We compare the four protocols in terms of the number of location update messages initiated or forwarded in two different speeds (Fig. 15). In both cases, we use 1000 nodes.

5. Memory cost and time complexity

Each device needs some physical memory to store the lists that contain the IDs of other devices. If we use IP addresses as node IDs, then the required memory for storing each address is 4 bytes. With N

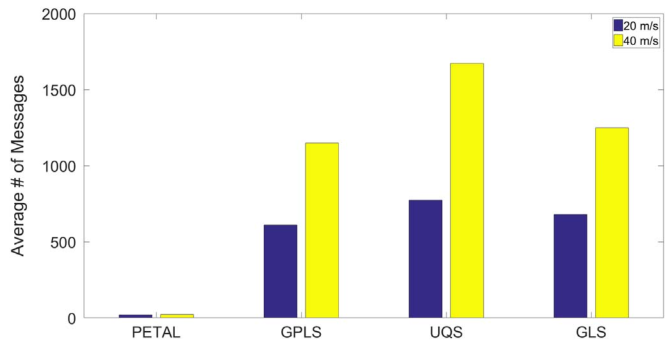


Fig. 15. Average number of location update messages sent per node for different speeds of swarm movement (2500 m×2500 m field, random waypoint mobility, transmission range=250 m, time=500 units).

number of total nodes in the network, there can be at most $N-1$ nodes in all petals. It means that the amount of required memory is at most 4×(N-1) bytes (which is of $O(N)$) plus some more memory required for reporters, virtual neighbors and repeated nodes (e.g., any node that is located in the vertical or horizontal line that must be placed in two petals). With the abundance of memory in modern wireless devices this level of memory requirement is not restrictive.

The only procedures that require ID lookup are initiating/forwarding the location inquiry and deleting an entry. Each node can find the next hop or the node to be deleted from its list in $O(\log m)$ time where m is the number of its immediate neighbors (m itself is bounded by N) such that neighbors are lexicographically sorted according to the polar angle and distance with respect to the current node.

6. Conclusion and future work

In this work, we propose PETAL, a distributed location service scheme with no centrality. Initially, the nodes start to communicate and collect knowledge about the surroundings which is a bandwidth consuming process but happens only once. Once the initial data is collected as states in each node, there remains less need for location updates since the location information is stored as relative instead of absolute. The collective knowledge of the nodes is combined to pinpoint the target instead of sending a query to a location server and waiting for a response. The states stored in the nodes are prone to be affected mostly by closer nodes. The state stored in a node provides a petal-shaped view of the whole network for that node. As the density of the network increases, the communication between any source and destination becomes faster because the areas (i.e., the petals) discovered by individual nodes grow.

If we consider the recently popular location service schemes, they normally face so much overhead if the network itself as a whole is mobile which is referred to as swarm movement. Our approach will not suffer too much in such situations. It is due to the fact that only relational arrangement of the nodes rather than their absolute position is taken into account.

The methods that apply a hash function on the ID of the nodes to find their location service grid require ID management for the nodes. The translation of node IDs into real IP or MAC addresses (and vice versa) needs to be addressed in these schemes. However, in our approach, any variation of node ID (IP, MAC, etc) can represent a node regardless of its initial or current location. Moreover, using consistent hashing to assign nodes into location servers adds more complexity to the location service as the quality of the deployed hashing method may affect the load balancing and availability aspects of these location services.

The most important aspect of our approach is that it does not strictly require the nodes to know their exact location. That is, any system or method that can approximate the relational location of a group of one-hop neighbors can be adequate in PETAL with possibly some modification on the current version. By approximating the relational location, we mean the ability of the nodes to discover the regions in which their neighbors reside. As a future work, this problem can be studied. If such a method that does not rely on the positioning systems exists, then our proposed scheme has both the advantages of a geographical routing approach and the advantage of not being solely dependent on highly energy demanding and non-ubiquitously operable positioning systems like GPS. It is worthy to mention that even without exact location information, routing is still possible by sending the data packets quite similar to the way that connection requests (or location queries) are sent. However, choosing the right edge in the case of multi-path routing for data packets and deciding which nodes are located at the network boundary are more challenging without knowing the exact location information. These mentioned problems can also be studied in a future work.

References

Ahmed, Sabbir, Karmakar, Gour C., Kamruzzaman, Joarder, 2009. Hierarchical adaptive location service protocol for mobile ad hoc network. IEEE Wirel. Commun. Network. Conf., 1–6.

Alotaibi, Eiman, Mukherjee, Biswanath, 2012. A survey on routing algorithms for wireless Ad-Hoc and mesh networks. In: Computer Networks, vol. 56, issue 2, pp. 940–965, February ISSN 1389–1286, http://dx.doi.org/10.1016/j.comnet.2011.10.011

Arora, Nisha, Jangra, Ajay, 2012. GLAAR: geographic location aware adaptive routing in mobile ad hoc networks (MANETs). Int. J. Comput. Appl. 50 (22).

Basagni, Stefano, Chlamtac, Imrich, Syrotiuk, Violet R., Woodward, Barry A., 1998. A distance routing Effect Algorithm for Mobility (Dream). In: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98), pp. 76–84.

Bok, Kyoungsoo, Yoon, Sooyong, Lim, Jongtae, Yoo, Jaesoo, 2016. Grid based Enhanced Routing Scheme in MANET. KSII Trans. Internet Inf. Syst. 10 (5).

Cadger, F., Curran, K., Santos, J., Moffett, S., 2013. A survey of geographical routing in wireless ad-hoc networks. IEEE Commun. Surv. Tutorials 15 (2), 621–653, Second Quarter.

Cheng, Rei-Heng, Huang, Chiming, 2012. Efficient Prediction-based location updating and destination searching mechanisms for geographic routing in mobile ad hoc networks. J. Inf. Sci. Eng. 28 (1), 115–129.

Clausen, Thomas, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, Laurent Viennot, 2003. Optimized link state routing protocol (OLSR). Available at ⟨http://www.ietf.org/rfc/rfc3626.txt⟩.

Derhab, Abdelouahid, Badache, Nadjib, 2008. Balancing the tradeoffs between scalability and availability in mobile ad hoc networks with a flat hashing-based location service. Ad Hoc Netw. 6 (7), 1013–1030.

El Defrawy, Karim, Tsudik, Gene, 2011. Privacy-preserving location-based on-demand routing in MANETs. IEEE J. Select. Areas Commun. 29 (10), 1926–1934.

Giordano, Silvia, Hamdi, Maher, 1999. Mobility management: the virtual home region. EPFL, Lausanne, Switzerland, Technical Report, SSC/1999/037, 1999.

Gorawski, Michal, Grochla, Krzysztof, 2014. Review of mobility models for performance evaluation of wireless networks, In: Springer International Publishing, Man–Machine Interactions, vol. 3, pp. 567–577.

Haas, Zygmunt J., Liang, Ben, 1999. Ad hoc mobility management with uniform quorum systems. IEEE/ACM Trans. Network. 7 (April (2)), 228–240.

Hwang, Haesu, In, Hur, Hyunseung Choo, 2009. GOAFR plus-ABC: geographic routing based on adaptive boundary circle in MANETs. In: Proceedings of the International Conference on Information Networking (ICOIN '09). IEEE, Chiang Mai, Thailand, pp. 1–3.

Jain, Sonam, Sahu, Sandeep, 2012. Topology vs position based routing protocols in mobile ad hoc networks: a survey. Int. J. Eng. Res. Technol. 1 (3).

Käsemann, Michael, Hartenstein, Hannes, 2002. Analysis of a location service for position-based routing in mobile ad hoc networks. In: Proceedings of the Mobile Ad-Hoc Netzwerke, 1. Deutscher Workshop über Mobile Ad-Hoc Netzwerke (WMAN 2002), pp. 121–133.

Kang, Mi-Seon, Dong-Won Kum, Jae-Seung Bae, You-Ze Cho, Anh-Ngoc Le, 2012. Mobility Aware Hybrid Routing protocol for mobile ad hoc network. In: The International Conference on Information Network, pp. 410–414.

Karp, Brad, Hsiang-Tsung Kung, 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking. ACM, Boston, Massachusetts, pp. 243–254.

Lee, Sung-Ju, William, Su, Julian Hsu, Gerla, Mario, Bagrodia, Rajive, 2000. A performance comparison study of ad hoc wireless multicast protocols. In: Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), vol. 2, pp. 565–574.

Li, J., Jannotti, J., DeCouto, D.S.J., Karger, D.R., Morris, R., 2000. A scalable location service for geographic ad hoc routing. In: Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), pp. 120–130.

Li, Jun, Zhou, Yifeng, Lamont, Louise, Yu, F. Richard, Rabbath, Camille-Alain, 2012. Swarm mobility and its impact on performance of routing protocols in MANETs. Comput. Commun. 35 (6), 709–719.

Liu, Jianqi, Wan, Jiafu, Wang, Qinruo, Deng, Pan, Zhou, Keliang, Qiao, Yupeng, 2016. A survey on position-based routing for vehicular ad hoc networks. Telecommun. Syst. 62 (1), 15–30.

Mahmood, Baban A., Manivannan, D., 2015. Position based and hybrid routing protocols for mobile ad hoc networks: a survey. Wirel. Personal Commun. 83 (2), 1009–1033.

Marwane, Ayaida, Barhoumi, Mohtadi, Fouchal, Hacéne, Yacine Ghamri-Doudane, Lissan Afilal, 2014. Joint routing and location-based service in VANETs. In: J. Parallel Distrib. Comput. 74 (2) 2077–2087, ISSN 0743-7315, http://dx.doi.org/10.1016/j.jpdc.2013.10.004.

Mauve, Martin, Widmer, Jorg, Hartenstein, Hannes, 2001. A survey on position-based routing in mobile ad hoc networks. IEEE Netw. 15 (6), 30–39.

Perkins, Charles, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (AODV) routing. No. RFC 3561. 2003

Preparata, Franco P., Ian Shamos, Michael, 1985. Computational Geometry: An Introduction, 38–45.

Riley, George F., Henderson, Thomas R., 2010. The ns-3 network simulator. In: Modeling and Tools for Network Simulation. Springer Berlin Heidelberg, pp. 15–34.

Roy, Radhika Ranjan, Handbook of Mobile Ad Hoc Networks for Mobility Models. Springer Science & Business Media, 2010.

Saleet, Hanan, Basir, Otman, Langar, Rami, Boutaba, Raouf, 2010. Region-based

location-service-management protocol for VANETs. IEEE Trans. Veh. Technol. 59 (2), 917–931.
Shen, H., Zhao, L., 2013. ALERT: an anonymous location-based efficient routing protocol in MANETs. IEEE Trans. Mob. Comput. 12 (June (6)), 1079–1093.
Stojmenovic, I., Liu, D., Jia, X., 2008. A scalable quorum-based location service in ad hoc and sensor networks. Int. J. Commun. Netw. Distrib. Syst. 1 (1), 71–94.
Stojmenovic, Ivan, 1999. Home agent based location update and destination search schemes in ad hoc wireless networks. Department of Computer Science, SITE, University of Ottawa, TR-99-10.
Wang, Zijian, Bulut, Eyuphan, Szymanski, Boleslaw K., 2013. Energy-efficient location services for mobile ad hoc networks. Ad Hoc Netw. 11 (1), 273–287.
Woo, Seung-Chul M., Singh, Suresh., 2001. Scalable routing protocol for ad hoc networks. Wirel. Netw. 7 (5), 513–529, Harvard.
Wu, C., Ohzahata, S., Ji, Y., Kato, T., 2014. Toward a totally distributed flat location service for vehicular ad hoc networks. In: IEEE 79th Vehicular Technology Conference (VTC Spring), Seoul, pp. 1–6, http://dx.doi.org/10.1109/VTCSpring.2014.7023128
Zhou, Jipeng, Liu, Liangwen, Liao, Guofang, Lu, Jianzhu, 2013. Predictive and fault-tolerant location service in mobile ad hoc networks. Wirel. Pers. Commun. 71 (4), 3115–3130.

Amir Rahimzadeh Ilkhechi received his B.S. degree in Information Technology from the University of Tabriz, East Azerbaijan, Iran, in 2011, and then joined as an M.S. student to the Department of Computer Engineering at Bilkent University (co-supervised by Associate Professor Ibrahim Korpeoglu and Professor Özgür Ulusoy), Ankara, Turkey, in 2012. He is currently a PhD student at the Department of Computer Science at Duke University, North Carolina, United States. His research interests cover Big Data Processing Platforms, Performance Modeling, Computer Networks, Distributed Systems, Cloud Computing, Graph Theory and Computational Geometry.

Ibrahim Korpeoglu received his Ph.D. and M.S. degrees from University of Maryland at College Park, both in Computer Science, in 2000 and 1996, respectively. He received his B.S. degree (summa cum laude) in Computer Engineering, from Bilkent University in 1994. Since 2002, he is a faculty member in the Department of Computer Engineering of Bilkent University. Before that, he worked in several research and development companies in USA including Ericsson, IBM T.J. Watson Research Center, Bell Laboratories, and Bell Communications Research (Bellcore). He received Bilkent University Distinguished Teaching Award in 2006 and IBM Faculty Award in 2009. His research interests include computer networks and systems, wireless networks, and cloud computing.

Uğur Güdükbay is a professor at Department of Computer Engineering, Bilkent University, Ankara, Turkey. He received his Ph.D. degree in Computer Engineering and Information Science from Bilkent University, Ankara, Turkey, in 1994. Then, he conducted research as a postdoctoral fellow at Human Modeling and Simulation Laboratory, University of Pennsylvania, Philadelphia, Pennsylvania.

Özgür Ulusoy received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. He is currently a Professor in the Computer Engineering Department of Bilkent University in Ankara, Turkey. His current research interests include web databases and web information retrieval, multimedia database systems, social networks, and cloud computing. He has published over 120 articles in archived journals and conference proceedings.