12-2006

Continuous monitoring of kNN queries in wireless sensor networks

Yuxia YAO

Xueyan TANG
Nanyang Technological University

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

DOI: https://doi.org/10.1007/11943952_56

Continuous Monitoring of kNN Queries in Wireless Sensor Networks

Yuxia Yao, Xueyan Tang, and Ee-Peng Lim

School of Computer Engineering
Nanyang Technological University
Singapore 639798
{yaoy0003, asxytang, aseplim}@ntu.edu.sg

Abstract. Wireless sensor networks have been widely used for civilian and military applications, such as environmental monitoring and vehicle tracking. In these applications, continuous query processing is often required and their efficient evaluation is a critical requirement to be met. Due to the limited power supply for sensor nodes, energy efficiency is a major performance measure in such query evaluation. In this paper, we focus on continuous kNN query processing. We observe that the centralized data storage and monitoring schemes do not favor energy efficiency. We therefore propose a localized scheme to monitor long running nearest neighbor queries in sensor networks. The key idea is to establish a monitoring area for each query so that only the updates relevant to the query are collected. Experimental results show that our scheme outperforms the centralized scheme in terms of energy efficiency and network lifetime.

1 Introduction

The development of wireless technology and sensors have enabled wide use of sensor networks. In these networks, a large number of low-powered sensor nodes are distributed in an area of interest and wirelessly connected. The sensor nodes are equipped with computation and communication capabilities [13]. Sensor networks are popular for a variety of applications, e.g., habitat monitoring, pollution monitoring, and object tracking [2,9]. Data collection and query processing in sensor networks are challenging research topics in sensor network database management. Existing work has focused on non-spatial query processing [8,16]. To the best of our knowledge, there has been little work on spatial query processing [19,21], especially spatial query monitoring. In this paper, we consider monitoring kNN queries in an object tracking sensor network.

Energy efficiency is a critical design consideration in wireless sensor networks. The sensor nodes usually with low battery power have to be deployed unattended for a long time. To prolong the network lifetime, we need to reduce network-wide energy consumption. Energy is mainly consumed during communication [13]. Thus, to reduce energy consumption, we need to reduce the number of message transmissions. Meanwhile, it is also important to balance energy consumption across sensor nodes since the sensor network may be disconnected and fail to operate properly if some nodes run out of energy and fail to communicate. A straightforward *centralized scheme* for monitoring kNN queries is to continuously send the sampled locations of objects to a base station.

J. Cao et al. (Eds.): MSN 2006, LNCS 4325, pp. 663–674, 2006.

User queries are also routed to the base station for initial and continuous evaluations. However, the centralized scheme is likely to suffer from unnecessary update messages. This is because kNN queries are usually *localized* in that only the locations of objects close to the query points need to be reported for kNN query processing and the objects farther away can be exempted from location updates. Moreover, in the centralized scheme, the energy consumption is highly unbalanced among the sensor nodes. Sensor nodes closer to the base station consume much more energy due to message relay and this would reduce the network lifetime. To improve energy efficiency, it is desirable to store data locally at the sensor nodes in a distributed manner and process the queries *in-network* [6,18,21]. In this way, we hope to extract only the relevant data from the network and cut down the communication cost compared to the centralized scheme.

Motivated by the localized property of kNN queries, we propose a *localized scheme* to continuously evaluate kNN queries in sensor networks. Each query is characterized by a geographical location q called the *query point*, and a number k of the required nearest neighbors. The objective of a kNN query over a set of objects O is to identify the k objects with the shortest distances to the query point, i.e., to find an ordered subset of k objects $\mathcal{N} = \{o_1, o_2, ..., o_k\} \subseteq O$ such that $\forall o_i \in \mathcal{N}$ and $\forall o \in O - \mathcal{N}$, $d(o_i, q) \leq d(o, q)$ and $\forall i < j$, d(o_i, q) $\leq d(o_j, q)$, where $d(o, q)$ denotes the Euclidean distance between an object o and the query point q. Our key idea is to establish a monitoring area for the query so that only the relevant updates are collected. In this way, we reduce the network-wide energy consumption and avoid hotspots in the network.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 introduces some preliminaries and Section 4 presents the localized scheme to continuously evaluate kNN queries. Section 5 describes the experimental setup and discusses the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

With the growing needs for location-based services, continuous monitoring of kNN queries is becoming more popular in spatial databases [4,10,11,14,17,22]. Recently, some grid-based methods are explored in continuous monitoring of kNN queries. Examples include YPK-CNN [22], CPM [10] and SEA-CNN [17]. These methods assume that there is a centralized repository to store all object locations and all location updates are simply reported to the centralized repository. However, such centralized storage is costly for object tracking sensor networks due to their energy constraints. Therefore, these methods are not appropriate for kNN monitoring in sensor networks.

Other relevant works include the MobiEyes algorithm proposed by Gedik [4] and a threshold-based algorithm proposed by Mouratidis [11]. Similar to YPK-CNN, SEA-CNN and CPM, there is a centralized server in the system. But differently, the MobiEyes and threshold-based algorithms assume *smart* objects that have some storing and processing capabilities. When an object moves away from its current position, the object can decide whether to send the location update to the server or not. Both the MobiEyes and threshold-based algorithms aim at reducing the communication cost between the objects and the server by eliminating unnecessary location updates. MobiEyes [4] focuses on monitoring range queries by assigning a safe region to each query. The objects

within the safe region periodically check whether they are in the query range. Only the objects within the query range report their locations to the server. The threshold-based algorithm [11] assigns a distance range to each object in the result set. The distance range for the ith nearest object is defined by two thresholds: the midpoint between the ith and the $(i-1)$th nearest objects and the midpoint between the ith and the $(i+1)$th nearest objects. Only when an object moves out of its distance range is the location update of the object sent to the server. However, in both [4,11], all the queries are still processed at one centralized server. In contrast, in this paper, we make use of the localized property of kNN queries to process them in-network.

3 Preliminaries

3.1 System Model

We consider a sensor network with sensor nodes distributed over a 2-dimensional space. The sensor nodes are aware of their locations through GPS [3] or other localization algorithms [12]. Each sensor node can communicate directly with the nodes (called neighbors) within the distance r_{tx} of radio communication. Through message exchange, each sensor node is aware of the geographical locations of its neighbors. We assume the network is connected, i.e., any sensor node can communicate with any other sensor node either directly or indirectly through a routing protocol. The sensor nodes detect moving objects within their sensing range r_s and sample their locations periodically. We assume a dense sensor network in which a geographical area of interest is fully covered by the sensing ranges of the sensor nodes. Instead of sending all collected location data to a central repository, we propose to store them locally at the detecting sensor nodes [18,20]. Recall that each kNN query specifies a query point q. A continuous kNN query issued by the user is injected into the sensor network at any sensor node and forwarded to q through GPSR routing [7,15]. The sensor node closest to q would receive the query. This sensor node is called the *query initiator*. Our objective is to continuously collect the kNN result at the query initiator which in turn returns it to the user.

3.2 One-Shot Remainder kNN Query Processing

We first consider the processing of a generalized one-shot kNN query called *remainder kNN query*. A remainder kNN query finds k nearest objects to a given query point q among the objects beyond a given distance r from q. When $r = 0$, a remainder kNN query reduces to the original kNN query. Remainder kNN queries are used for query reevaluation in kNN monitoring (refer to Section 4). The evaluation of a one-shot remainder kNN query proceeds in two phases: (i) *preliminary search* and (ii) *expanded search*. The purpose of the preliminary search is to find a *boundary object* and define the search space. In this step, the sensor nodes surrounding the query initiator are visited by message passing until at least k objects are collected. Among the k objects detected, the kth closest one to the query point is selected as the boundary object. A search space is defined based on the location of the boundary object to guarantee that it includes all sensor nodes possibly detecting an object closer to the query point than the boundary

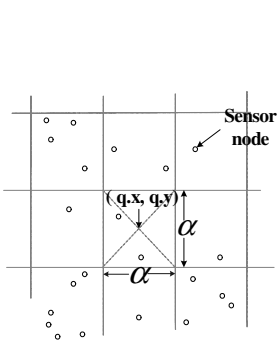

Fig. 1. Grid Structure in Sensor Networks

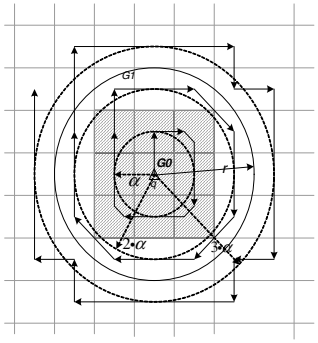

Fig. 2. Msg. Routing Path in Preliminary Search

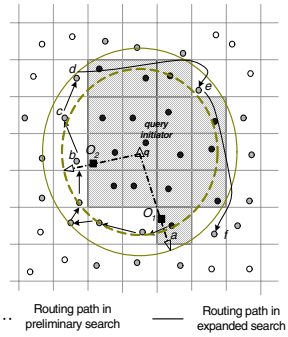

Fig. 3. Expanded Search

object. During the expanded search, the sensor nodes in the search space that are not yet visited in the preliminary search are visited to locate the k nearest objects. Finally, the query result is routed back to the query initiator.

To facilitate message traversal among the sensor nodes, the sensor network is partitioned into a set of grid cells. As shown in Figure 1, each grid cell is a square of size $\alpha \times \alpha$. For a query (q, k), the grid structure is constructed by designating q as the centroid of a grid cell. Then, given any sensor node located at (x, y) on the plane, the centroid of the grid cell containing (x, y) is given by $\big(q.x + \frac{\alpha}{2} + \frac{1}{2} \cdot (\lfloor \frac{x-(q.x+\frac{\alpha}{2})}{\alpha} \rfloor \cdot \alpha + \lceil \frac{x-(q.x+\frac{\alpha}{2})}{\alpha} \rceil \cdot \alpha), q.y + \frac{\alpha}{2} + \frac{1}{2} \cdot (\lfloor \frac{y-(q.y+\frac{\alpha}{2})}{\alpha} \rfloor \cdot \alpha + \lceil \frac{y-(q.y+\frac{\alpha}{2})}{\alpha} \rceil \cdot \alpha)\big)$.

The preliminary search and expanded search are carried out by visiting a series of grid cells. When visiting grid cell G, one sensor node (called the R-node) is responsible for collecting the object locations detected by the sensor nodes in the cell. Once receiving the query, the R-node broadcasts a one-hop *probe* message to the sensor nodes in G. To guarantee that all sensor nodes within G can hear the probe message, the diameter of the grid cell (i.e., $\sqrt{2}\alpha$) should be less than the transmission range r_{tx}. Therefore, we set $\alpha = \frac{1}{\sqrt{2}} \cdot r_{tx}$ and the R-node of a grid cell to be within $\frac{1}{2} r_{tx}$ to the centroid of the cell. The query point and the centroid location of G are included in the probe message broadcast by G's R-node. Each sensor node knows the value of α based on r_{tx} and thus knows the centroid of the grid cell containing itself autonomously. Only sensor nodes in G will reply to R-node with the detected object locations if any.

After collecting the data from the sensor nodes in G, the R-node processes the object locations accordingly (see Sections 3.3 and 3.4 for details) and continues forwarding the query message to the next grid cell. To select the R-node of the next grid cell G' to be visited, the R-node of G checks whether any of its neighbors is within distance $\frac{1}{2} r_{tx}$ to the centroid of G'. If multiple such neighbors exist, the one closest to the centroid of G' is selected as the R-node. If there is no such neighbor, the message is routed to the centroid of G' by GPSR routing. The sensor node S closest to the centroid would receive the query message and become the R-node. For simplicity, we assume a dense sensor network where there is at least one sensor node in each grid cell. So, S must be within distance $\frac{1}{2} r_{tx}$ to the centroid of G'. Due to space limitation, the handling of empty grid cells will be discussed in the extended version of this paper.

3.3 Preliminary Search

In the preliminary search, we need a rule to determine the visiting order of grid cells. Since the location of boundary object determines the search circle for expanded search, to reduce search cost, we would like the boundary object to be as close to the query point q as possible. Thus, it is intuitive to visit the grid cells based on their distances to q. We propose a *circle* approach to determine the visiting order of grid cells.

Specifically, the search is divided into rounds. In each round i ($i \geq 1$), the unvisited grid cells intersecting with the circle centered at q and with a radius of $i \cdot \alpha$ are visited in clockwise order (see the dash line circle in Figure 2). The query message contains the location of the query point q. Note that given the location of q, each sensor node can determine autonomously which grid cell to visit next. In case of $r = 0$, the query starts from the cell G centered at q. In case of $r > 0$, the query starts from a grid cell G' which is not within the circle centered at the query point q and with radius r. To determine this starting grid cell, we list all grid cells from round 1 to round $\lceil \frac{r}{\alpha} \rceil$ in a clockwise order. The first grid cell in the list whose maximum distance to q exceeds r is selected as the starting grid cell. Figure 2 shows the message routing path when $r = 0$ and $r > 0$. When $r = 0$, the preliminary search starts from G_0. When $r > 0$, the grid cells within the circle with radius r are labelled as shadow. They are exempted from being visited and the preliminary search starts from G_1. The query message is then routed to the centroid of G_0 or G_1 and received by the R-node. The R-node of each visited grid cell collects object locations from the sensor nodes in the cell and records them in the query message. The preliminary search completes when the number of collected objects is no less than k. Among these objects, the kth object closest to the query point q is chosen as the boundary object. The search space is then defined as a circle centered at q and with a radius of $d = r_b + r_s$, where r_b is the distance between the boundary object and q, and r_s is the sensing range. Intuitively, if the minimum distance between a grid cell and the query point q is smaller than d (the radius of the search circle), the sensor nodes in the grid cell are likely to detect objects less than distance r_b away from q.

3.4 Expanded Search

In expanded search, a *search list* is given by all grid cells within or intersecting with the search circle, excluding those already visited in the preliminary search. The query message passed between the grid cells in the expanded search contains the search list, the locations of the k recorded objects, and the query point q. When the R-node of a grid cell G receives the query message, it first removes G from the search list. After sending a probe message and collecting object locations from the other sensor nodes in G, one of the following three cases can occur at the R-node: (i) no object is detected by any sensor node in G; (ii) all objects detected are further away from the query point q than the boundary object; (iii) at least one object detected is closer to q than the boundary object. In cases (i) and (ii), the search list and the objects recorded in the message do not change. In case (iii), the detected objects nearer to q than the boundary object are used to update the k nearest objects recorded in the message. Meanwhile, the boundary object is updated as the new kth nearest object and the search circle is shrunk accordingly. The search list is then updated by removing all grid cells outside the new search circle. On finishing with a grid cell G, the query message is routed to the cell on the search list

that is closest to G. The expanded search continues until the search list becomes empty. On completion of the expanded search, the message is routed to the query initiator and the locations of k recorded objects are returned to the user as the query result.

Figure 3 shows an example of 1NN query processing. The grid cells in shadow are visited in the preliminary search. Suppose the boundary object O_1 is found by R-node a. Node a determines the search circle (shown by the outer solid circle in Figure 3) and derives the set of grid cells in the search list (i.e., all the grey R-nodes in Figure 3). The query message is passed among the gray R-nodes of grid cells in the search list, and the object locations are collected. Suppose that at R-node b, a nearer boundary object O_2 is found. Then, the search circle is shrunk accordingly. The dotted circle in Figure 3 is the new search circle. The grid cells containing R-nodes c, d, e and f are now the only four unvisited grid cells left in the revised search list. After visiting the grid cell with R-node f, the search list becomes empty and the expanded search completes.

4 Localized Scheme for Continuous kNN Queries

4.1 Overview

The set of kNNs and their locations may change over time as the objects move. To continuously report the kNN result to the query initiator, we propose a localized scheme to monitor the updates of object locations in kNNs. The key idea of the localized scheme is to collect only the location updates that may potentially affect the kNN result at the query initiator. To this end, a *monitoring area* is setup for a continuous kNN query in the sensor network. It is defined as a circle centering at the query point q. The radius of the circle is set to $d(o_k, q) + r_s$, where $d(o_k, q)$ is the distance between the kth nearest object o_k and q, and r_s is the sensing range. Sensor nodes within the monitoring area will report all detected objects at each sampling interval to the query initiator. Upon receiving these reports, the query initiator reevaluates the kNN query. For simplicity, we assume the sampling interval is long enough for all necessary messaging to occur in order to complete the reevaluation of kNN query. On the other hand, the sensor nodes outside the monitoring area do not report their location updates.

This localized scheme for continuous kNN query processing requires the query initiator to centrally coordinate the maintaining of the monitoring area, collecting object updates from the monitoring area, and reevaluating the query results. In the first sampling interval, the query initiator evaluates one-shot kNN query, which is equivalent to a remainder kNN query for q with $r = 0$, using the scheme described in Section 3.2. The monitoring area is setup simultaneously during the query evaluation. Recall that during the query evaluation, i.e., preliminary and expanded search, all grid cells within or intersecting with the circle centering at q and with radius $d(o_k, q) + r_s$ are visited. When a grid cell is visited, the R-node will broadcast a probe message for data collection. Besides the location of the query point, the identity of the query initiator is also included in the probe message. To this end, the probe message is also an "include" notification message which helps setup the monitoring area. All sensor nodes in the grid cell can receive the probe message. They will start reporting the location updates to the query initiator from the following sampling interval.

At each subsequent sampling interval, the query initiator collects the locations of the objects detected and reported by the sensor nodes in the monitoring area. Objects beyond distance $d(o_k, q)$ are removed by the query initiator and only objects within distance $d(o_k, q)$ from the query point q are retained. We denote the number of these objects by k'. If $k' \geq k$, the query reevaluation is simply done at the query initiator. A set of new kNNs are derived by ordering the k' objects according to their distances to the query point and selecting the first k objects. If $k' < k$, we need to search $k-k'$ more objects outside the circle centered at q and with a radius of $d(o_k, q)$. This is equivalent to a remainder $(k - k')$NN query for q with $r = d(o_k, q)$. This query can be evaluated using the scheme described in Section 3.2.

4.2 Maintenance of the Monitoring Area

On computing the one-shot kNN result in the first sampling interval, the query initiator can set the radius of the monitoring area at $d(o_k, q) + r_s$, i.e., the distance between the kth nearest object and the query point plus the sensing range. The monitoring area is setup during the query evaluation.

At the subsequent sampling interval, the monitoring area may need to be updated due to the change in the kNN result. If the new kth nearest object is further away from the query point than the old one, the monitoring area should be expanded to include the new kth nearest object. That is, the radius of the new monitoring area is $d(o'_k, q) + r_s$, where $d(o'_k, q)$ is the distance of the new kth nearest object to the query point q. In case of monitoring area expansion, the difference between the new and old areas is a ring whose inner radius is the radius of the old monitoring area and whose outer radius is the radius of the new monitoring area. Sensor nodes in the ring are notified to report their location updates starting from the following sampling interval. Note that the monitoring area expansion only happens when the number of reported objects within distance $d(o_k, q)$ to q, i.e., k', is less than k. A new set of kNNs are derived through a remainder $(k-k')$NN query with $r = d(o_k, q)$. Since all grid cells within or intersecting with the ring are visited either in preliminary search or expanded search, it is guaranteed that all sensor nodes newly included in the monitoring area are notified.

On the other hand, if the new kth nearest object is closer to the query point than the old one, the monitoring area can be shrunk to reduce the number of sensor nodes reporting object locations to the query initiator. In this case, the difference between the new and old monitoring areas is a ring whose inner radius is the radius of the new monitoring area and whose outer radius is the radius of the old monitoring area. Sensor nodes in the ring need to be informed to stop reporting location updates to the query initiator. To do so, an "exclude" notification message is sent to all sensor nodes in the ring. The notification message takes the parameters of the query point q, the inner and outer radius of the ring and a flag (i.e., exclude) to indicate the action. The notification message is first sent to a sensor node within the ring. After reaching the sensor node in the ring, the message is flooded to all sensor nodes in the ring. When a sensor node within the ring receives the notification message for the first time, the sensor node further broadcasts the message to all its neighbors. Otherwise, if the sensor node receiving the message is outside the ring or it has already received the message before, the message is dropped. Sensor nodes receiving the message will stop reporting their location updates.

Since notification messages are sent to update the monitoring area, it incurs message overhead. There is in fact a tradeoff between the cost of updating the monitoring area and the saving in location updates from the sensor nodes in the area. We propose two methods to deal with the shrinking of the monitoring area. One *naive* method is to shrink the monitoring area to a minimum circle covering the new kNNs and their detecting sensor nodes whenever the new kth nearest object is found to have moved closer to q. The naive method is intuitive and it aggressively eliminates unnecessary location updates. However, it may lead to an expansion of the monitoring area soon after the kth nearest object moves further away from q again. As a result, the cost of updating the monitoring area may exceed the saving of location updates.

As an improvement to the naive method, we propose an *adaptive* method by considering the tradeoff between the saving of location updates and the cost of updating the monitoring area. Assume the radius of the monitoring area at current sampling interval is r_{old}. After updating the answer set, the kth nearest object moves nearer to q than the old one. Let $r_{new} = d(o'_k, q) + r_s$ be the radius of the new monitoring area if it is shrunk, where $d(o'_k, q)$ is the distance between the new kth nearest object and q. Let $\mathcal{S}$ be the set of sensor nodes in the ring defined by two circles with radii r_{old} and r_{new}. Let $\mathcal{O}$ be the set of objects detected and reported by the sensor nodes in $\mathcal{S}$. Note that the query initiator is aware of $\mathcal{O}$ by checking the location updates collected during current sampling interval. The expected saving of object location updates in the next sampling interval (in terms of message complexity) is $C_{saving} = \sum_{o_i \in \mathcal{O}} d(o_i, q)/r_{tx}$, where $d(o_i, q)$ is the distance between the object o_i and the query point q and r_{tx} is the communication range. The expected cost of updating (i.e., shrinking) the monitoring area consists of two parts: the cost of sending the "exclude" notification message from the query initiator to a sensor node within the ring; the cost of flooding the notification message to all sensor nodes in the ring. The expected cost of the first phase is r_{new}/r_{tx}. The expected cost of the second phase is approximated by the number of sensor nodes in the ring: $\pi(r_{old}^2 - r_{new}^2) \cdot f$, where f is the sensor node density and $\pi(r_{old}^2 - r_{new}^2)$ is the area of the ring. Therefore, the expected cost of updating the monitoring area is $C_{updating} = r_{new}/r_{tx} + \pi(r_{old}^2 - r_{new}^2) \cdot f$. In the adaptive method, if $C_{saving} \leq C_{updating}$, the new monitoring area is kept unchanged. Otherwise, if $C_{saving} > C_{updating}$, the monitoring area is shrunk.

5 Performance Evaluation

5.1 Experimental Setup

We simulated the sensor networks continuous kNN query processing using a simulator called J-Sim [1]. We simulated a connected sensor network geographically covering a $1000m \times 1000m$ area. A number of 2500 sensor nodes were deployed in the sensor network, implying that on average, there was one sensor node in an area of $400m^2$. The transmission and the sensing range for each sensor node were set at $35m$ and $28m$ respectively [5]. The maximum number of retransmissions at the MAC layer was set at 7. The sensor nodes were randomly deployed in the sensor network. On average, each sensor node can communicate directly with 9 neighbors. Since this paper focuses on query processing, we do not include the energy consumed in tracking objects. Only the

Table 1. Message Types and Sizes

type	size	type	size	type	size
query	>20 bytes	probe	16 bytes	query result	$k \cdot 24$ bytes
probe reply	24bytes	object update	24 bytes	monitoring area update	36 bytes

energy consumption of message exchanges was counted. The energy used to transmit and receive a bit was set at $2.64 \times 10^{-6}J$ and $1.58 \times 10^{-6}J$ respectively. The initial energy budget for each sensor node was set at $30J$. The default number of objects to be tracked was set at 250. The objects were initially placed in the network at random. The object movement follows a random walk model. In this model, the object repeatedly picks a random destination in the network and moves to the destination at a speed randomly chosen from a range $[0, V_{max}]$. V_{max} was set at $40m$ per time unit. The object locations were sampled by the sensor network at every time unit. In our experiments, we assume that the detecting sensor node of an object is the one closest to the object [23]. The interval of the GPSR beacon message is set at 1 time unit.

We compared the proposed localized scheme (naive and adaptive) with the centralized scheme. In the centralized scheme, we assume that there is a base station deployed at the centroid of the sensor network. All location updates are reported by the detecting sensor nodes to the base station at each sampling interval. Queries injected from any sensor node are routed to the base station for initial evaluation and are reevaluated at the base station continuously. If the new kNNs differ from the previous kNNs, the new result will be sent from the base station to the query initiator. Due to space limitation, we shall present only a set of most informative results. To evaluate the performance, we will show the average energy consumption in the sensor network as well as the message complexity. Table 1 shows the major message types and respective sizes.

5.2 Impact of the Number of NNs

In this section, we investigate the two methods for monitoring area update in the localized scheme, i.e., naive and adaptive methods. Figure 4 shows the average energy consumption of a single query for the localized and the centralized scheme with different k's. The average energy consumption is derived at the 65th time unit, i.e., the network lifetime under the centralized scheme with $k = 1$. From the figure we can see that for localized scheme, the adaptive method generally outperforms the naive method, especially for large k's. This is because the size of the monitoring area is generally larger with a larger k. The naive method simply shrinks the monitoring area whenever the kth nearest object moves closer to the query point. Thus, it incurs high volume of notification messages to update the monitoring area. On the other hand, the adaptive method considers the tradeoff between the cost of updating the monitoring area and the saving of object location updates, and shrinks the monitoring area only when it is beneficial. Table 2 presents the breakdown of the messages for the naive and adaptive method at the 65th time unit with $k = 8$. It shows that the naive method induces much higher number of messages in query processing (query message and probe message) and monitoring area update. Thus, we use the adaptive method throughout the remaining experiments.

Figure 4 also shows the impact of the number of NNs. With a given number of objects, the average energy consumption of the sensor nodes increases with k. This is

Table 2. Breakdown of Messages

Message	query	query result	probe	probe reply	object update	monitoring area update
k=8 (naive)	8813	525	1705	217	11957	224
k=8 (adaptive)	756	525	179	6	12297	13
Centralized (k=8)	6	443	0	0	258347	0

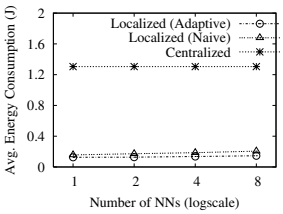

Fig. 4. Avg. Energy Consumption v.s. No. of NNs

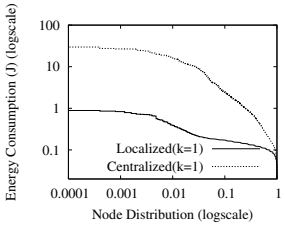

Fig. 5. Distribution of Energy Consumption v.s. No. of NNs

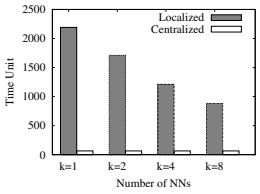

Fig. 6. Network Lifetime

because a larger monitoring area is established when more NNs are required. As a result, more location updates are sent to the query initiator at each sampling interval. Figure 4 also compares the average energy consumption for the localized and centralized schemes with different k's. The energy consumption is derived at the 65th time unit, i.e., the network lifetime under the centralized scheme with $k = 1$. We show only the performance of the centralized scheme with $k = 1$ in Figure 4. For the localized scheme, the average energy consumption for $k = 1, 2, 4, 8$ are given. Compared to the centralized scheme, the average energy consumption of the localized scheme is much smaller. This is because in the localized scheme, only relevant object location updates were reported to the query initiator, while in the centralized scheme, all location updates were sent to the base station. The number of sensor nodes involved in sending the updates was much smaller in the localized scheme than the centralized scheme which led to a lower average energy consumption. To gain more insight of how the localized scheme improves, we also present the breakdown of messages for the centralized scheme at the 65th time unit with $k = 8$ in Table 2. For the centralized scheme, the number of object updates was 20 times more than that of the localized scheme. For the localized scheme, the message overhead of query processing and monitoring area update was much smaller than the messages for object updates in the centralized scheme.

Figure 5 shows the energy distribution for the localized and centralized schemes with $k = 1$. The energy consumption is derived at the 65th time unit, i.e., the network lifetime under the centralized scheme with $k = 1$. A point (x, y) on the curve means that a fraction x of all sensor nodes consume more than yJ energy each. For the centralized scheme, among all sensor nodes, the top 1 percentile energy consumption is $16.3J$ which is 7 times higher than the top 10 percentile energy consumption (i.e., $2.3J$). For the localized scheme, the top 1 percentile energy consumption is $0.393J$ which is only 2.3 times higher than the top 10 percentile energy consumption (i.e., $0.17J$). This implies that the energy consumption in the centralized scheme is highly unbalanced compared to the localized scheme and thus, leads to a short network lifetime. Figure 6

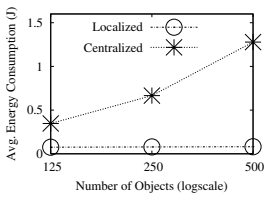

Fig. 7. Avg. Energy Consumption v.s. No. of Objects

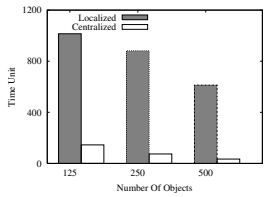

Fig. 8. Network Lifetime

shows the network lifetimes for both schemes. It shows that the localized scheme extends the network lifetime by 33 times over the centralized scheme when $k = 1$ and 13 times when $k = 8$. We also tested different numbers of queries in the experiments. The results, not included here due to space limitation, showed that the localized scheme can support up to 40 queries under the same network lifetime as the centralized scheme.

5.3 Impact of Number of Objects

Figure 7 compares the localized and centralized schemes with different numbers of objects when $k = 8$. The energy consumption is derived at the 34th time unit, i.e., the network lifetime under the centralized scheme with 500 objects. It is seen that the average energy consumption of the localized scheme is much smaller than that of the centralized scheme due to fewer location updates. The performance for the centralized scheme degrades rapidly with increasing number of objects due to the fact that more location updates are sent to the base station with more objects in the network. Figure 8 shows that when the number of objects increases to 500, the localized scheme extends the network lifetime by 18 times over the centralized scheme.

6 Conclusion

In this paper, we have proposed a localized scheme for continuous monitoring of kNN queries in wireless sensor networks. To avoid sending all the object location updates to a centralized repository, we store object locations locally at the detecting sensor nodes and monitor the queries in a localized manner. We setup a monitoring area for each query. Only the updates from the monitoring area are sent to the query initiator. Experimental results show that the localized scheme achieves low energy consumption than the centralized scheme over a wide range of system settings. Meanwhile, the energy consumption is more balanced among the sensor nodes in the localized scheme and therefore, the network lifetime is prolonged.

References

1. J-sim homepage. http://www.j-sim.org.
2. J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proceedings of Sensys'03*.

3. Hoffmann-Wellenhof B, Lichtenegger H, and Collins J. GPS theory and practice. *Springer-Verlag, New York*, 1997.
4. B. Gedik and L. Liu. Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In *Proceedings of EDBT'04*.
5. C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of MobiCom 2004*, 2004.
6. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of Mobicom'00*, 2000.
7. B. Karp and H.T. Kung. GPSR: Greey perimeter stateless routing for wireless networks. In *Proceedings of Mobicom'00*, 2000.
8. S. Madden, M.J. Franklin, J.M. Hellestein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of OSDI'02*, 2002.
9. A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM Workshop on Sensor Networks and Applications*, pages 88–97, 2002.
10. K. Mouratidis, M. Hadjieleftheriou, and D. Papadias. Conceptual partitioning: An efficient method for continous nearest neighbor monitoring. In *Proceedings SIGMOD'05*.
11. K. Mouratidis, D. Papadias, S. Bakiras, and Y. Tao. A threshold-based algorithm for continuous monitoring of k nearest neighbors. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(11):1451–1464, 2005.
12. D. Niculescu and B. Nathi. Ad hoc positioning system. In *Proceedings of INFOCOM'03*, 2003.
13. G. Pottie. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.
14. S. Prabhakar, Y. Xia, D. Kalashnikov, W. Aref, and S. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Transactions on Computers*, 51(10):1124–1140, 2002.
15. S. Ratnasamy, B. Karp, L.Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.
16. M. Wu, J. Xu, X. Tang, and W.-C. Lee. Monitoring top-k query in wireless sensor networks. In *Proceedings of ICDE'06*.
17. X. Xiong, M. Mokbel, and W. Aref. SEA-CNN: Scalable incremental processing of continuous queries in spatio-temporal databases. In *Proceedings of ICDE'05*.
18. J. Xu, X. Tang, and W.-C. Lee. EASE: An energy-efficient in-network storage scheme for object tracking in sensor networks. In *Proceedings of IEEE SECON'05*, 2005.
19. Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell. Processing window queries in wireless sensor networks. In *Proceedings of ICDE'06*, 2006.
20. W.-C. Lee Y. Xu, J. Winter. Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceedings of MDM'04*, 2004.
21. Y. Yao, X. Tang, and E-P. Lim. In-network processing of nearest neighbor queries for wireless sensor networks. In *Proceedings of DASFAA'06*, 2006.
22. X. Yu, K. Pu, and N. Koudas. Monitoring k-nearest neighbor queries over moving objects. In *Proceedings of ICDE'05*.
23. W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *Proceedings of INFOCOM'04*, 2004.