Writing on Water, A Lightweight Soft-State Tracking Framework for Dense Mobile Ad Hoc Networks

Xuming Lu Murat Demirbas
Computer Science and Engineering Department
University at Buffalo, SUNY
Email: xuminglu, demirbas@cse.buffalo.edu

***Abstract*— In a mobile ad hoc network, tracking protocols need to deal with—in addition to the mobility of the target— the mobility of the intermediate nodes that maintain a track toward the target. To address this problem, we propose the MDQT (Mobility-enhanced Distributed QuadTree) tracking framework. MDQT employs a static *cell abstraction* to mask the mobility of the nodes and provide the illusion of a logical static network overlaid on the mobile network. MDQT implements this virtual static network layer in a lightweight/communication-free manner by exploiting the *soft-state principle* and the *snooping feature* of wireless communication.**

Using simulations, we study the effects of the mobility speed and the percentage of mobile nodes on the performance of our MDQT framework. We find that even at very high mobility speeds (50 meters per second), low update rates (1 update per second), and 100% node mobility, the success rate of MDQT tracking is above 85% and the latency is comparable with that of static networks.

I. Introduction

Location tracking problem is of great importance in surveillance, security, and information systems. Most of the work on tracking focus on an environment where all the nodes are static and only the target node is mobile [4], [5], [17], [20]. Even for this static network setting, the tracking problem is nontrivial. Flooding based solutions are unscalable as they impose a large control traffic overhead and devour the network bandwidth [14]. Centralized solutions where location queries are answered from a central basestation node are undesirable performance-wise since they violate the *distance-sensitivity* property: the querier may be closer to the target, but yet still have to communicate all the way to the central basestation. Furthermore, distance-sensitivity is required even for correctness of the tracking application as it has been shown that, for satisfying optimality constraints, the latency with which an interceptor requires information about the intruder it is tracking depends on the relative locations of the two: the closer the distance, the smaller the latency [3]. To achieve distance-sensitivity, a distributed location lookup directory needs to be implemented to provide the location of the tracked node to any querying node in the network [1], [5], [6].

The tracking problem becomes most challenging in the Mobile Ad Hoc Networks (MANets) setting, where all the nodes may potentially be moving at any time. Since the intermediate nodes that maintain the tracks to the target and that relay protocol messages are also allowed to be mobile, maintaining a distributed location lookup/tracking directory over the network is hard, let alone doing it in a bandwidth-efficient and distance-sensitive manner. We address these challenges for tracking in MANets with our MDQT (Mobility-enhanced Distributed Quad-Tree) framework. MDQT provides efficient location updates as well as distance-sensitive latency. Our simulation results show that even at very high mobility speeds (50 meters per second), low update rates (1 update per second), and 100% node mobility, the success rate of MDQT tracking is above 85% and the latency is not much worse than that of static networks. We summarize the key ideas behind the success of MDQT next.

To implement MDQT, we provide an efficient implementation of a static *cell abstraction* to mask the mobility of the nodes and present an illusion of a logical static network overlaid on the MANet. This is achieved by mapping mobile nodes that fall within a geographic area (cell) to model a *virtually static node* [7] for that area. We use *loose-synchronization* among the mobile nodes to implement this virtual static node model in a lightweight manner. In fact, in our implementation, nodes in a cell do not explicitly coordinate at all, and the maintenance of the virtual static node is achieved in a communication-free manner. In this loose-synchronization approach, new nodes catch up with the state information of other nodes opportunistically by snooping ongoing wireless broadcast communication.

Armed with this overlaid static virtual node layer, we adapt our earlier work on distance-sensitive tracking [6] in Wireless Sensor Networks(WSNs) to the MANet domain to embed a hierarchical in-network tracking structure over the virtual node layer. In order to maintain the tracking information for the targets in a lightweight manner, we employ the soft-state mechanism [9], [15]. Periodic location updates advertised by the targets refresh and keep-alive the tracking information at higher layers of the hierarchy, and the stale tracks are removed by timeouts automatically. The advantage of the soft-state approach is that it obviates the need for explicit messages for handouts among nodes and for removing old tracks, and hence leads to a simpler and more robust system. We further optimize the update frequency of our soft-state based track maintenance by devising an *intermittent update forwarding* mechanism, which enables higher level nodes in the hierarchy to update with exponentially less frequency than those at the lower levels. Our simulations show that in dense MANets, our loose-synchronization and soft-state based implementation achieves good scalability in creating the illusion of a static

logical overlay to the face of high mobility speeds and low update frequency.

Another key idea behind the success of MDQT tracking is the *query-forwarding* mechanism used for dealing with the target misses at high speeds. In the high mobility speed setups a query may arrive to the location indicated by the tracking structure but the target may have moved away from there in the meanwhile. To address these cases, the query forwarding mechanism prescribes restarting the query from this new location, by leveraging on the distance-sensitivity of the MDQT lookups. The idea here is that, since the new location is closer to the target than the original query location, the forwarded query succeeds with higher probability in finding the target.

Contributions. Our main contributions in this paper are as follows:

- We provide a distance sensitive target tracking service, MDQT, for fully mobile networks.
- We present a loose-synchronization based approach for maintaining a virtual static cell abstraction for MDQT over the physical mobile nodes in a communication-free manner.
- We present a soft-state implementation of the MDQT lookup directory to keep the tracking framework lightweight and simple.
- We provide an experimental spectrum study, analyzing the performance from low speeds and few mobile nodes, to high speeds and all mobile nodes. We show that the performance degradation of MDQT is graceful with respect to the mobility speed, and is not sensitive to the percentage of mobile nodes.
- We enhance the wireless sensor network simulator Prowler to enable support for node mobility. Information about this enhancement is available at http://ubicomp.cse.buffalo.edu/mdqt.

The rest of the paper is organized as follows: Section II describes the MDQT framework. We discuss implementation considerations and tuning of parameters in Section III. Section IV presents the simulation results and shows MDQT's performance in diverse mobility environments. We discuss optimizations and extensions in Section V. Section VI reviews the related research and shows how MDQT compares with these prior work. Finally, we conclude the paper in Section VII.

II. Mobility-enhanced Distributed-Quadtree

Here we first describe the DQT structure that MDQT is built on, and then introduce MDQT operations for location update and querying. Finally we discuss how the cell abstraction is maintained as soft-state.

A. DQT structure

MDQT is built on our previous work DQT [6], which overlays a virtual grid in a field as shown in Figure 1. The field is partitioned hierarchically, each partition divides a region into 4 sub-regions, which are encoded as 0,1,2,3 corresponding

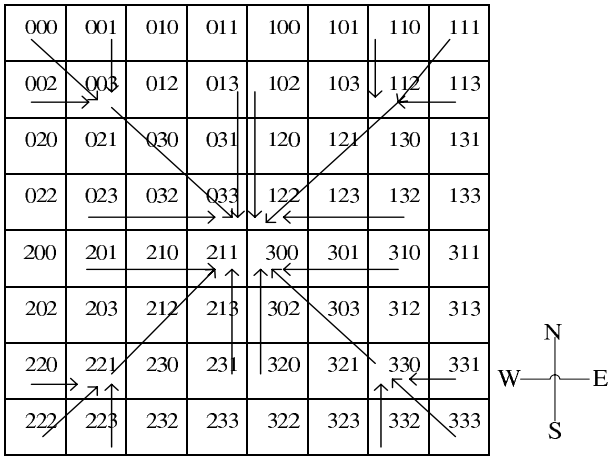

Fig. 1. MDQT coding with 3 levels

to NW, NE, SW and SE partitions. As such, each smallest partition is assigned an ID which uniquely identifies a region. Each sensor node can calculate this logical ID given its location(x,y)(E.g, via GPS). We use this addressing scheme to encode the location information of a node in DQT. Figure 1 illustrates the addresses of the nodes in a partitioned region with 3 partition levels.

DQT maintains a minimal structure. The implication is that the construction of structure is local and does not require any communication at all. In each level of partition, a cell (a least level partition) is assigned as clusterhead of the corresponding region. The clusterhead at each partition is statically assigned to be the closest cell to the geographic center of the entire region. This is to avoid backward links between parent and children. For example, in level 1 partition, cell 003 is selected as clusterhead for 00 region and cell 033 is selected as level 2 clusterhead. A cell may belong to different levels in the hierarchy depending on its location. If a cell is a clusterhead at level k, it is also a clusterhead at all levels less than k.

DQT is originally designed for static sensor networks, in this paper we embed DQT over MANet by using the static cell abstraction mentioned in the Introduction. For the rest of the paper, we investigate the new problems that arise due to node mobility in tracking applications.

While adapting DQT to mobile networks, we select the cell size to be less than half of the node communication range (as in GAF [16]) in order to ensure that a node in one cell can reach any node in neighboring cells. This selection enables us to use ad hoc routing among cells, which is more suitable for MANets. (The ad hoc routing among cells is discussed later in Section III.)

B. Location updates

The target location update event is disseminated following the logical MDQT hierarchy. Once an advertisement message reaches an intermediate cell in the hierarchy, it is forwarded upwards to its parent until it reaches the root (Figure 2). Different forwarding mechanisms are possible. An immediate

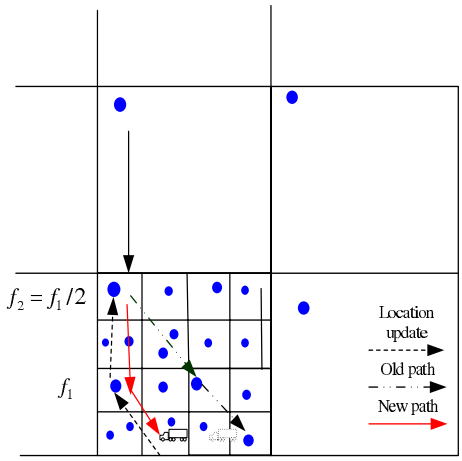

Fig. 2. Location update causes update of track path: higher level cells update the location information less efficiently.

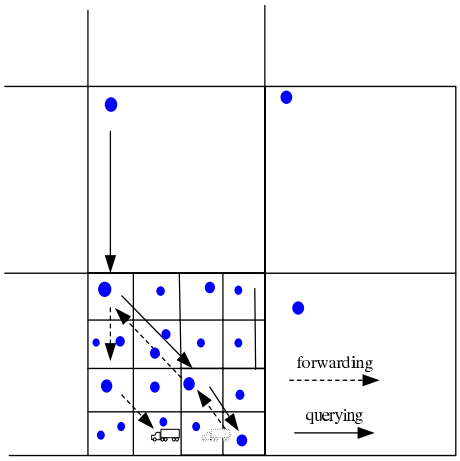

Fig. 3. MDQT uses query forwarding to track targets.

forwarding strategy forwards the message immediately after a cell receives a message. It is effective, but includes unnecessary overhead, because some updates may not be needed if the target moves back and forth in the same cell. Another method-Incremental forwarding strategy-forwards the message only if the update causes change of the tracking path upwards. If the target moves inside a cell, nodes inside that cell updates their location information as usual, e.g., timestamp of the record. However, high level clusterheads are unaware of these local changes. Only if the target passes through cell borders, this update will be forwarded upwards. This way the forwarding traffic are reduced because an advertising message stops at a certain level, however, this causes a problem in mobile networks: nodes at high levels may move out of their cells and new nodes moving in are unaware of the event due to the "shield" of low level cells, which in turn leads to failures in tracking.

To compromise the overheads in "immediate forwarding" and the deficiency in "incremental forwarding", MDQT uses the "intermittent forwarding" approach. Here intermediate clusterheads forward their events for each predefined period of time. In contrast to "immediate forwarding", in which clusterheads at various levels immediately forward their information to their immediate parents, here cells follow their own update frequency. Higher level clusterheads update less frequently while at lower levels, updates occur more frequently. This is helpful in balancing the load and reducing traffic collisions toward MDQT roots as well. We choose the updating frequency at every level i as:

$$f_i = f_1/2^{(i-1)}$$

The reason for this selection is that a clusterhead covers two times length of its children in each dimension, and as a consequence, it is approximately two times less sensitive to node mobility than its children at each dimension.

C. Tracking

Once a query is started from an initiator, it is forwarded to its immediate parent, and the process continues until finally the target's track is found. The query is then pushed downward following MDQT hierarchy until it reaches the target. More specifically, when a node receives a querying message, it performs the following:

- If the cell has no clue of target's location, it forwards the query to its parent cell. If the query has already reached the root, yet the root has no clue of the target, it returns a failure. A failure may be caused by the incorrect receiving of messages during forwarding, missing update of location advertisement, or highly dynamic node mobility.
- If the cell has an advertisement of the target, it points the query to the corresponding child from which the advertisement is last received.
- If the query reaches a bottom level of cell, but it finds out that the target is no longer there, it waits for certain period of time, then performs a query forwarding.

Query forwarding is an important part of the MDQT tracking process. After every forwarding, a query message is closer to the target than before, which in fact is the essence of our tracking algorithm. Although the query may fail to find the target at one round, the target is located somewhere not far away, and the forwarded query message is more likely to hit the target within a few hops. Figure 3 illustrates how query forwarding can keep tracking the target effectively.

Tracking results can be replied back to the querying node through reversed searching. This operation is symmetric to the searching of target, hence we do not include it in our discussion and implementation.

D. Loose synchrony

To maintain the data consistency in a cell, the previous mobile tracking frameworks VINESTALK [13], GLS [12] and HGRID [14], used strict synchronization, where when a new node enters a cell requests for the state information of this cell. This causes extra overhead and consumes unnessary network

bandwidth when all nodes are mobile. We observe that such strict synchronization for the nodes is not necessary as it is sufficient for one node in a cell to respond to a query, while other nodes catch up opportunistically. Considering the constant changing of topology in our model, the loose synchrony approach is most appropriate because it avoids extra traffic in high mobility circumstances where a lot of nodes move in/out cells.

In MDQT, we loosely synchronize the state of nodes in each cell through snooping the location update event by utilizing the nature of wireless broadcast communication. More specifically, 1) when a new node enters a cell and detects an event/query forwarded/replied by neighboring nodes in the same cell, this information is inserted as if the event has been advertised, 2) when a record reaches its lifetime, it is deleted, 3) when a node leaves a cell, the data associated is dropped.

Of course, such lightweight implementation trade-offs the possibility of failures. Transient data inconsistencies occur when all the nodes holding the data move out of a cell, while those newly joined nodes do not have any clue of the data. In this case, the nodes simply forward the query to the parent cell for a resolution. If this happens at the root, it becomes a failure. Other reasons for failure are: a query is forwarded to an empty cell; a query message is lost due to collisions; or the number of query forwarding exceeds the maximum limit.

E. Overhead analysis

We analyze the overhead and efficiency in this section. Our model considers a l leveled MDQT field that consists of n nodes randomly distributed.

Indexing. The expected location indexing message overhead is $O(sqrt(n))$. This is because each advertisement needs to travel $O(2^{l/2})$ hops, and $l = log(n)$.

Querying. Assume that the target is d distance away from the tracking node and moves at speed v, the cost for a query to reach the target given the assumption that message routing follows shortest path in message delivery is:

$$s(d + vt)$$

where t is the delay from the query message being initiated to reach the target, and s is the distance stretch factor. Stretch factor measures distance-sensitivity and implies that the cost of answering a query for an event should be at most a constant factor "s" of the distance "d" to the event in the network. We cannot guarantee distance-sensitivity for each query due to the hierarchical boundary issue, however we achieve distance-sensitivity on average. Our simulation results confirm this claim. Later in the discussion section, we introduce an efficient method to handle this multi-level boundary problem.

Space overhead. Since each node can be at any level in the hierarchy, the space required for each node is $O(\log m)$, where m is the number of targets in the field.

III. Implementation

A. Routing

Here we present the MDQT routing protocol. Many work have been done in routing from different perspectives including energy aware routing [8], [18] and geographic routing [10], [11]. These work need to maintain clusters or neighbor list information, however node mobility results in continuous topology changes and maintaining neighbors table information is very costly for high mobility environments. Instead MDQT routes messages in an ad hoc fashion using "polite gossip". In this approach, all the nodes in the cell may intend to act as a relay and forward a received message towards its destination. However, nodes select a random backoff time, and are allowed to relay the message only after this backoff time. If during the backoff period, a node hears any other node in the cell forwarding the same message, the node suppresses repeating this message and removes the message from its sending queue. This way nodes in the same cell coordinate implicitly, and as a result, duplicate message sends are reduced, if not completely avoided. Below, we show that with careful design of application layer backoff timers, we can reduce such collisions to an acceptable level.

As mentioned before, MDQT querying operation consists of two phases: an "upward" phase and a "downward" phase. At the "upward" phase, messages are forwarded to higher level cells until they find the track of the target; at the "downward" phase, messages are passed down to the leaf cells until the real target is reached. MDQT Routing is performed as follows: If the forwarding destination (e.g., to its parent) is a neighboring cell, the node sets the next hop exactly as the destination; If the forwarding destination is not within one hop distance, the cell calculates next hop locally by taking advantage of MDQT_ID properties. The next relay hop is selected as the closest neighboring cell to the forwarding destination. Therefore messages follow a cell level shortest path. The resulting routing path is similar to geographic routing in a sense that they both follow a relatively short path. Nevertheless, MDQT routing is more lightweight and superior to geographic routing in mobile networks as it does not require maintaining a neighbor list during each move, which is very costly or even impossible for some high mobility environments.

B. Backoff timer-Application layer

There are two types of backoff timers in our model- a default CSMA backoff timer and an application layer backoff timer. These two timers perform different functions in our model: the default CSMA backoff timer is for reducing collisions and application layer backoff timer is used for suppressing the sending of duplicated messages. The application layer backoff timer should be carefully designed to reduce message duplication and conflicts in routing. Let T_{app} denote the application layer maximum backoff time. If there are n nodes within the same cell, the joint probability of backoff time $p(t_1, t_2, ..., t_n)$ can be written in conditional probability:

$$p(t_1, t_2, ..., t_n) = p(t_1/t_2, t_3, ...t_n) * p(t_2, t_3, ..., t_n)$$

The probability that the rest $n - 1$ nodes can hear the first node's transmission and disable their own transmission

attempts is:

$$P = \int_0^{T_{app}-T_{msg}} \int_{t_1+T_{msg}}^{T_{app}} ... \int p(t_1, t_2, ..., t_n)..., dt_2, dt_1$$

The result of this integration is:

$$P = (1 - \frac{T_{msg}}{T_{app}})^n$$

Using this formula, we can set T_{app} appropriately according to system requirements. For example, assuming there are two nodes sharing same MDQT_ID on average and the duplication of sending is less than 10%, the application layer backoff is:

$$T_{app} \approx 20 * T_{msg}$$

C. Setting of soft-state parameters

One advantage of our framework is the tunability of the parameters to achieve better performance in querying cost, energy and fault rate. The cost C includes querying cost and indexing cost. We analyze parameter constraints and show how to reduce the cost by adjusting advertising frequency. We use the following notations in this discussion:

- The source node queries the target at each interval t_q and the target node advertises at each t_a period.
- The average distance(in hops) between the querying node and target is d_q and the average distance for event advertisement is d_a.
- The stretch factor s used to measure querying/advertisement distance sensitivity implies that the cost of the querying/advertising task is resolved by at most constant factor s of the distance d to the target interested. s_q is the stretch factor for querying tasks and s_a for advertisement tasks. In MDQT, a query is forwarded at the missing point in order to catch up with the moving target eventually. So s_f is used to represent the stretch factor of such forwarding effort and t_f is the forwarding waiting time.

Suppose with the target moving speed v_t, p_f percent of querying requests need forwarding. We can obtain following observations:

- The overall stretch factor can be approximately expressed as:
$$s = s_q \cdot (1 - p_f) + p_f \cdot s_f \cdot \frac{d_f}{d_q}$$
- The forwarding waiting time t_f should be carefully chosen. If t_f is too short, the forwarding message will keep bouncing in the false positive loop, causing large overhead and wasting energy; while t_f is too long, it not only introduces more latency but also more likely to miss in next round. Therefore:
$$t_a \leq t_f \leq t_l$$
Although setting of t_f and t_f largely reduces traps, we avoid traps by performing loop detection/cancellation on receiving each querying message as well. Each query message has a unique MessageID, which is stored in event table corresponding to the particular EventID. It is flushed once the event information is updated through advertising. A query message avoids looping back to its child node on witnessing the same querying MessageID as itself.

TABLE I

PARAMETERS FOR SIMULATION

Topology update frequency	15/s
Data lifetime(t_l)	10s
Advertisement frequency(t_a)	1/s
Query frequency(t_q)	0.22/s
Forwarding delay(t_f)	1s
Application layer backoff(T_{app})	0.3s(max)
Mobility waiting time	0-1.5s
Average density	4/cell
Cell size	100m
Transmission range	250m

In the following we show how to tune the advertising frequency to minimize the overhead. The cost C of querying/advertisement within period T is:

$$C = \frac{T}{t_q} \cdot d_q \cdot s_q \cdot (1 - p_f) + \frac{T}{t_a} \cdot d_a \cdot s_a + \frac{T}{t_q} \cdot p_f \cdot t_a \cdot v_t \cdot s_f$$

In order to optimize advertisement timer, we assume that the new interval becomes $\lambda \cdot t_a$. With the same assumption, the cost becomes:

$$\hat{C} = \frac{T}{t_q} \cdot d_q \cdot s_q \cdot (1 - p_f) + \frac{T}{\lambda \cdot t_a} \cdot d_a \cdot s_a + \frac{T}{t_q} \cdot p_f \cdot \lambda \cdot t_a \cdot v_t \cdot s_f$$

The goal is to choose an appropriate λ to minimize the target function:

$$\lambda = argmin(\hat{C} - C)$$

The result of λ is achieved by following:

$$\lambda = \frac{1}{2} + \frac{t_q \cdot d_a \cdot s_a}{2 \cdot p_f \cdot t_a \cdot v_t \cdot s_f}$$

IV. SIMULATION

We evaluate MDQT performance and mobility behavior in Prowler. Prowler is a discrete time, event-driven wireless sensor network simulator. We enhanced Prowler such that it can support node mobility. In our enhanced version, nodes' positions are refreshed at each topology update interval.

The network consists of 256 nodes, uniformly placed with 100 meters spacing initially. We implement a 4-level-MDQT structure for tracking where MDQT_ID is achievable in real time for each node. The transmission power is set to cover neighboring cells, thus all nodes in neighboring cell are reachable via single hop. The target node is placed at the corner initially and the tracking node is placed randomly at the beginning of each experiment. We investigate the impact of different mobility scenarios using the following metrics:

- Success Rate. Success rate is used to measure the reliability of MDQT tracking scheme with node mobility and possible message collisions. A tracking is considered successful if the querying message finally reaches the

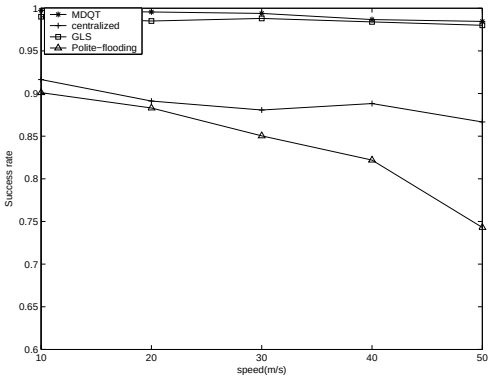

Fig. 4. Success rate under different mobile node speed

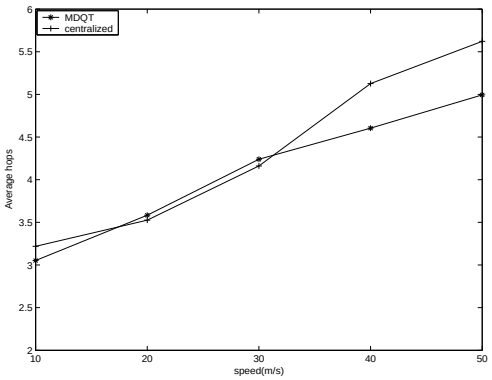

Fig. 5. Average hops under different mobile node speed

target. Here we only allow at most one query forwarding in our simulation.

- Average hops. Average hops measure the expected communication cost for the querying of target. In order to measure the cost, each query message contains a segment indicating how many hops it has experienced.
- Latency. Latency is an important performance measurement especially for time-critical applications. It is measured from the starting of a message until it reaches the target. Failure cases are not counted in calculation.
- Forwarding rate. Forwarding rate measures the fraction of querying that are resolved through in-network forwarding operations. Of course, low forwarding rate is desired as each forwarding causes extra delay and communication cost.

Every experiment is repeated 10 rounds with 500 seconds in simulation time for each round, and metrics are calculated in average. The settings of parameters are listed in Table I. Node movement follows random waypoint mobility model: a node randomly selects a point in the field and moves toward that position with a certain speed; when it arrives, the node waits for certain period and selects next waypoint. The waiting time in our simulation is set to be randomly in range [0-1.5] seconds.

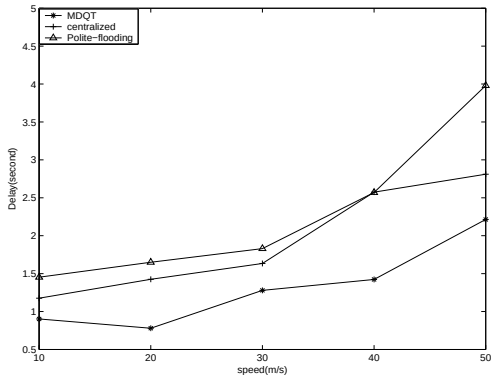

Fig. 6. Average delay under different mobile node speed

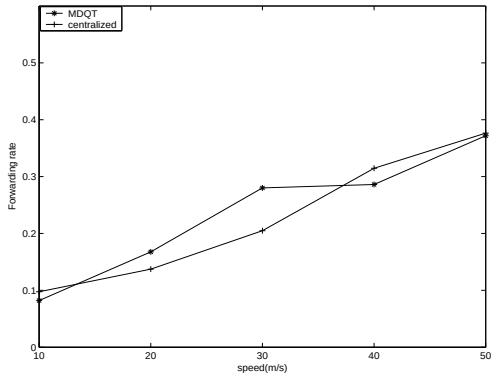

Fig. 7. Forwarding rate under different mobile node speed

A. Target mobility only

In this experiment, all nodes are static except the target. The target is located in the northeast corner at the beginning of the simulation and follows random way point mobility model. The querying node is placed somewhere around the center of the simulation area. We compare MDQT with centralized tracking solution, where both location updates and querying are directly sent to a centralized server (we set a root cell as centralized server), and polite-flooding scheme over logical cell layer. In the polite-flooding scheme, the location event is flooded to every cell, and a query is forwarded to the target directly.

Figure 4 shows the success rate for different schemes with different target mobile speed. As comparable to GLS, MDQT maintains more than 98% percent of tracking success rate despite of the target movement speed. With aggressive updating, GLS and MDQT performs similarly for static networks. Nevertheless, the centralized approach is greatly affected by target mobility speed. This is mainly due to the increasing of hops and longer delay it travels before a query forwarding reaches the server, which leads to more failures for higher speed targets. Polite-flooding based tracking scheme behaves even worse due to the longer delay and more collisions (Fig.6).

The results in Figure 5,6,and 7 can be explained as follows: the average number of hops for tracking a mobile target increases approximately linearly with the increasing of target speed (Fig.5); this is mainly because of the linear increment of

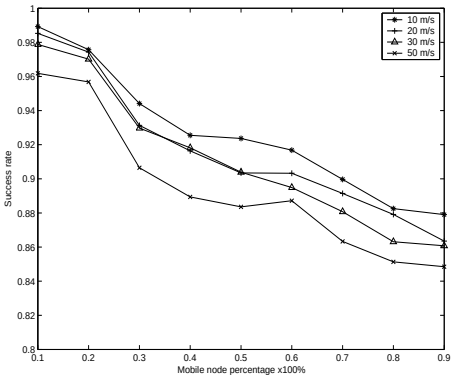

Fig. 8. Success rate vs mobile node percentage

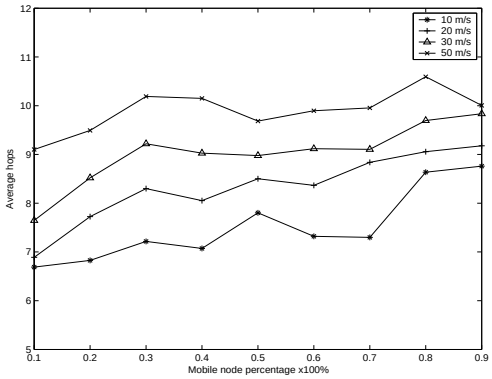

Fig. 10. Average hops vs mobile node percentage

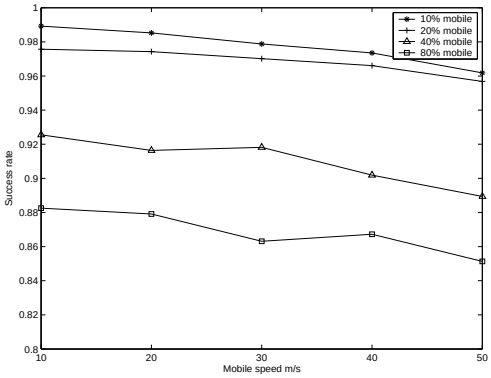

Fig. 9. Success rate vs mobile speed

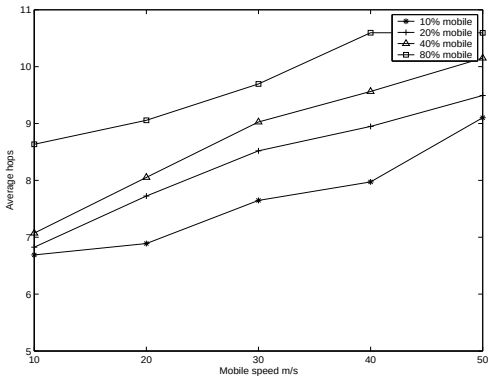

Fig. 11. Average hops vs mobile speed

query forwarding that the system spends to catch up with the target (Fig.7), which in turn causes the increase of average delay (Fig.6). The increasing of forwarding rate and delay is approximately linear with respect to mobility speed. This is because the message is more likely to miss the target with higher mobile speed, and the miss of target results in more querying forwardings, hence the increasing of delay and average hops.

The reason we only see slight differences between MDQT and centralized tracking in Figure 5 and 6 is that those metrics are calculated only on those successful queries. With the increasing number of levels in cell partition, we can expect larger gap in those performance metrics.

B. Fully mobile networks

An important part of our experiment is to evaluate the MDQT performance for fully mobile networks, where every node is able to move. We use the same initialization topology as section A but vary the percent of mobile nodes. Again, to ease comparison, we measure the same metrics under random waypoint mobility model for the querying node in every experiment. We evaluate the performance using a spectrum analysis from low mobile speed to high mobile speed and scale from static networks to fully mobile networks.

We can see from Figure 8 that failure increases when more nodes become mobile. The success rate is much lower than static networks mainly due to: 1) mobility increases the probability of node isolation and message loss is more likely to happen when the querying node moves to edges of the grid (due to the fact that the network tends to be sparse at edges with random waypoint model); 2) mobility increases the probability of data inconsistency in a cell. Here the inconsistency refers to the cell emptiness or that although nodes exist, they hold incorrect information. Figure 9 shows the correlation between success rate and mobile speed: high speed leads to low success rate under the same network configuration. However, after comparing Figure 8 and Figure 9, we observe that the success rate is more sensitive to the percentage of mobile nodes than the mobile speed, because query forwarding partially improved success rate. Nevertheless, even under a very highly dynamic circumstance (high mobile percentage, high speed), the tracking success rate remains above 85%.

Figure 10 measures the cost (in average hops) of tracking under mobile node percentage. The average hops increase very slowly (almost negligible) for each particular mobile speed with changing of mobile node percentage. This is mainly due to the soft-state nature of MDQT structure: mobility does not change the cell logical hierarchy. On the other hand, the cost is proportional to the mobile speed (Figure 11): high speed increases the probability of query forwarding, hence the average hops. This also validates our analysis that due to the static cell layer, the cost is proportional to mobility speed, but

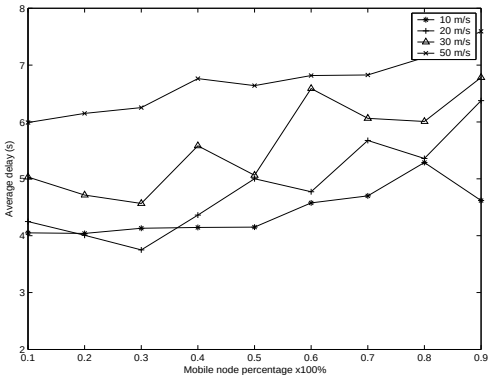

Fig. 12. Average delay vs mobile node percentage

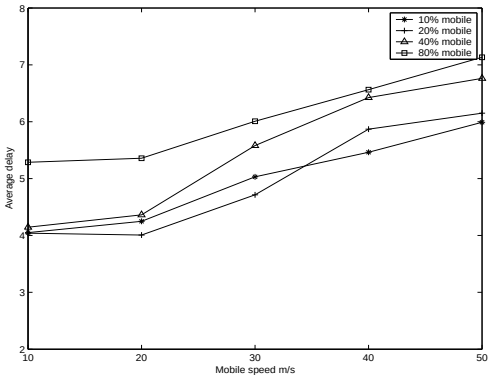

Fig. 13. Average delay vs mobile speed

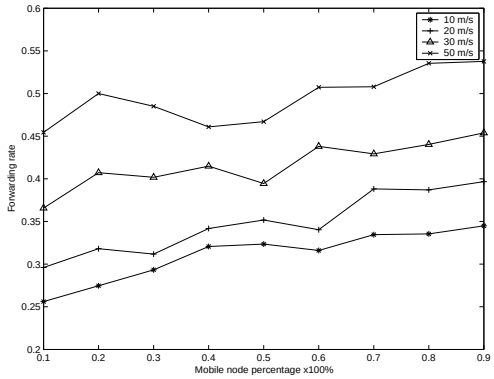

Fig. 14. Forwarding rate vs mobile node percentage

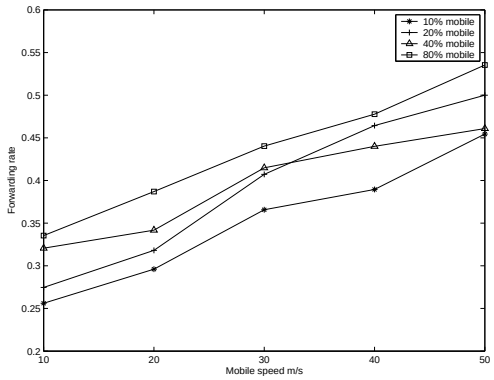

Fig. 15. Forwarding rate vs mobile speed

insensitive to the amount of mobile nodes.

Figure 14 and Figure 15 illustrate the forwarding rate under different configurations. The forwarding rate increases nearly linearly with the increase of mobile speed, however, it is less affected by increasing the percentage of mobile nodes. That is the reason we see some intersections in Figure 15 where the mobile node percentage differs slightly. Meanwhile, the increase of the forwarding rate is the cause of the increase of the cost as well as tracking delay.

Tracking delay (Fig.12) also demonstrates great resiliency with respect to mobile node percentage: it is less affected by increasing the percentage of mobile nodes than mobile speed. For example, with speed 10m/s, the increase of mobile node percentage from 10% to 90% only adds to several hundreds of milliseconds. The large variance of delay with same speed is caused by the randomness of backoff timers and relatively small number of repeated experiments. Again, the average delay increases nearly linearly with the rise of mobile speed (Fig.13). To sum up, MDQT is sensitive to mobile speed: high speed results in more chances of query forwarding and thus higher cost and longer delay, and MDQT is less sensitive to mobile node percentage due to cell abstraction and soft-state nature.

C. Multiple Targets

All the experiments till this point only consider one single target. The extension to multi-target tracking is straightforward: MDQT treats every target as a distinct type of event, which is stored in the cell hierarchy. Hence tracking of each target uses the same strategy. To show scalability of MDQT in multi-targets applications, we present simulations with multiple targets in Figure 16. We set 40% of nodes to be mobile and the mobility speed is about 25m/s.

As shown in Figure 16 the success rate is impacted slightly, while the delay, average hops and forwarding rate remain the same for multiple targets. Note that we use Prowler-a WSN simulator, which has extremely limited bandwidth and low bit rate radio. The increasing of bandwidth will greatly reduce the transmission delay and increase success rate due to: 1) the time for transmitting messages is reduced; 2) the application layer backoff timer can be lowered. Our simulations show that MDQT is scalable to node mobility for tracking applications.

V. Discussion

Does mobility always make things worse? MDQT addresses mobility by introducing the static cell abstraction and masks the impact of node mobility within a MDQT cell. Inter-cell mobility is handled by query forwarding techniques at the failure position. While the sensor node mobility causes

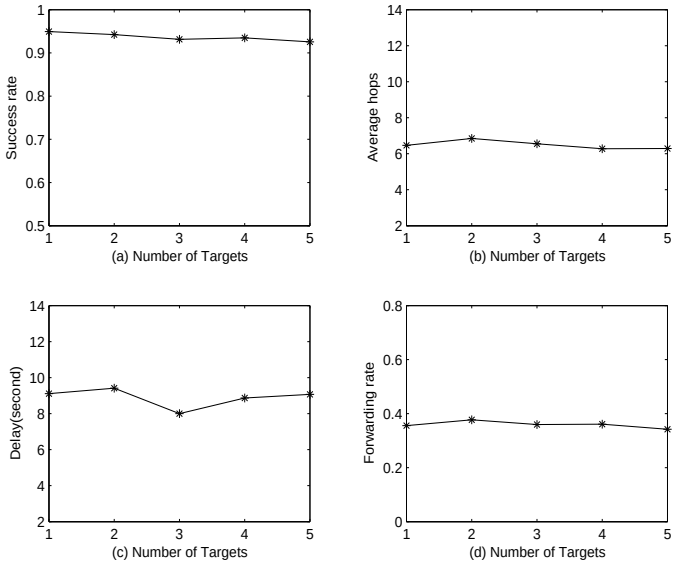

Fig. 16. Tracking performance with multiple targets

overhead in maintaining the soft-state, sensor node mobility also brings following benefits:

- **Load balancing**. MDQT is a hierarchical mobility management framework, high level clusterheads handle more traffic than lower level cells. This causes load balancing issue for static networks and this happens to be alleviated by node mobility in MDQT.
- **Information dissemination**. When a node moves from one cell to another, it may carry some information from the old cell. Such information can be used for tracking. For example, if a querying does not find the target at a certain level, nevertheless it may learn from a node newly moved from a neighboring cell that the target is located somewhere a short time ago. To utilize this information we may consider forwarding the query directly to that place, reducing the communication cost by preventing propagation of messages to high level parents.

Reducing tracking failures. The tracking success rate in MDQT is related to factors such as mobility speed, percentage of mobile sensor nodes, traffic load, as well as MDQT specific parameter settings for instance location publishing frequency. Those settings should satisfy the constraints we mentioned in section III to achieve better performance. We can apply immediate forwarding strategy in location update to improve success rate by spending more energy and space. The selection of cell size also influences the tracking performance–proper setting of cell size depends on the node density and transmission range. We set MDQT cell size so that every neighboring cell can be reached via one hop similar to GAF [16].

Handling hierarchical boundary issues. Even though MDQT achieves distance sensitive latency in average, the worst case scenario suffers from hierarchical boundaries. A small move of the target may lead to many extra hops in tracking and introduce very long delay. This can be improved by applying a boundary-aware solution. To keep the boundary solution lightweight, we only need to focus on the most problematic cases–deep boundaries: for instance, only if the boundary is level two or above, we query the neighbors across the boundary before starting the query forwarding process.

Handling node failures. Based on the scales of failure, node failure can be classified as single node failure and failure of one or more cells. Single node failure is neglectable in MDQT, while regional failure may cause routing and storage failures. Emptiness of a cell due to node mobility has similar impact in the system as regional failure. Some techniques, for instance, cell by-passing, or right-hand-rules as in geographic routing, might be helpful. The study of MDQT performance against node failure is left for future work.

Optimizations. So far we have not tuned the parameters for optimization. In our experiments, all parameters are set statically in Table I, and it is easy to verify that the setting conforms to those constraints mentioned in section II and III. The settings can be tuned according to different optimization purposes: for example, if the purpose is to improve the tracking success rate, higher advertisement frequency is desired; if the queries become less frequent, we can reduce advertising frequency and increase data lifetime to reduce communication overhead.

VI. Related Work

Hierarchical approaches have received considerable attention in MANets [2], [5], [13], [16]. Most of these work are focusing on data processing and routing in static or low mobility networks. VINESTALK [13] is a recent algorithm for tracking objects in MANets. Each node is associated with a Virtual Stationary Automata(VSA) [7], where detected events are stored. Tracking path is thus maintained by VSAs instead of real nodes. The cost of finding a mobile object distance d away take $O(d)$ and updates to the tracking structure of moving d distance take $O(d * log$ network diameter). MDQT shares similar idea of "virtual nodes"(which we abstract as a cell) but is lightweight both due to the local construction of cells and due to the soft-state information maintenance. In contrast to VINESTALK where nodes simulating a virtual node have to perform consensus over every input/output of the virtual node, we observe that loose eventual synchrony can be sufficient for nodes in a cell for the tracking application. We observe that it is sufficient for one node in a cell to respond while other nodes catch up opportunistically. By adopting this loose eventual synchrony approach, we get away without the strict replication, synchronization, and consistency requirements in VSA [7]. Since we avoid explicit update message exchange operations, our soft-state cell abstraction is lightweight and simple to implement.

Another closely related work to MDQT is Grid Location Service(GLS) [12]. GLS is a scalable and distributed location service structure which divides the global map into hierarchical grids with increased size at high levels. It selects location servers at each level based on *least greater ID* rules. Queries are forwarded in the same rule by checking location tables.

However GLS results in extra overhead because: 1) GLS periodically broadcasts HELLO messages to maintain the table of neighbor information such as IDs, locations. 2) Under high mobility circumstances, each virtual cell keeps "forwarding pointers" of the nodes moved out of the cell. This not only creates extra traffic to the system but also degrades the performance of the system. The success rate in GLS when all the nodes are mobile drops to 60%. In contrast, MDQT pays no cost to such maintenance operations and the the success rate remains higher than 85% even if all nodes are mobile.

Similarly, Hierarchical Grid Location Management (HGRID) [14] provides a location service for mobile ad hoc networks. Although HGRID is a grid based hierarchical structure like MDQT, MDQT differs in many aspects from HGRID. HGRID uses hard-state approach and handoff-based location update scheme - when a node moves to a new grid, a location handoff happens; HGRID focuses on routing purposes: all mobile nodes need to update their location servers periodically. In contrast, MDQT uses soft-state and loose synchrony to maintain targets' location, requires less updates and bandwidth yet achieves better success rate in high speed fully mobile networks; MDQT studies both the impact of mobile nodes and mobile speed using a spectrum analysis.

TTDD [17] (Two-Tier Data Dissemination) aims to address sink mobility. It proactively constructs a grid structure, so that only the sensors on the grid point need to acquire the forwarding information. The mobile sink needs to reconstruct its entire path whenever it moves to a new grid. Flooding is used in the local cell containing current sink's location. Compared with MDQT, TTDD is limited to static intermediate nodes and the success rate is below 90%, while MDQT allows intermediate node to mobile and achieves higher success rate.

Y. Zhou et. al proposed a distributed mobility management for target tracking in mobile sensor networks in [19]. Their mobility management scheme considers node movement decisions as part of a distributed optimization problem to improve the quality of target tracking under constraints of energy consumption and link quality. Based on those metrics, a node can move to another location purposely after selecting the "best" candidate location. Our framework considers more generalized cases - uncontrolled mobility e.g., random waypoint mobility, because intentional movement is often unrealistic.

VII. Conclusions

In this paper, we have proposed the MDQT tracking framework for large scale mobile ad hoc networks. Here a "cell" abstraction is used to dynamically map mobile nodes that fall within a geographic area to model a *virtually static node* for that area and implement the virtual static node layer in a lightweight and communication-efficient manner using soft-state principle.

Our experiments on the impact of mobility speed as well as percentage of mobile nodes in terms of success rate, communication hops, latency and forwarding rate show that MDQT is an efficient and scalable in-network tracking framework for MANets. In our future work, we will study the impact of node density and node failures on MDQT performance using a spectrum of dense to sparse MANets.

References

[1] I. Abraham, D. Dolev, and D. Malkhi. Lls: a locality aware location service for mobile ad hoc networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 75–84, 2004.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, pages 102–114, 2002.

[3] H. Cao, E. Ertin, V. Kulathumani, M. Sridharan, and A. Arora. Differential games in large scale sensor actuator networks. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 77–84, 2006.

[4] M. Demirbas, A. Arora, and M. Gouda. Pursuer-evader tracking in sensor networks. *To appear in Sensor Network Operations, IEEE Press*, 2006.

[5] M. Demirbas, A. Arora, T. Nolte, and N. Lynch. A hierarchy-based fault-local stabilizing algorithm for tracking in sensor networks. *8th International Conference on Principles of Distributed Systems (OPODIS)*, pages 299–315, 2004.

[6] M. Demirbas and X.Lu. Distributed quad-tree for spatial querying in wireless sensor networks. In *Proc. IEEE International Conference on Communication (ICC)*, June,2007.

[7] S. Dolev, S. Gilbert, L. Lahiani, N. Lynch, and T. Nolte. Timed virtual stationary automata for mobile networks. *9th International Conference on Principles of Distributed Systems (OPODIS)*, 2005.

[8] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. Application specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Networking*, 2002.

[9] P. Ji, Z. Ge, J. Kurose, and D.Towsley. A comparison of hard-state and soft-state signaling protocols. *IEEE/ACM Transactions on Networking*, 15(2):281–294, 2007.

[10] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.

[11] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, May 2005.

[12] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 120–130, 2000.

[13] T. Nolte and N. Lynch. A virtual-node based tracking algorithm for mobile networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems(ICDCS)*, page 1, 2007.

[14] S. Philip. *Scalable Location Management for Geographic Routing in Mobile Ad hoc Networks*. PhD thesis, University at Buffalo, SUNY, 2005.

[15] S. Raman and S. McCanne. A model, analysis, and protocol framework for soft state-based communication. In *SIGCOMM*, pages 15–25, 1999.

[16] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin. Topology control protocols to conserve energy in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2003.

[17] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 148–159, 2002.

[18] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. *Tech Report UCLA/CSD-TR-01-0023*, May 2001.

[19] Y.Zhou and K.Chakrabarty. Distributed mobility management for targte tracking in mobile sensor networks. *Proc. IEEE Trans. Mobile Computing*, 8(8):872–887, 2007.

[20] W. Zhang and G. Cao. Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communication*, 3(5):1689–1701, 2004.