Proc. 25th International Symposium on Computer and Information Sciences
22-24 September 2010, London, UK
Lecture Notes in Electrical Engineering, vol. 62, Spinger, pp. 163-168

An Energy Efficient Location Service for Mobile Ad Hoc Networks

Zijian Wang[1], Eyuphan Bulut[1] and Boleslaw K. Szymanski [1],

[1] Department of Computer Science, Rensselaer Polytechnic Institute,
Troy, NY 12180 USA
{wangz, bulute, szymansk}@cs.rpi.edu

Abstract. Location based routing protocols are heavily dependent on location services which provide the position information of the desired destination node. Seldom location service schemes include energy efficiency metrics when evaluating their performance in forwarding location update and query packets. We propose a novel location service that aims at decreasing the distance traveled by the location update and query packets and, thus, at reducing the overall energy cost. Simulation results are presented to demonstrate that the new scheme achieves energy efficiency while maintaining all the other performance metrics comparable to the previously published algorithms.

Keywords: Location service, mobile ad hoc networks, routing

1 Introduction

A critical issue for location based routing protocols is to design efficient location services that can track the locations of mobile nodes. The earliest of location service protocols were based on flooding-based approaches. Then, to restrict the resulting location update and query flooding, quorum-based protocols were proposed. Recently, hashing-based protocols have been proposed, which can further be divided into flat or hierarchical ones. In the first category [1-2], each node's identifier is mapped to a home region consisting of one or more nodes within a fixed location. However, a large overhead is introduced during the location update procedure and frequent location queries and replies cause early death of the nodes within such home region. In the second category [3-5], the network area is recursively divided into a hierarchy of squares. For each node, one or more nodes in each square at each level of the hierarchy are chosen as its location servers. Thus, the location update cost is significantly reduced and location servers are scattered all over the network.

However, the main goal of the hierarchical hashing-based protocols is just to find the location of the destination nodes. They seldom take energy efficiency issue into consideration during design of forwarding location update and query packets. We propose a novel location service scheme which attempts to decrease the distance

traveled by the location update and query packets and, thus, to reduce the overall energy cost.

2 Energy Efficient Location Service

2.1 Network Partition and Coordinate System

Each node knows its own position and the positions of its neighbors. The whole square network area is recursively divided into a hierarchy of squares which are known to each node in the network. At the top level, the entire area is called a level-N square. Each of level-i ($1 < i \le N$) squares is further divided into four level-(i-1) quadrants, until the entire region is divided into $4^{(N-1)}$ level-1 squares. Given L as the side length of the whole network area, the side length of a level-i square is $L_i = L/2^{N-i}$. Fig. 1(left) illustrates an example of a 4-level hierarchy network.

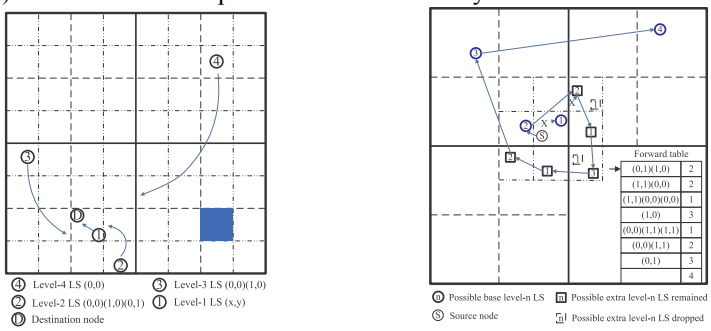


Fig. 1. An example for a 4-level hierarchy network (left) and location query procedure and forward table (right)

Using the lower left point as the origin, we define the address of level-i square as a sequence of coordinate pairs $(a_x^{N-1}, a_y^{N-1})...(a_x^i, a_y^i)$ ($a_{x|y}^i$ in short) computed as: $a_{x|y}^i = (s_{x|y}^i - \sum_{k=1}^{N-i-1} L_{N-k} \bullet a_{x|y}^{N-k}) / L_i$, where (s_x^i, s_y^i) ($s_{x|y}^i$ in short) is the lower left coordinate of the level-i square. For example, the address sequence for the marked level-1 square in Fig. 1(left) is (1,0)(1,0)(0,1). Inversely, the lower left coordinate of the level-i square can be computed as follows: $s_{x|y}^i = \sum_{k=1}^{N-i} L_{N-k} \bullet a_{x|y}^{N-k}$. Given a node's coordinate *(n_x, n_y)*, the address sequence $(na_x^{N-1}, na_y^{N-1})...(na_x^i, na_y^i)$ ($na_{x|y}^i$ in short) of the level-i square to which this node belongs is calculated as: $na_{x|y}^i = floor((n_{x|y} - \sum_{k=1}^{N-i-1} L_{N-k} \bullet na_{x|y}^{N-k}) / L_i)$.

2.2 Location Update

Each node selects one level-i location server in each level-i square in which it resides. The position of the level-i location server (ls_x^i, ls_y^i) (referred to as location server point) for each node in level-i square is determined as: $(ls_x^i, ls_y^i) = (s_x^i, s_y^i) + hash(ID, L_i)$, where ID is the unique identifier of the node. Hash is a global function known to each node that maps a node's ID to a relative position in a level-i square.

In our method, each location server maintains a list of nodes whose location information it stores. Each element of the list stores the following information: node ID (32 bits), location server level (log_2N bits), location information (introduced in the following), and expiration time (32 bits). Please note that the destination node's exact location information is only stored at level-1 location servers. At all other levels, the location servers only store the address sequence of the square in which the level-(i-1) location server resides, as shown in Fig. 1(left). Thus, 1) the memory usage is reduced; 2) the size of the location update packet is also reduced; 3) the location information at level-i location server needs to be updated only when the destination node moves out of the corresponding level-(i-1) square, which significantly reduces the frequency of location updates, thereby saving a lot of energy.

In previous methods, all the location update packets are sent to location servers individually. In our method, if one node needs to send a location update to more than one location server, it first calculates the distances traveled by the update messages both for sending them to each desired location server individually (referred as d-indiv) and sending them in one packet which traverse all the desired location servers (referred as d-one). If d-indiv is smaller than d-one, the location update messages are sent to each desired location server individually. Otherwise, all the update messages are integrated into one packet that is forwarded according to a forward table which indicates the sequence of location servers to be visited. Traversing multiple points in a plane is a kind of Hamiltonian path problem. We use a simple greedy solution in which the next visited node is always the nearest one to the currently visited node. Any intermediate node greedily forwards location update packet to the neighbor nearest to the position of the next location server in the forward table. Once the location update packet reaches a location server at certain level, the corresponding location information will be stored at this server and the next entry in the forward table pops up. All the outdated table entries are deleted.

2.3 Location Query

When the source node resides within a level-s (predefined parameter, we set it to 1) square that is beside the boundary of any level-h (predefined parameter, we set it to *N-1*) square, the search proceeds as follows. The source node calculates all candidate location server points (from level-N to level-1) that fall into the adjacent level-s squares (we refer to them as candidate adjacent squares) located on the opposite side of the boundary of a level-h square. If any level-k candidate location server point is found in each adjacent level-s square, there is no need to find level-i (i<k) candidate

location server points in the same level-s square. Hence, only the highest candidate location servers in every adjacent square searched need to be found (we refer to them as *extra location servers*). Then, the source node follows HIGH-GRADE method [5] (abbreviated HGM) and calculates, from the lowest level to the highest, all candidate level-i location server points of squares in which source node resides until such level is reached that its square contains also the destination node (we refer to them as *base location servers*). Both of these two kinds of location points (if the extra location servers exist) are sorted into a list that can be traversed by a path starting from the source node, using the same greedy Hamiltonian path method as used in sending location update packets. If any high level base location server is in front of low level base location server, the lower one is deleted from the list because if the location query packet checks the high level base location server points first, then there is no need to check the low level base location server points. But for extra location servers, we need to check all of them in each adjacent level-s square, so we keep all of them.

An example in which the source node resides in the level-1 square which is beside the boundary of level-3 square is shown in Fig. 1(right). The source node calculates all candidate location server points in the adjacent squares (there are at most five of them, as shown in Fig. 1(right)). There are two candidate location server points in adjacent level-1 square (1,0)(0,1)(0,1) (one is level-1, the other is level-3). Only the level-3 one will be kept. Then, all candidate location servers are sorted in the order shown in Fig. 1(right), as defined by the path traversing from the source node. Since the level-2 base location server is in front of level-1 base location server on this sorted list, the level-1 base location server will be removed from the list.

4 Simulations

We used NS-2.33 to evaluate our scheme and compared it with the HIGH-GRADE method [5]. The whole network is deployed over a 1000 m by 1000 m area partitioned into 4-level squares. The following metrics are evaluated: (1) the total distance (measured in meters and hops) traveled by all location update packets for all nodes; (2) the average distance traveled by location query packets; (3) the average distance traveled by location query packet for specific destination node (for this kind of location query, we select the source node that resides within a level-1 square that is beside the boundary of level-3 square; moreover, there is at least one location server for the destination node residing in the adjacent level-1 square which is also beside the boundary of level-3 square on the opposite side); (4) the average energy usage; and (5) the location query success rate.

In static network scenario, we keep the average number of neighbor nodes constant and vary the number of nodes from 200 to 600. For each randomly generated topology, nodes send location update packets at first, then, 20 randomly location queries start. In the mobile network scenario, nodes move according to the random way-point model with no pause time. We keep the number of nodes at 400 but vary the maximum nodal speed V_{max} from 2.5 m/s to 7.5 m/s.

Table 1 gives the average results for metric (1) to (3). Clearly, the location update cost for our method is much lower than for HGM. This is mainly because the location

update messages in our method could be sent in one packet. It is also clear that the cost of a specific location query in our method is much lower than for HGM. This advantage is the result of its properties: quick search within the adjacent low level squares and visiting higher level candidate location server first. However, for randomly selected location queries, the costs of the two compared methods are almost the same. The reason is that the quick search within adjacent low level squares does not always find the desired location servers. In such cases, the distance traveled by the quick search just increases the cost without any benefit. However, our method still benefits from visiting higher level candidate location servers first, preventing unnecessary travel by a location query packet.

Table 1. Simulation results for the first three metrics

Static network scenario for metric (1)

Node Number	200	400	600
Our method: distance (hops)	154129.1(1259.6)	302726.2(3285.0)	452976.9(5845.0)
Compared method: distance (hops)	227285.2(1736.0)	435527.5(4495.4)	649542.7(8037.4)

Mobile network scenario for metric (1)

V_{max} (m/s)	2.5	5	7.5
Our: distance (hops)	324755.9(3581.9)	352415.3(3903.1)	388754.6(4528.7)
Compared: distance (hops)	475739.1(4946.3)	529158(5736.5)	575242.9(6327.1)

Static network scenario for metric (2)

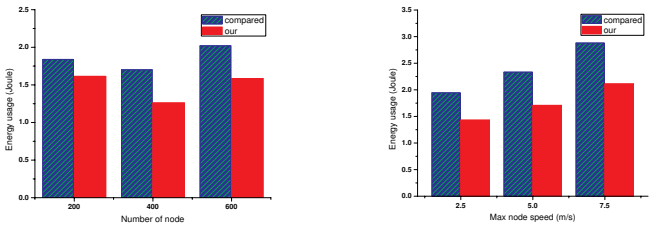
Node Number	200	400	600
Our method: distance (hops)	1288.6(9.9)	1315.5(13.9)	1317.1(16.6)
Compared method: distance (hops)	1299.3(10.0)	1303.4(13.7)	1365.5(17.2)

Mobile network scenario for metric (2)

V_{max} (m/s)	2.5	5	7.5
Our method: distance (hops)	1021.6(10.7)	1017.5(11.0)	967.3(10.4)
Compared method: distance (hops)	959.2(10.1)	1055.1(11.3)	1080.8(11.7)

Static network scenario for metric (3)

Node Number	200	400	600
Our method: distance (hops)	878.6(7.2)	966.8(10.3)	1098.3(14.4)
Compared method: distance (hops)	1443.9(11.2)	1424.8(14.5)	1497.1(18.7)



Static network scenario Mobile network scenario

Fig. 2. Simulation results for metric (4), the energy usage

The energy usage for both methods is shown in Fig. 2. The energy usage for our method is lower, in the range of 74% to 88% of the HGM energy use. Table 2 shows the location query success rate results. This rate is a little higher for our method.

Table 2. Simulation results for metric (5), the location query success rate

Node Number	200	400	600	V_{max} (m/s)	2.5	5	7.5
Our method:	99%	96%	96%	Our method:	80%	72%	59%
Compared method:	91%	94%	93%	Compared method:	74%	69%	56%

5 Conclusion

In this paper, we introduced a novel location service that aims at reducing the overall energy cost by decreasing the distance traveled by the location update and query packets. Extensive simulations are performed to demonstrate that the new scheme achieves energy efficiency while maintaining all the other performance metrics.

Acknowledgement

Research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Woo, S. C., Singh, S.: Scalable Routing Protocol for Ad Hoc Networks. ACM Wireless Networks, vol. 7(5), pp. 513--529 (2001)
2. Das, S. M., Pucha, H., Hu, Y. C.: Performance Comparison of Scalable Location Services for Geographic Ad Hoc Routing. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp. 1228--1239. IEEE Press, New York (2005)
3. Yan, Y., Zhang, B. X., Mouftah, H.T., Ma, J.: Hierarchical Location Service for Large Scale Wireless Sensor Networks with Mobile Sinks. In: IEEE Global Telecommunications Conference (GLOBECOM), pp. 1222--1226. IEEE Press, New York (2007)
4. Ahmed, S., Karmakar, G. C., Kamruzzaman, J.: Hierarchical Adaptive Location Service Protocol for Mobile Ad Hoc Network. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 1--6. IEEE Press, New York (2009)
5. Yu, Y. Z., Lu, G.-H., Zhang, Z.-L.: Enhancing Location Service Scalability with HIGH-GRADE. In: IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp. 164--173. IEEE Press, New York (2004)