

# Dynamic Wake-up and Topology Maintenance Protocols with Spatiotemporal Guarantees

Sangeeta Bhattacharya, Guoliang Xing, Chenyang Lu, Gruia-Catalin Roman, Octav Chipara and Brandon Harris

Department of Computer Science and Engineering

Washington University in St. Louis

{sangbhat, xing, lu, roman, ochipara, bbh2}@cse.wustl.edu

**Abstract**—Many mission-critical applications require spatiotemporal data services for mobile users or objects. Examples include distributed object tracking and fire monitoring by firefighters. To support such applications, wireless sensor networks must satisfy a set of stringent spatiotemporal constraints despite having low network duty cycles and scarce resources. We have developed two new wake-up and topology maintenance protocols, Directional Tree Maintenance (DTM) and Omni-directional Tree Creation (OTC), to support spatiotemporal services in mobile environments. A key feature of our protocols is that they provide robust spatiotemporal performance while maintaining low overhead and energy consumption. Our simulations showed that both DTM and OTC can successfully deliver over 85% of query results to a mobile user within desired spatiotemporal constraints, even when the sleep schedule is as long as 15s, the user changes direction every minute, and the location error is as high as 10m. The benefits of our protocols have been validated through theoretical analysis and empirical results on a testbed of Mica2 motes.

## I. INTRODUCTION

Wireless sensor networks (WSNs) are envisioned as an enabling technology for a broad class of *mission-critical* applications in mobile environments. Such applications often involve mobile entities moving through a WSN that provides continuous data services to the mobile entities. The mission-criticality of these services imposes fundamental *spatiotemporal constraints* on the WSNs, which require the sensor data to be delivered at the right time and also at the right location. As motivating examples, we now describe two important applications that impose stringent spatiotemporal constraints.

*Spatiotemporal query:* Users of many mission-critical applications need to continuously gather real-time information from their vicinities as they move through the environment. For example, a fireman fighting a wildfire may require a periodic update of a temperature map of the area within one mile of his current location, to remain alert to the surrounding fire conditions. Query results are subject to a spatial constraint, that, all and only the sensors around the current position of the moving fireman must contribute to the query. At the same time, fresh sensor data must be collected and aggregated before each query deadline, which constitutes the temporal constraint. The fireman may be endangered by a quickly evolving wild fire if any one constraint is violated, *e.g.*, the query results are delivered too late or are aggregated from sensors at the wrong locations.

*Object tracking:* Many WSNs are deployed in unmanned environments to detect and track moving objects such as intruders. When a moving object is present, the sensors in its vicinity form a group and track the object collectively. The tracking results from sensors in the current group are delivered to the group leader that aggregates the data and then transmits it to a base station. The network may lose track of a moving object if the sensors around the current object position do not respond or if their sensor data is not delivered to the group leader on time.

As shown in the above examples, a common requirement of many mission-critical applications is the need to meet the stringent

spatiotemporal constraints associated with mobile entities. Meeting these constraints is especially challenging in WSNs due to their severe power and resource limitations. First, sensor networks usually need to operate under very low duty cycles in order to maintain a long lifetime. For instance, for a Mica2 mote to remain operational for 450 days, the duty cycle needs to be lower than 1% [1]. This corresponds to an active duration of 150 milliseconds in every 15 seconds, resulting in a wake-up delay as high as 14.85 seconds. Empirical results on a surveillance system based on Mica2 motes reported in [2] show that long wake-up delays can result in poor tracking performance. Therefore, we need wake-up protocols to forewarn sleeping nodes before the mobile entity reaches their vicinity. Second, the topology of the woken sensors must be dynamically configured and maintained in a timely fashion to adapt to entity mobility and to facilitate data aggregation and delivery. For instance, to support a spatiotemporal query, a routing tree needs to be maintained in order to continuously aggregate and deliver query results to the user as it moves. Similarly, for object tracking, group members must send their sensing results to the current group leader as the group moves with the object. Finally, the storage cost and network contention caused by frequent topology reconfigurations, induced by entity mobility, must be minimized, since sensors typically have very limited memory and bandwidth.

In this paper, we present two novel protocols, DTM and OTC, for wake-up and topology maintenance in mission-critical applications over mobile environments. Our protocols enable existing object tracking and query services to achieve desired spatiotemporal guarantees. The Directional Tree Maintenance (DTM) protocol minimizes the communication overhead, while the Omni-directional Tree Creation (OTC) protocol is robust against unpredictable movement patterns. To evaluate these protocols, we provide both analysis and extensive simulation results under a wide range of realistic settings. We also present preliminary results obtained from the implementation of our protocols on a physical testbed composed of Mica2 motes. Our simulation results show that both DTM and OTC can successfully deliver over 85% of the query results obtained from the network to a user querying its surrounding area with 150m radius twice every second, even when the wake-up delay is as high as about 15s, the user changes its motion pattern about every 60s, and the location error is as large as 10m.

The rest of the paper is organized as follows. We present related work in section II and then present our protocols in section III. Simulation results are discussed in section IV, followed by a brief description of the implementation of our protocols on a Mica2 mote testbed, in section V. We finally conclude in section VI.

## II. RELATED WORK

Several general power efficient MAC protocols [3]–[7] exist that maintain node sleep schedule and wake up nodes dynamically. Unlike

these protocols, our protocols are specially designed to maintain desirable network topology for applications with spatiotemporal constraints, by waking up the right nodes at the right time. However, our protocols can be implemented on top of such MAC-layer solutions.

A number of promising signal processing algorithms and middleware have been developed for object tracking in WSNs. Some of the earlier solutions [8]–[10] do not employ wake-up protocols to handle sleeping nodes. As a result, they may suffer from poor tracking performance when nodes sleep most of the time and cannot contribute to the tracking process. For example, He et al. [2] observed that the sleeping schedule severely affects the tracking performance of their surveillance system when an object moves fast. Our work is complementary to these existing approaches in that our protocols can be used to wake up nodes and maintain desired network topology so that node duty cycles have no negative impact on the tracking performance.

Several recent projects have studied wake-up protocols for object tracking [11]–[13]. Gui et al. propose a wake-up mechanism to achieve specified quality of surveillance for moving objects [13]. A selective sensor activation scheme is proposed in [12] to achieve tradeoff between energy and quality of tracking. DCTC [11] is an object tracking protocol that maintains a tree around the predicted route of a moving target. All the above protocols are *best-effort* solutions that are not designed to meet the spatiotemporal constraints associated with object tracking. As a result, they may not be able to maintain the desired tracking performance in the face of fast objects and/or network low duty cycles. The ability to meet stringent spatiotemporal constraints differentiate our protocols from earlier work. Furthermore, the performance of the above protocols was verified only in simulation environments.

A protocol that bears some similarity to our protocols is mobicast [14], a spatiotemporal multicast protocol that disseminates data to a changing area just in time. However, Mobicast does not deal with sensor sleep schedule nor does it maintain a topology associated with entity mobility, which is needed for efficient data aggregation and fusion.

### III. PROTOCOL DESIGN AND ANALYSIS

In this section, we present the two new protocols, DTM and OTC, and show how they address the limitations of an earlier spatiotemporal query protocol called DTC .

#### A. Problem Formulation

We first present the protocol requirements and spatiotemporal constraints in the context of spatiotemporal query applications. As discussed in Section I, object tracking applications share similar constraints as spatiotemporal query applications.

A spatiotemporal query application involves a user who periodically queries the surrounding sensors (e.g., through a PDA) when he travels through a sensor field. A query is specified by the query period  $T_p$ , the data freshness  $T_f$ , and the query area  $A(p)$  defined relative to the user location  $p$ .  $T_p$  is the period at which the user expects to receive query results from the network.  $T_f$  specifies the maximum age of the query results, i.e., a result reported by a sensor is acceptable only if it is no more than  $T_f$  seconds old. The query area specifies which nodes should contribute to the query result. For simplicity, in the rest of the paper, we assume the query area to be a circle with radius  $R_q$  centered around the user. Our design can be easily extended to other types of query areas.

Protocols supporting spatiotemporal queries need to be able to (1) wake up all nodes in a query area so that it can deliver the query

result to the user before the end of the corresponding query period; (2) create and maintain a routing tree rooted at the mobile entity to support result delivery and in-network aggregation.

#### B. Design Considerations

This section outlines general issues related to wake-up protocol design. Wake-up protocols can be of two types: *greedy* and *just-in-time*. Greedy wake-up protocols wake up all required nodes much ahead of time by disseminating a wake-up message ahead of the user in an as-soon-as-possible fashion. However, this method requires the nodes to maintain state information for an extended period of time that results in high storage cost, which is not acceptable due to sensor memory limitations. This drawback promotes the use of the second approach to node wake-up, where nodes are alerted just-in-time for them to respond to the query at the right time. Just-in-time wake-up protocols employ a hold and forward strategy where the forwarding is timed, in order to reduce the storage cost. We incorporate the just-in-time approach in our protocols.

Some wake-up protocols [11], [15] rely on the motion profile of the mobile entity to predict its future locations. Two methods of generating a motion profile are motion planning and motion prediction. Autonomous robots usually generate motion profiles based on motion planning and control their future movement accordingly. In such cases, the motion planner can provide the motion profiles to the protocol. Alternatively, a motion profile can be generated based on the entity's movement history. As a simple example, a motion profile can be generated as follows. Assume  $p_1$  and  $p_2$  are two recent entity positions at time instances  $t_1$  and  $t_2$ , respectively. In object tracking, this information can be provided by the localization service of the tracking sensors. In a spatiotemporal query, the user can obtain this information through a GPS and provide it to the network. The future velocity of the user can then be estimated as  $\vec{v} = \frac{p_1 p_2}{t_1 - t_2}$ . This simple technique is used in our simulations. Other more complex techniques [16] can be used to improve the accuracy of the motion prediction. However, it may be difficult to obtain the motion profile using these two methods when the user motion is highly random or when the motion history has a high degree of inaccuracy (e.g., due to GPS location errors).

Our protocols run on top of a power management protocol, which maintains the connectivity of the network through a backbone of *active* nodes. The power management protocol maintains synchronous node duty cycles such that nodes periodically wakeup to receive messages and then go back to sleep. Messages to be sent to sleeping nodes are buffered by active nodes and are sent during the active window of the duty cycle. The backbone allows each sleeping node to be directly connected to an active node and hence bounds the communication delay between any two nodes to the order of one duty cycle. Several power management protocols [17]–[19] provide such a backbone. Our protocols are however not dependent on the backbone and can work with other protocols that do not provide such a backbone, with a slight modification in the timing analysis.

#### C. Directional Tree Creation

As a baseline for our new protocols, we first describe a wake-up protocol called *Directional Tree Creation (DTC)*, developed in our earlier work on MobiQuery, a spatiotemporal query middleware [15]. DTC uses the user motion profile to wake-up nodes along the predicted user path. The motion profile provides the prospective user positions called *pickup points* where the user expects the query results.

In DTC, after the user issues a query, the network sends a prefetch message to the first pickup point. The node receiving the prefetch message forwards it to the next pickup point. It then floods the query area and builds a tree spanning all nodes in the query area. A node in the query area sends its data to its parent at a scheduled time. The root of the query tree aggregates all results and delivers the final result to the user, when the user reaches the pickup point. The time instance when a prefetch message is sent is determined analytically to ensure that sleeping nodes in each query area are woken up in time to form a tree and send their results before the query deadline.

Although DTC significantly improves the spatiotemporal performance under certain conditions [15], it has the following limitations: (1) DTC creates a new tree spanning all nodes in a query area in each query period. As a result, it incurs a high communication overhead and hence fails to meet more stringent spatiotemporal constraints (e.g., higher query frequency). (2) DTC requires a motion profile that predicts the future user path, which may be inaccurate or unavailable in certain application scenarios. DTM and OTC, address these limitations of DTC.

#### D. Directional Tree Maintenance

DTM reduces the communication overhead by maintaining a *single* moving tree, that travels along with the user. It depends on neighborhood information as well as knowledge of the user motion profile to facilitate local decisions. Neighbor information is obtained from a lower power management protocol. Several power management [17]–[19] and geo-routing protocols [20] maintain neighborhood information and hence this information can be easily obtained without any additional overhead.

DTM comprises two main phases: *tree building* and *tree maintenance*. The tree building phase consists of creating an initial spanning tree in response to a broadcasted *Query* message from the user containing the following parameters:  $(M, R_q, T_p, T_f)$ , where  $M$  is the user motion profile consisting of the starting and ending locations and times,  $R_q$  is the radius of the query area,  $T_p$  is the query period, and  $T_f$  is the data freshness constraint.

Once the initial tree is built, it moves along the predicted user path. This is achieved by waking up nodes along the predicted path and growing the tree in the direction of user motion through node additions and deletions. This process defines the tree maintenance phase, and is explained below.

Nodes in the existing tree broadcast *Join* messages to inform nodes in future query areas about the query. On receiving a *Join* message, a node in a future query area joins the tree by first computing the earliest query period  $j$  that it participates in, and then selecting an active neighbor that is closest to the  $j^{\text{th}}$  pickup point as its parent. After joining the tree, the node performs the following actions: (1) it precomputes its parents for all query periods that it participates in, (2) it rebroadcasts the *Join* message at time  $t_{\text{join}}$ , (3) after sending the result for the current query period, it adjusts its parent for the next pickup point. A node always selects its parent to be the active neighbor that is closest to the concerned pickup point. If itself is the closest to the pickup point and is within the communication range of the pickup point, it sets the user as its parent. However, if it is not within the communication range of the pickup point, a network void exists; the approach to deal with this is discussed in section III-D.2. When a node is no longer in any future query area, it stops sending query results and hence automatically drops out of the tree. Nodes closest to the pickup point form the roots of the forest of trees, collect the data from their subtrees, and send the data to the user.

When *Join* message is received

- 1) Accept if participating in current or future query area.
- 2) For each query period  $k$  that this node is a part of
  - a) Set active neighbor, closest to the  $k^{\text{th}}$  pickup point, as parent.
  - b) If this node itself is the closest to the  $k^{\text{th}}$  pickup point, set user as parent.
- 3) Calculate  $t_{\text{join}}$  and set timer *JoinTimer* to fire at  $t_{\text{join}}$ .
- 4) Calculate time to send result and set timer *SendTimer* to fire at the right time.
- 5) If sleeping node, go back to sleep schedule.

When *JoinTimer* fires

- 1) If sleeping node, wake up.
- 2) Broadcast *Join* message.
- 3) If sleeping node, go back to sleep.

When *SendTimer* fires

- 1) If sleeping node, wake up.
- 2) Send query result to parent.
- 3) Adjust parent for next query period.
- 4) If sleeping node, go back to sleep.

Fig. 1. DTM Algorithm.

With this method of tree maintenance, at any given point of time, the tree spans multiple query areas. However, only nodes that are in the current query area participate in the data aggregation process. Sleeping nodes only form the leaves of the query tree so that they can maintain their sleep schedule and only wake up to broadcast the *Join* message and send the query result. Note that the tree maintenance phase is usually robust to *Join* message loss, since the same message is broadcasted by a number of nodes, which leads to redundant messages. *Join* message retransmission has been avoided in the protocol, to reduce overhead. In DTM, since each node maintains neighbor information and also knows the predicted user path, local computations at each node allow the tree to move in the right direction with no reconfiguration cost. The DTM algorithm is shown in Figure 1.

1) *Forwarding Time Analysis*: Future query area nodes are woken up by forwarding the *Join* message ahead of the user. The time to forward this message is critical to the working of DTM. If the nodes are woken up too late, they fail to participate in the query. While if they are woken up too early, they incur a high storage cost, as query related information must be maintained on the nodes for a long time [15]. Thus, in DTM, a node holds the received *Join* message until time  $t_{\text{join}}$  at which time it rebroadcasts the *Join* message to notify its neighbors to join the tree.  $t_{\text{join}}$  is calculated as follows.

Let  $k$  be the first query period that a node  $i$  participates in and  $t_{\text{recv}}$  be the latest time by which the nodes in the  $k^{\text{th}}$  query area should receive the *Join* message. A *Join* message should be sent out to the nodes in the  $k^{\text{th}}$  query area before time  $t_{\text{join}}$ , where  $t_{\text{join}} \leq t_{\text{recv}} - T_{\text{msg}} - T_s$ .  $T_s$  is the wake-up delay caused by the duty cycle of the sleeping nodes, and  $T_{\text{msg}}$  is the time taken by the *Join* message to reach the furthest node to be included in the query area. For the network communication to be able to catch up with the user movement,  $T_{\text{msg}}$  must be shorter than the time it takes the user to travel between two pickup points, which is equal to  $T_p$ . Hence, it is reasonable to assume that  $T_{\text{msg}} \leq T_p$ .

We now analyze  $t_{\text{recv}}$ . Once the nodes receive the *Join* message, they need to aggregate the partial query results and deliver the aggregated result to the user, when the user reaches the pickup point at time instance  $kT_p$ . Hence,  $t_{\text{recv}} = kT_p - T_{\text{collect}}$ , where  $T_{\text{collect}}$  is the time spent in data collection. Due to the freshness requirement,

<p>When <i>Query</i> message is received</p> <ol style="list-style-type: none"> <li>1) If there is a neighbor that is closer to the user, set neighbor as parent, else set user as parent.</li> <li>2) Calculate <math>n</math> using (6) and send <i>Setup</i> message.</li> <li>3) If sleeping node, go back to sleep after <math>nT_p</math> seconds.</li> </ol> <p>When <i>Setup</i> message is received</p> <ol style="list-style-type: none"> <li>1) Accept if in query area.</li> <li>2) Set node source as parent.</li> <li>3) Rebroadcast the <i>Setup</i> message.</li> <li>4) Set <i>SendTimer</i> to fire at data collection time.</li> </ol> <p>When <i>SendTimer</i> fires</p> <ol style="list-style-type: none"> <li>1) Send result to parent.</li> <li>2) If leaf, broadcast <i>Wakeup</i> message.</li> </ol> <p>When <i>Wakeup</i> message is received</p> <ol style="list-style-type: none"> <li>1) Accept if in circle <math>C</math>.</li> <li>2) Rebroadcast the message.</li> <li>3) If sleeping node, go back to sleep after <math>nT_p</math> seconds.</li> </ol>
--

Fig. 2. OTC Algorithm.

a query result has to be sent to the user within  $T_f$  seconds after it is gathered by a node. Hence  $T_{collect} < T_f$ . Thus, from the above discussion, we obtain the following inequality:

$$t_{join} \leq (k-1)T_p - T_f - T_s \quad (1)$$

Therefore, the network will meet its temporal constraints if each node broadcasts the Join message at a time  $t_{join}$  that satisfies (1). To minimize the storage cost, DTM sets  $t_{join} = (k-1)T_p - T_f - T_s$ .

2) *Handling Network Voids*: The above algorithm is designed considering a highly dense network and does not address the presence of voids. Network voids can however lead to cycles in the tree, thus reducing data fidelity. In order to handle such situations, we associate a property called *safe* with all nodes. A node is *unsafe* with respect to a pickup point if it is a *local minima* (that is, it is closest to the pickup point among its neighbors but the pickup point is not within its communication range) or does not have safe neighbors that are closer to the root than itself. A previously *safe* node broadcasts an *unsafe* message when it becomes *unsafe*. Nodes also include this information in the *Join* message, by specifying whether they are safe for each query period that they are a part of.

In this modified method, nodes choose only safe nodes as parents. If an *Unsafe* message is received from a parent, nodes adjust their parent to be a safe neighbor. If the receiving node becomes unsafe in the process, because it no longer has any safe neighbor closer to the pick up point, it sends out an *Unsafe* message, and the process is repeated. All neighbors are initially considered safe. This information is updated after receiving a *Join* message or an *Unsafe* message.

#### E. Omni-directional Tree Creation

Both DTM and DTC assumed the availability of an estimated motion profile. In situations where the user movement pattern is highly unpredictable, or the motion history information has high location error, it is not possible to wake up just the right nodes ahead of time. However, if the maximum user speed is known, we can wake up all the nodes in a circle  $C$ , henceforth called *wake-up area*, centered at the current user location, such that nodes in future query areas encircled by  $C$ , are ready to aggregate and provide the query result to the user, irrespective of the user speed and direction of motion. This is the approach taken in OTC. The size of the wake-up area is chosen such that it encloses nodes belonging to a certain number of query areas  $n$ . The constant  $n$ , called *wake-up horizon*, is dependent on the node duty cycle and is chosen such that nodes

in the  $n^{th}$  query area are woken up by the time the user reaches the query area.

Since the nodes are not aware of the user location ahead of time, the query result cannot be made available to the user exactly when it reaches a pickup point. Nodes learn about a pickup point only when the user reaches the pickup point, and deliver the query result to the user within a certain delay  $T_d$  of its passing the pickup point.  $T_d$  is set by the user and is included in the information sent to the nodes.

In this algorithm, since the user motion profile is not available, the user is required to broadcast a *Query* message at each pickup point. The *Query* message contains the parameters  $T_p$ ,  $T_d$ ,  $R_q$ ,  $v_{max}$ , and  $p_{user}$  where  $v_{max}$  is the maximum user speed and  $p_{user}$  is the user location. A query tree is built at each query point, in response to a *Query* message, by flooding a *Setup* message in the query area. The *Setup* message contains the same information as in the *Query* message, the pickup point number and the value of the wake-up horizon. Nodes in the tree then calculate the time after which the result needs to be sent to the parent. Once the result is sent, leaf nodes further broadcast a *Wakeup* message containing  $T_p$ ,  $n$ ,  $p_{user}$ , and  $R$ , where  $R$  is the radius of the circle  $C$ . The *Wakeup* message is flooded in the circle  $C$ , waking up all possible future participating nodes ahead of time. Once woken up, nodes with a sleep schedule go back to sleep only after  $nT_p$  seconds. The OTC algorithm is shown in Figure 2.

1) *Analysis of Wake-up Area*: The radius of the wake-up area,  $R$ , is chosen such that sleeping nodes in the  $n^{th}$  query area are woken up by the time the user reaches the query area. Thus

$$R = nv_{max}T_p + R_q \quad (2)$$

which is the distance of the furthest node in the  $n^{th}$  query period from the current user location.  $nv_{max}T_p$  in the above equation is the distance between the  $n^{th}$  pickup point and the current pickup point while  $R_q$  is the maximum distance of a node in the  $n^{th}$  query area from the corresponding pickup point.

Let  $t(d)$  represent the time taken by a message to travel distance  $d$ .  $t(d)$  can be calculated as a product of the maximum number of hops and the 1-hop-delay as follows

$$t(d) \leq \alpha \lceil \frac{d}{R_c} \rceil \tau \quad (3)$$

where  $\alpha$  is the *network dilation* and  $\tau$  is the 1-hop broadcast delay. Network dilation is defined as the upper bound on the ratio between the actual network distance (the number of hops) between any two nodes and the minimum network distance  $\lceil \frac{d}{R_c} \rceil$  between them, where  $d$  is the distance between two nodes. Network dilation is shown to be bounded in sensor networks with sensing coverage [20]. For networks without sensing coverage, the network dilation can be measured as shown in [14]. In our simulations, we use the network dilation bound as given in [20] and measure the 1-hop broadcast delay by timestamping messages and calculating the delay in message reception.

The time taken by the user to travel distance  $R$  is  $nT_p$ , while the time taken by a message to travel this distance is bounded by  $t(nT_p v_{max} + R_q) + T_s + \epsilon$  where  $\epsilon$  is the additional time taken by a node to send out a *Wakeup* message. Since a node sends out a message only after sending the query result,  $\epsilon \leq T_d$ . Since the message should reach the nodes before the user gets to the pickup point,

$$nT_p \geq t(nT_p v_{max} + R_q) + T_s + T_d \quad (4)$$

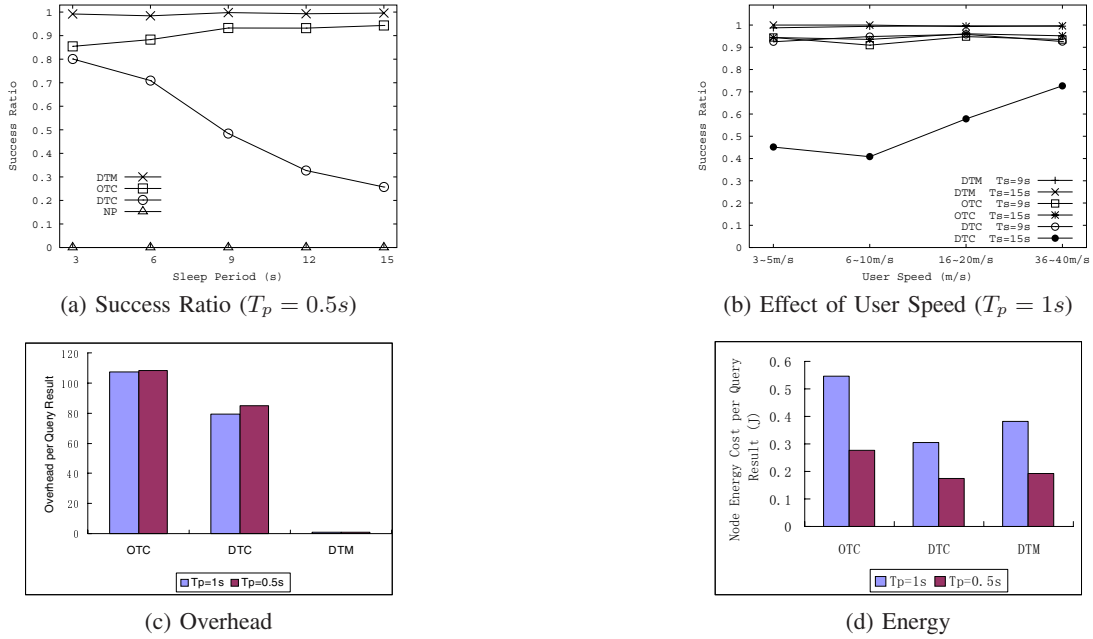


Fig. 3. Comparison of protocols under Accurate Motion Profile.

Applying (3) in (4), we see that the timing constraints of the query will be met, if

$$nT_p \geq \alpha \left( \frac{nT_p v_{max} + R_q}{R_c} + 1 \right) \tau + T_s + T_d \quad (5)$$

In our protocol we use the minimum  $n$  that satisfies this inequality, denoted as  $n^*$ . Thus,

$$n^* = \frac{\alpha(R_q + R_c)\tau + (T_s + T_d)R_c}{T_p(R_c - \alpha\tau v_{max})} \quad (6)$$

The radius of the wake-up area is thus obtained by replacing  $n^*$  from the above equation in (2).

#### IV. SIMULATIONS

In this section, we present the simulation results on ns2. In the simulations, Coverage Configuration Protocol (CCP) [17] based on IEEE 802.11 Power Saving Mode (PSM) with the extension from [18] is used as the power management protocol. CCP maintains coverage and network connectivity through the backbone of active nodes. The duration of the active window of the sleeping nodes is set to 100ms, the radius of each query area is set to 150m, the node bandwidth is set to 2 Mbps, and the communication and sensing range of the network nodes are set to 105m and 50m, respectively. The results in this section are the averages over 3 runs with different network topologies. We use the following metrics in our performance evaluation: (1) *Data fidelity*, defined as the ratio of the number of nodes that contribute to a query result to the total number of nodes in a query area. (2) *Success ratio*, defined as the ratio of the number of queries that meet deadlines and have data fidelity above a threshold, to the total number of queries. The success ratio indicates the overall quality of service received by the user.

##### A. Performance with Accurate Motion Profile

In this subsection, we evaluate the performance of our protocols when the user's motion profile accurately describes the future path of the user. We set the threshold for data fidelity to 90%, in the success ratio metric. The performance with inaccurate motion profile is studied in Section IV-B. We compare our protocols against a

baseline protocol: No-prefetching (NP). In NP, the user broadcasts a query to the network at the beginning of each query period and receives the query results before the current query deadline. In each simulation, 200 nodes are randomly distributed in a 450m × 450m region and the user starts from a corner of the region and moves in a random direction with a speed randomly chosen between 3m/s and 5m/s, which corresponds to the speed of a walking or jogging human being. The user changes its direction and speed every 50 seconds. The motion profile that specifies the complete user path is provided to each protocol at the beginning of the simulation. Each simulation lasts for 400 seconds.

**Effect of Sleep Periods:** Figure 3(a) shows the average success ratios of all protocols with different sleep periods, when the query period is 0.5s. Freshness requirement in all simulations is 0.5s. The success ratio of NP remains below 10% in all settings, which clearly indicates that the prefetching mechanism used by other protocols is crucial in networks with low duty cycles. The performance of DTC drops quickly when sleep period increases. Due to its high communication overhead, more packets are dropped when the effective network bandwidth becomes lower as the sleep period increases. In sharp contrast, DTM achieves a success ratio of nearly 100% in all settings. OTC maintains a success ratio of above 90% in most settings. Its performance degrades slightly when the sleep period becomes shorter. OTC wakes up sleeping nodes before a tree is created. As the sleep period becomes shorter, more traffic (i.e., overhead packets of 802.11 PSM and CCP) is produced by the active nodes in the network, resulting in higher network contention during tree creation. In contrast, the performance of DTM is not affected by the sleep period since it requires much lower bandwidth.

**Effect of User Speed:** Figure 3(b) shows the performance of the different protocols when the user moves at different speeds. Four speed ranges: 3m/s ~ 5m/s, 6m/s ~ 10m/s, 16m/s ~ 20m/s and 36m/s ~ 40m/s are used, which correspond to the speeds for walking or jogging, running, driving at moderate speed and driving at high speed, respectively. Although DTC delivers above 90% of the query results under all user speeds when the sleep period is 9s, its performance degrades significantly when the sleep period is 15s. This

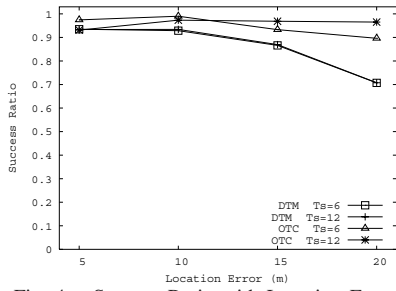


Fig. 4. Success Ratio with Location Errors

is because DTC fails to wake up many sleeping nodes in time due to the network congestion caused by its high communication overhead. On the other hand, the success ratio of both DTM and OTC remains above 90% under all settings. This result shows that both protocols adapt successfully to different ranges of user speed by waking up sleeping nodes in advance, with moderate communication overhead.

**Overhead:** We define the overhead as the total number of messages (except query messages and query results) sent by a protocol, normalized by the total number of query results received by the user. Figure 3(c) shows the different protocol overheads when the sleep period is 9s. As seen in the figure, DTM incurs much lower communication overhead than the other protocols. This is due to the low overhead of query tree maintenance in DTM. OTC, on the other hand, incurs the highest overhead among the three protocols since it wakes up a large number of nodes due to lack of knowledge of the user motion profile. Even though OTC has a higher communication overhead, it is offset by its better performance in comparison to DTC, as seen in Figure 3(a).

**Energy Consumption:** We used the energy model of Cabletron Roamabout 802.11 network card, measured in [18]. In accordance with this model, the power consumption of transmit, receive, idle and sleeping modes were set to 1400mW, 1000mW, 830mW and 130mW respectively. Figure 3(d) shows the per node energy expenditure for the different protocols, normalized by the total number of query results received by the user, when the sleep period is 9s. As seen in the figure, OTC has the highest energy expenditure among the protocols. This is expected, since in OTC, a large number of nodes are woken up due to the lack of a user motion profile. DTM has slightly more energy expenditure than DTC for a query period of 1s and similar energy expenditure for a query period of 0.5s. The advantage of DTM however lies in its high success ratio and low overhead, as seen in figures 3 (a) and 3(c).

#### B. Performance under Inaccurate Motion Prediction

In this section, we study the performance of the proposed protocols when the motion profile is unknown. As simulation results in Section IV-A show, NP and DTC cannot provide satisfactory performance even when the user path is known. Hence we focus on the performance of OTC and DTM in this section. The accuracy of the motion prediction is inherently affected by the location error and the motion pattern of the user. We evaluate these effects in the rest of this subsection. In this section, we set the threshold of data fidelity to 80% for the success ratio metric.

**Effect of Location Error:** Figure 4 shows the performance of DTM and OTC under different location errors. As expected, location error has minimal impact on OTC, since OTC does not make use of the user motion profile, and always wakes up nodes in a large area around the user that covers all possible user locations during the next few query periods. DTM, on the other hand, performs very well for location errors less than 15m (typical GPS location error is 10m-15m), and is able to deliver 80% of the query results successfully.

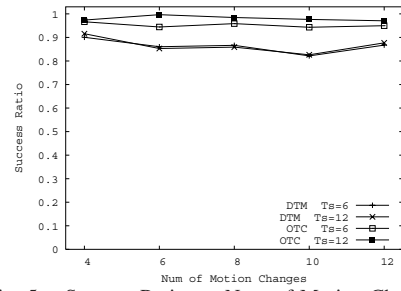


Fig. 5. Success Ratio vs. Num of Motion Changes

These results show that the design of DTM is robust to moderate location errors.

**Effect of Motion Changes:** Next, we evaluate the performance of the protocols under different number of changes in the user's velocity. In these simulations, the user moves in a straight line at a constant speed until it makes a turn and chooses a new speed. Figure 5 shows the performance of the protocols when the number of turns increases from 4 to 12 within a simulation time of 500s. The location error for all the simulation runs is fixed at 10m.

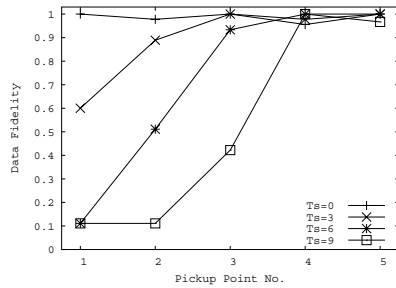
As expected, the performance of OTC is not affected by the user motion changes. DTM performs slightly worse than OTC, because when the user makes a turn, the partial query tree that resides around the original predicted path becomes invalid and new nodes have to be woken up in order to form a query tree around the new predicted path. Consequently, some sleeping nodes along the new path may not be woken up early enough to respond to the query. However, DTM still maintains a success ratio over 80% in all settings.

## V. IMPLEMENTATION ON MOTES

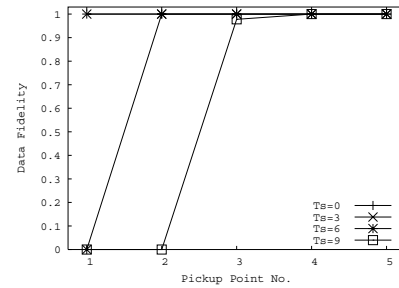
We implemented MobiQuery, on Mica2 motes using both DTC and DTM. MobiQuery was used to provide a moving user with a temperature map of a specified region around the user. The experimental setup consisted of 18 motes arranged on a table in a 6x3 grid. Due to the small area, logical multihop was used to obtain a more realistic scenario. The clocks on the motes were synchronized at the start, and a backbone of active nodes was setup, with non-backbone nodes following a synchronized sleep schedule. The user was emulated by an Acroname PPRK robot carrying a Stargate that collected query results from the network and transmitted them to a laptop for display. The robot was preprogrammed to move across the sensor network along a predetermined path. The precomputed robot path was fed to both DTC and DTM.

Experiments were conducted for  $T_p = 3s$ ,  $T_f = 1s$ , sleep schedules of 3s, 6s and 9s with an active window of 1s, and without sleep schedule. The protocol performance was evaluated in terms of data fidelity. Due to the small sensor network deployed, only 5 query periods were set up. The results were obtained from the motes at the right time, in accordance with the design.

Figure 6 shows the data fidelity obtained at each query period, averaged over 5 runs, for DTC and DTM. As is shown in the two graphs, after an initial expected warmup interval, in which some sleeping nodes have not yet been alerted about the query, both algorithms reach a stable state with high data fidelity. Due to the limited data points neither the difference in the performance of the two algorithms nor the stable working phase of the algorithms is clearly visible in the figures. However, it can be observed that after the warmup interval, DTM immediately achieves 100% data fidelity while DTC slowly achieves good data fidelity. The slow increase in data fidelity under DTC is due to contention caused by overlapping



(a) DTC



(b) DTM

Fig. 6. Data Fidelity

query tree setup as well as data collection.

## VI. CONCLUSION

We have presented two novel wake-up and topology maintenance protocols, DTM and OTC. They can be integrated with mission-critical applications such as object-tracking and spatiotemporal query. The simulation results show that DTM can maintain satisfactory performance with minimum communication overhead even when stringent spatiotemporal constraints are imposed by the application. OTC also achieves very good performance and is more robust to location error and user motion changes but it has higher communication overhead and energy consumption. In the future, we plan to compare our protocols to existing protocols as well as evaluate our protocols on a larger testbed under more realistic settings.

## ACKNOWLEDGEMENT

This work is funded in part by the NSF under an ITR grant CCR-0325529 and the ONR under MURI research contract N00014-02-1-0715. We also thank the reviewers for their valuable feedback.

## REFERENCES

- [1] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler, "The mote revolution: Low power wireless sensor network devices," in *Hot Chips 16*, 2004.
- [2] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-efficient surveillance system using wireless sensor networks," in *MobiSys'04*.
- [3] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated, adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, 2004.
- [4] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys'03*.
- [5] C. Schurgers, V. Tsiatsis, and M. B. Srivastava, "Stem: Topology management for energy efficient sensor networks," in *MobiHoc'02*.
- [6] "Wireless lan medium access control and physical layer specifications," Aug. 1999, IEEE802.11 Standard (IEEE Computer Society LAN MAN Standards Committee).
- [7] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys'04*.
- [8] D. Li, K. Wong, Y. H. Hu, and A. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 19 (2), 2002.
- [9] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, 2002.
- [10] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *SenSys'03*.
- [11] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," *IEEE Transactions on Wireless Communication*, vol. 3 (5), 2004.
- [12] S. Patten, S. Poduri, and B. Krishnamachari, "Energy-quality tradeoffs for target tracking in wireless sensor networks," in *IPSN'03*.
- [13] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *MobiCom'04*.
- [14] Q. Huang, C. Lu, and G.-C. Roman, "Spatiotemporal multicast in sensor networks," in *SenSys'03*.
- [15] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, "A spatiotemporal query service for mobile users in sensor networks," in *ICDCS'05*.
- [16] A. R. Aljadhah and T. Znati, "Predictive mobility support for qos provisioning in mobile wireless environments," *JSAC*, vol. 19(10), 2001.
- [17] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. D. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *SenSys'03*.
- [18] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *MobiCom'01*.
- [19] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *MobiCom'01*.
- [20] G. Xing, C. Lu, R. Pless, and Q. Huang, "On greedy geographic routing algorithms in sensing-covered networks," in *MobiHoc'04*.