In-Network Processing of Nearest Neighbor Queries for 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. The sensor nodes in the network have the abilities to sense, store, compute and communicate. To enable object tracking applications, spatial queries such as nearest neighbor queries are to be supported in these networks. The queries can be injected by the user at any sensor node. Due to the limited power supply for sensor nodes, energy efficiency is the major concern in query processing. Centralized data storage and query processing schemes do not favor energy efficiency. In this paper, we propose a distributed scheme called DNN for in-network processing of nearest neighbor queries. A cost model is built to analyze the performance of DNN. Experimental results show that DNN outperforms the centralized scheme significantly in terms of energy consumption and network lifetime.

1 Introduction

A sensor network is a distributed *ad-hoc* network comprised of a large number of sensor nodes equipped with capabilities of computing, storing and communicating [1]. The sensor nodes are usually battery operated and are deployed in an unattended manner to gather and process information without human intervention. Therefore, *energy efficiency* is the major concern in accessing the data captured by the sensor network.

A simple centralized method is to send all collected data to the base station for storage [2, 3]. The queries are also forwarded to and processed at a central base station. This approach involves unnecessary communication cost if only a portion of the data are accessed by the user. Moreover, due to message relay, the energy consumed by the sensor nodes closer to the base station is much higher than that by the nodes further from the base station. Unbalanced energy consumption reduces network lifetime [4, 5]. To improve energy efficiency, it is desirable to store the data at the sensor nodes in a distributed manner and apply *in-network processing* techniques to user queries [6, 7]. In this way, only the relevant data are extracted from the network and the communication cost is greatly reduced compared to the centralized scheme. Existing in-network query processing techniques have focused on aggregation and join queries [7, 8, 9, 10]. However, not much work has been done on spatial queries.

Nearest neighbor queries are an important class of spatial queries in object tracking applications [11]. In this paper, we consider in-network processing of nearest neighbor

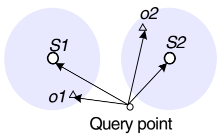

Fig. 1. Nearest Sensor Node vs. Nearest Object

queries. Our objective is to locate the nearest objects to a given query point. For example, consider a sensor network tracking the movement of taxies. The pedestrians carry devices, such as PDAs, to interact with the sensor network. Each PDA accepts from its user queries for nearest taxies and their locations, and injects the queries into the network by sending them to nearby sensor nodes. The data will be extracted from the relevant sensor nodes to respond to the user.

Existing work on nearest neighbor queries has focused on finding the *nearest sensor nodes* to a specified query point [12, 13]. This is different from our objective to locate the *nearest objects* to the query point because the nearest object may not be detected by the nearest sensor node. Figure 1 shows an example where S_1 and S_2 are two sensor nodes that detect objects o_1 and o_2 respectively. S_2 is closer to the query point than S_1. However, the nearest object to the query point is o_1.

In this paper, we propose a distributed scheme called *DNN* for in-network processing of nearest neighbor queries in wireless sensor networks. A grid structure is constructed for in-network storage of the collected data. Query processing in DNN proceeds in four steps: query routing, preliminary search, expanded search and result routing. We build a cost model to analyze the energy consumption of DNN and compare it with the centralized scheme. Experimental results show that DNN achieves significant energy saving over the centralized scheme.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the DNN scheme for in-network processing of nearest neighbor queries. Section 4 develops a cost model. Section 5 describes the experimental setup and discusses the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

R-tree is a widely used indexing structure to support spatial queries in these databases [14]. M. Demirbas and Hakan [12] applied R-tree to locate the nearest sensor nodes in wireless sensor networks. In their approach, the sensor nodes are organized into a distributed R-tree in a bottom-up fashion. Each node keeps pointers to the lower level children and the higher level parent in the tree. Queries may be injected at any sensor node. However, to locate the nearest sensor nodes, the query has to trace back to the root of the tree making it a hotspot in the network. In addition, the tree structure is difficult to maintain in a dynamic environment.

Lee *et al.* [13] proposed an algorithm to locate k nearest sensor nodes in wireless sensor networks. They first locate the nearest sensor node to the query point and a set of perimeter nodes around the query point. A circle centered at the query point is then determined and is further divided into a set of subspaces each containing a perimeter node. The information of each subspace is collected by the perimeter node through a tree structure. After the query is resolved, the perimeter tree is destroyed to avoid the high cost of tree maintenance.

The above work has focused on locating the nearest sensor nodes. The locations of sensor nodes usually do not change over time. Different from [12, 13], we focus on the moving objects tracked by the sensor network. Our objective is to locate the nearest objects to a given query point. Although a number of indexing schemes have been proposed for moving object databases [15, 16], they targeted at centralized databases only and therefore do not apply to in-network processing in wireless sensor networks.

3 In-Network Processing of Nearest Neighbor Queries

In this section, we propose a distributed scheme called DNN for in-network processing of nearest neighbor queries in wireless sensor networks.

3.1 Distributed Data Storage

We consider a sensor network with the sensor nodes spreading over a 2-dimensional space. The sensor nodes are aware of their locations through GPS [17] or other localization algorithms [18]. The sensor nodes can sense the moving objects and collect their location information. Instead of sending all collected data to a central repository, we propose to store them at the sensor nodes in a distributed manner by partitioning the sensor network into a set of grid cells.

As shown in Figure 2, each grid cell has an area of $\alpha \times \alpha$, where α is a system parameter known to all sensor nodes in the network. The grid structure is constructed by designating a reference point (x_r, y_r) as the corner of a grid cell. Then, given any point (x, y) on the plane, the centroid of the grid cell containing (x, y) is given by $\big(x_r + (\lfloor \frac{x-x_r}{\alpha} \rfloor + \frac{1}{2}) \cdot \alpha, y_r + (\lfloor \frac{y-y_r}{\alpha} \rfloor + \frac{1}{2}) \cdot \alpha\big)$. The sensor node closest to the centroid of a grid cell is called a *grid index node* (shown by a solid dot in Figure 2). It is responsible for maintaining the location data of the objects detected in the grid cell. The object locations are periodically sampled by the sensor nodes and reported to the

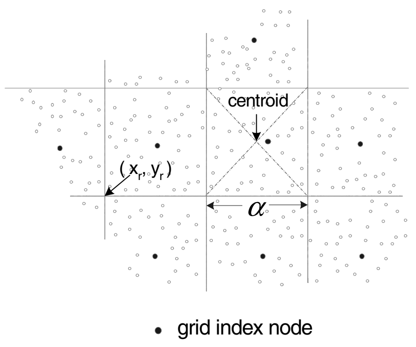

Fig. 2. Grid Structure in Sensor Network

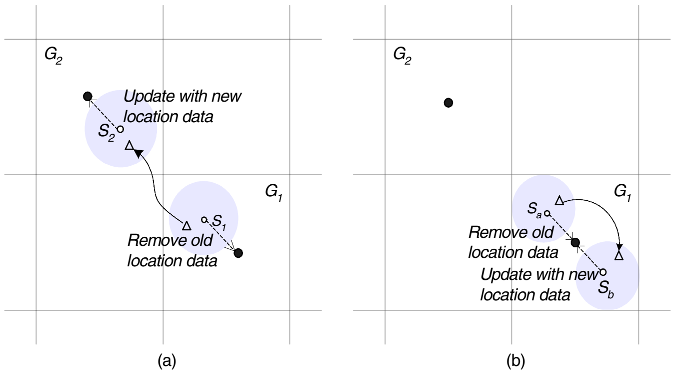

Fig. 3. Update of Object Location

grid index node[1]. The location data can be sent to the corresponding grid index node through GPSR routing[2] [22] by setting the centroid position of the grid cell as the destination.

To save communication cost, at each sampling, the location of an object is reported to the grid index node only if its location has changed since the last sample. We shall call it a *location update*. We assume that the objects being tracked are identifiable. They are electronically tagged or are identified based on the pre-embedded object code table in the sensor nodes [20, 21]. When the object location changes, at most two messages are needed to update the data at the grid index nodes. One message is used to signal the grid index node to remove the old location data and the other message is used to update the grid index node with the new location data. For example, Figure 3(a) shows the case where an object moves from one grid cell G_1 to another cell G_2. The location of the object is detected by sensor nodes S_1 and S_2 at two successive samplings. At the latter sampling, S_1 sends a message to the grid index node in G_1 to remove the old location data. Meanwhile, S_2 sends the new location data to the grid index node in G_2. Figure 3(b) shows another case where an object moves within a grid cell G_1. The location of the object is detected by sensor nodes S_a and S_b at two successive samplings. At the latter sampling, both S_a and S_b send a message to the grid index node in G_1. S_a's message signals the index node to remove the old location data while S_b's message feeds the index node with the new location data. If the object moves within a grid cell and its location is detected by the same sensor node in two successive samplings, only one message is sent from the sensor node to the grid index node for location update.

[1] Although the sensor nodes may work collaboratively to determine the location of an object in their vicinity [19], we assume that for each object, only one sensor node (the sensing leader or cluster head) is responsible for reporting its location at each sampling [20, 21]. For simplicity, the detecting sensor node in the rest of this paper refers to this node.

[2] GPSR is a greedy location-based routing scheme. Given the geographic locations of the source and the destination, GPSR routes the message to the node closest to the destination location. All message routing in this paper refers to GPSR routing.

3.2 Query Processing

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 the search cost, we would like the boundary object to be as close to the query point as possible. Thus, it is intuitive to visit the grid cells based on their distances to the cell G_0 containing the query point. We propose a *circle* approach to determine the visiting order of grid cells. Users issue queries for the locations of the nearest objects to given query points. In this paper, we focus on one-shot queries which complete once the results are returned. The queries can be injected into the sensor network at any sensor node (e.g., depending on the locations of the users). Each query Q is characterized by two locations (x_s, y_s) and (x_0, y_0), where (x_s, y_s) is the location where the query is issued (called the *query source*), and (x_0, y_0) is the location of the *query point*. If a user queries for the nearest object in his proximity, then $(x_s, y_s) = (x_0, y_0)$.

Query processing in DNN proceeds in four steps: *query routing*, *preliminary search*, *expanded search* and *result routing*. When a sensor node receives a query message, it calculates the centroid position of the grid cell G_0 containing the query point (x_0, y_0). In the query routing step, the query message is routed to the grid index node in G_0. The purpose of preliminary search is to find an object (called the *boundary object*) and define the search space. In this step, the grid cells surrounding G_0 (more specifically, the index nodes in these grid cells) are visited by message passing until a grid cell containing at least one object is found. Among the objects detected in that grid cell, the one closest to the query point is selected as the boundary object. A search circle centered at the query point and with a radius of the distance between the query point and the boundary object is defined as the search space. The nearest object to the query point is guaranteed to be located within the circle. The next step is expanded search. In this step, the grid cells within or intersecting with the circle, excluding those visited in the preliminary search, are visited by message passing to locate the nearest object. Finally, the query result is routed back to the user at (x_s, y_s).

Now, we discuss the preliminary search and the expanded search in detail.

Preliminary Search. The search is divided into rounds. In each round i, the unvisited grid cells intersecting with the circle centered at the centroid of G_0 and with a radius of $i \cdot \alpha$ are visited in clockwise order (see Figure 4(a)). This is done by sequentially passing a message from the grid index node of one cell to that of another. The message contains the locations of the query source and query point. Note that given the location of the query point, each grid index node can determine autonomously which grid cell to visit next. Figure 4(b) shows the route of the message in the preliminary search. The preliminary search completes when a grid cell containing at least one object is found.

Expanded Search. On completion of the preliminary search, a search circle centered at the query point and with a radius of the distance between the query point and the boundary object is defined. Let d be the radius of the circle. Intuitively, if the minimum distance between a grid cell and the query point is smaller than d, the grid cell is likely to contain objects less than distance d away from the query point. Therefore, a *search list* for the expanded search is given by all grid cells within or intersecting with the search circle, excluding those visited in the preliminary search. Figure 5 shows an example.

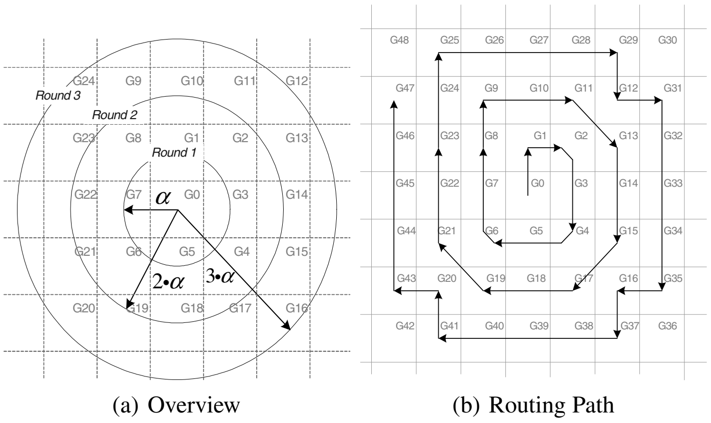

Fig. 4. Preliminary Search

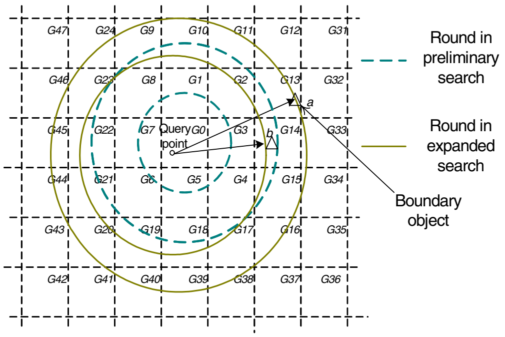

Fig. 5. Expanded Search

Suppose the query point (x_0, y_0) is in G_0 and the boundary object a is found in grid cell G_{13}. The grid index node in G_{13} determines the search circle (shown by the outer solid circle in Figure 5) and derives the set of grid cells within or intersecting with it: $Set_1 = \{G_0 - G_{24}, G_{33}, G_{34}, G_{38} - G_{41}, G_{43} - G_{46}\}$. The grid index node in G_{13} also computes the set of grid cells that have been visited in the preliminary search (based on the rounds shown by the dashed circles in Figure 5): $Set_2 = \{G_0 - G_{11}, G_{13}\}$[3]. Therefore, the search list in the expanded search is given by $Set_3 = Set_1 - Set_2 = \{G_{12}, G_{14} - G_{24}, G_{33}, G_{34}, G_{38} - G_{41}, G_{43} - G_{46}\}$.

The message passed between cells in the expanded search contains the search list and the locations of the boundary object, query source and query point. At each step, the message is routed to the grid cell on the search list that is closest to the cell currently holding the message. When a grid cell G_c receives the message, it first removes itself from the search list. One of the following three cases can occur: (i) no object is detected in G_c; (ii) all the objects detected in G_c are further away from the query point than

[3] The preliminary search completes in the middle of round 2, so the sensor nodes G_{14}– G_{15}, $G_{17} - G_{19}$ and G_{21}– G_{23} have not been visited in the preliminary search.

the boundary object; (iii) at least one object detected in G_c is closer to the query point than the boundary object. In cases (i) and (ii), the search list is not updated and the message is simply routed to the next grid cell on the search list that is closest to G_c. In case (iii), the object detected in G_c that is closest to the query point is selected as the new boundary object by updating the message content. A new search circle is derived accordingly. The search list is then updated by removing all grid cells outside the new search circle. The message is then routed to the next grid cell on the updated search list that is closest to G_c. The expanded search continues until the search list becomes empty. On completion of the expanded search, the message is routed to the query source and the location of boundary object is returned to the user as the query result.

In the example of Figure 5, the message is first routed to cell G_{14} in the expanded search (among the cells on the search list, G_{14} is closest to the grid cell G_{13} last visited in the preliminary search). Object b detected in G_{14} is closer to the query point than the current boundary object a. Thus, the search circle is shrunk and the search list is updated by removing cells G_{16}, G_{38}, G_{39}, G_{40}, G_{41}, G_{43}, G_{44}, G_{45}, G_{46} and G_{24} from the search list because they are outside the new search circle (shown by the inner solid circle in Figure 5). Among the cells on the updated search list, G_{15} is closest to cell G_{14}. So, the message is routed to G_{15} to continue the expanded search.

4 Cost Model and Analysis

In this section, we analyze the cost of the DNN scheme presented in Section 3. It is known that the energy consumption in wireless sensor networks is dominated by the communication cost [1]. We assume a dense network. In this case, the cost of message routing, i.e., the number of hops on the route from a source to a destination, is proportional to the Euclidean distance between the source and the destination. Therefore, we shall analyze the distance travelled by messages. We consider a square sensor field of size $s \times s$. It is divided into grid cells of size $\alpha \times \alpha$. A total number of N sensor nodes are randomly deployed in the network. A total of n objects are tracked by the sensor network.

4.1 Cost Model for DNN

In DNN, query processing and location update both involve communication.[4] For query processing, let C_{query}, C_{pre}, C_{exp} and C_{result} be the expected costs of query routing, preliminary search, expanded search and result routing per query respectively. The expected cost of a location update shall be denoted by C_{update}.

Query Routing and Result Routing. The expected costs of query and result routing are approximated by the distance between the query source and the query point. Assume the query source and query point are both randomly distributed in the network. Then, the expected routing distance is given by the average distance between any two points in the sensor network:

[4] Since this paper focuses on energy-efficient query processing, we do not include the communication overhead in sensing and data fusion. Such overhead is the same for the proposed DNN scheme and the centralized scheme we shall compare.

$$\frac{\int_0^s \int_0^s \int_0^s \int_0^s \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} dx_i dx_j dy_i dy_j}{(s \cdot s)^2}.$$

It follows from the mathematical results [23] that

$$C_{query} = C_{result} = \frac{s}{15}[\sqrt{2} + 2 + 5ln(1 + \sqrt{2})] = 0.5214 \cdot s.$$

Preliminary Search. To derive the cost of preliminary search, we need to know the number of grid cells visited. For simplicity, we assume that the probabilities of detecting objects in different cells are identical and independent. We use p to denote the probability that at least one object is detected in a grid cell. If the number of objects n is much smaller than the number of grid cells $(s \times s)/(\alpha \times \alpha)$, p is approximated by $\frac{n}{(s \times s)/(\alpha \times \alpha)}$. Then, the probability that we need to visit i grid cells in the preliminary search to locate a boundary object is $p(1-p)^{i-1}$. Therefore, the average number of grid cells visited in the preliminary search is given by

$$p + 2p(1-p) + 3p(1-p)^2 + \cdots = \frac{1}{p}.$$

Starting from the grid cell containing the query point, to visit i cells, the message needs to be sent between $i - 1$ pairs of neighboring cells. Since the distance between a pair of neighboring cells is bounded by $\sqrt{2}\alpha$, the cost of preliminary search is bounded by

$$C_{pre} = (\frac{1}{p} - 1) \cdot \sqrt{2}\alpha.$$

Expanded Search. Similar to the preliminary search, we need to derive the number of grid cells visited in the expanded search. As described in Section 3.2, a search circle is derived at the end of preliminary search. If we know the total number of grid cells in the circle and the number of grid cells visited in the preliminary search, we can calculate the upper bound on the number of grid cells to visit in the expanded search.

We start by analyzing the relationship between the radius of a circle and the number of grid cells within or intersecting with the circle. It is intuitive that the number of cells is proportional to the area of the circle. Therefore, we used quadratic regression. The regression result shows that, given the circle radius r (r is a multiple of α), the number of grid cells within or intersecting with the circle is: $ar^2 + br + c$, where $a = \frac{3.1417}{\alpha^2}, b = \frac{4.1178}{\alpha}, c = 2.3241$. Figure 6 shows that the regression result (i.e., the number of grid cells computed by $ar^2 + br + c$) well matches the empirical result (i.e., the actual number of grid cells within or intersecting with the circle).

Let i be the number of grid cells visited in the preliminary search. We assume that the last grid cell visited in the preliminary search is in round x_i, i.e., the circle with radius $r_i = x_i \cdot \alpha$. Note that $a(r_i - \alpha)^2 + b(r_i - \alpha) + c$ indicates the number of grid cells visited in the first $x_i - 1$ rounds, and $ar_i^2 + br_i + c$ is the number of grid cells visited if round x_i completes. It follows that

$$a(r_i - \alpha)^2 + b(r_i - \alpha) + c \leq i \leq ar_i^2 + br_i + c.$$

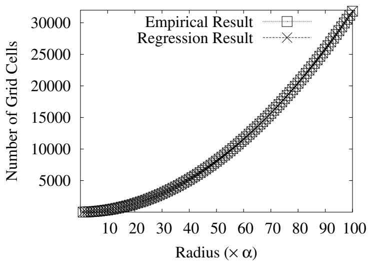

Fig. 6. Number of Grid Cells in a Circle

Therefore,

$$\frac{-b+\sqrt{b^2-4a(c-i)}}{2a} \le r_i \le \frac{-b+\sqrt{b^2-4a(c-i)}}{2a} + \alpha.$$

So, we approximate r_i by

$$r_i = \frac{-b+\sqrt{b^2-4a(c-i)}}{2a} + \frac{1}{2}\alpha.$$

We then derive the average number of grid cells visited in the expanded search as

$$\sum_{i=1}^{\infty} p(1-p)^{i-1} \cdot (ar_i^2 + br_i + c - i), \tag{1}$$

where $p(1-p)^{i-1}$ is the probability that i cells are visited in the preliminary search, and $ar_i^2+br_i+c-i$ is the corresponding number of grid cells to visit in the expanded search. Since the sum (1) converges when i approaches infinity, we can compute it numerically.

Similar to the preliminary search, we use $\sqrt{2}\alpha$ as a bound on the distance between neighboring grid cells. Thus, the cost of expanded search is bounded by

$$C_{exp} = \sum_{i=1}^{\infty} p(1-p)^{i-1} \cdot (ar_i^2 + br_i + c - i) \cdot \sqrt{2}\alpha.$$

Location Update. The communication cost of each location update is determined by the following factors: average distance from any point in a grid cell to the centroid of the grid cell; and the number of messages per location update. Following the mathematical results [23], the average distance is given by

$$\frac{\int_0^{\frac{\alpha}{2}} \int_0^{\frac{\alpha}{2}} \sqrt{x^2+y^2}dxdy}{\frac{\alpha}{2} \cdot \frac{\alpha}{2}} = 0.3825 \cdot \alpha.$$

As discussed in Section 3.1, at most two messages are required for each location update. Therefore, the cost per location update is bounded by

$$C_{update} = 2 \cdot 0.3825 \cdot \alpha = 0.7650 \cdot \alpha.$$

Let q be the rate at which queries are injected into the network. Let u be the total number of location updates per time unit for all objects in the network (it is obvious that u depends on the movement pattern of objects). Then, to summarize, the total communication cost of DNN is given by

$$\begin{aligned} C_{DNN} &= (C_{query} + C_{pre} + C_{exp} + C_{result}) \cdot q + C_{update} \cdot u \\ &= (2 \cdot 0.5214 \cdot s + \frac{1-p}{p} \cdot \sqrt{2}\alpha \\ &\quad + \sum_{i=1}^{\infty} p(1-p)^{i-1} \cdot \big(ar_i^2 + br_i + c - i\big) \cdot \sqrt{2}\alpha) \cdot q + 0.7650 \cdot \alpha \cdot u. \end{aligned}$$

4.2 Cost Model for Centralized Scheme

For comparison purpose, we also derive the communication cost of the centralized scheme in which all sensor nodes send the collected data to the base station and all queries are also forwarded to the base station for processing. We refer to this scheme as *CNN*. We assume the base station is located at the centroid of the network.[5] The cost of CNN consists of three parts: query routing, result routing and location update. The expected costs of query and result routing as well as the cost of per location update are all given by the average distance between any point in the network and the centroid of the network, i.e.,

$$C_{query} = C_{result} = C_{update} = \frac{\int_0^{\frac{s}{2}} \int_0^{\frac{s}{2}} \sqrt{x^2 + y^2} dxdy}{\frac{s}{2} \cdot \frac{s}{2}} = 0.3825 \cdot s.$$

In CNN, all collected data are maintained at the base station. Thus, only one message needs to be sent from the detecting sensor node to the base station at each location update. Therefore, the total communication cost of CNN is given by

$$C_{CNN} = (C_{query} + C_{result}) \cdot q + C_{update} \cdot u = 2 \cdot 0.3825 \cdot s \cdot q + 0.3825 \cdot s \cdot u.$$

5 Performance Evaluation

5.1 Experiment Setup

We conducted a wide range of experiments to evaluate the performance of the proposed DNN scheme and compared it with CNN. Table 1 summarizes the system parameters and their settings. We simulated a sensor network geographically covering a $50000m \times 50000m$ area. The number of sensor nodes deployed in the sensor network was set at 4×10^6, implying that on average, there is one sensor node in each $25m \times 25m$ square. The default size of a grid cell was set at $125m \times 125m$. The default number of objects being tracked was set at 800. The object were initially placed in the network at random. Their movement followed the random walk model. Specifically, time was divided into

[5] We set the base station at the centroid of the network to favor the centralized scheme.

Table 1. System Parameters and Settings

Parameters	Description	Default Value	Range
N	Number of sensor nodes	4×10^6	—
R	Communication range	$40m$	—
$s \times s$	Size of sensor network	$50000m \times 50000m$	—
$\alpha \times \alpha$	Size of grid cell	$125m \times 125m$	$[125m \times 125m, 3125m \times 3125m]$
n	Number of objects tracked	800	[160, 1600]
r	Sampling rate of sensor nodes	1 per time unit	—
v	Object moving velocity	$50m$ per time unit	—
P_{move}	Probability of object moving in each time interval	0.5	[0.1, 0.9]
q	Query rate	50 per time unit	[10, 100]

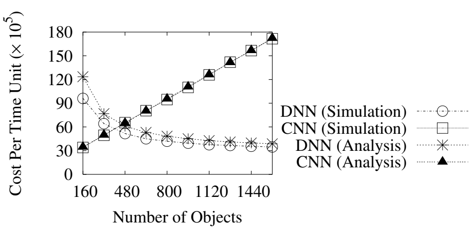

Fig. 7. Cost vs. Number of Objects

intervals. At the beginning of each interval, the object decided whether to *move* or *pause* according to the probabilities P_{move} and P_{pause} ($P_{move} + P_{pause} = 1$). If it decided to move, the move direction was randomly selected between 0 and 2π. The moving speed was set at $50m$ per time unit. The default length of the interval was set at 1 time unit. The default value for P_{move} was set at 0.5. The object locations were sampled by the sensor network once every time unit. At each sampling, the sensor node closest to an object was assumed to report the object location to the grid index node (in DNN) or the base station (in CNN). The default query rate was set at 50 per time unit. Both the query source and query point were randomly distributed in the network. The default communication range of each sensor node is set at $40m$.

5.2 Impact of Number of Objects

Figure 7 shows the simulation and analytical results of the Euclidean distance travelled by messages[6] as a function of number of objects. As seen from Figure 7(a), the analytical and simulation results match well. The analytical cost of DNN is slightly higher than the simulation result. This is because the DNN cost analyzed in Section 4 is an upper bound. As shall be explained soon, the cost of CNN increases with the number of objects, while that of DNN decreases with increasing number of objects. DNN outperforms CNN over a wide range of object numbers.

[6] We measured the total number of location updates per time unit in the simulation experiments and plugged it into the analytical model presented in Section 4.

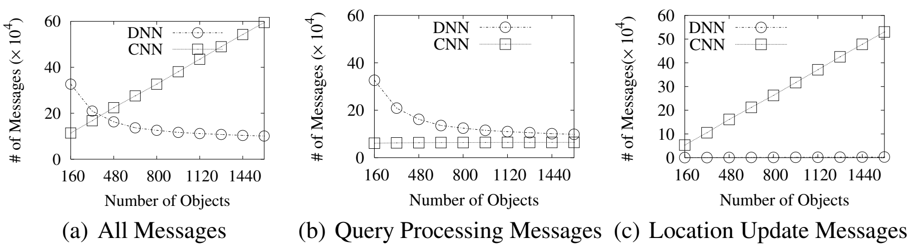

Fig. 8. Number of Messages vs. Number of Objects

Figure 8(a) shows the total number of messages sent by the sensor nodes in the simulation experiments. It is seen that the curves have the same trend as those in Figure 7. This verifies that the cost defined using Euclidean distance (Section 4) is a good measure of message complexity. Figures 8(b) and 8(c) show the breakdown of query processing and location update messages. As shown in Figure 8(b), when the number of objects increases, the number of query processing messages in CNN remains unchanged. This is because in CNN, query processing consists of query routing and result routing only, the cost of which are independent of the number of objects. For DNN, the number of query processing messages reduces with increasing number of objects because the boundary object is located closer to the query point. This not only cuts down the number of grid cells visited in the preliminary search but also reduces the size of the search circle and hence the number of cells to visit in the expanded search. Figure 8(c) shows that the number of location update messages in DNN is considerably lower than that of CNN. It also grows much slower compared to that of CNN when the number of objects increases. With large number of objects, the overall message complexity of CNN is dominated by the location update messages and is much higher than that of DNN.

Figure 9 shows the distribution of the number of messages sent by the sensor nodes for DNN and CNN when the object number is 800. A point (x, y) on the curve means that a fraction x of all sensor nodes send more than y messages each. As shown in Figure 9, the workload distribution among the sensor nodes is highly unbalanced in CNN. The top 0.1% of the nodes send substantially high numbers of messages than the remaining nodes. On the other hand, the workload hence energy consumption is much more balanced among the sensor nodes in DNN. The numbers of messages sent by the top nodes are more than two orders of magnitude lower than those in CNN. If we define the network lifetime as the time duration before the first sensor node runs out

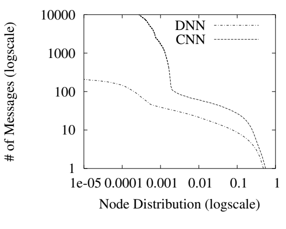

Fig. 9. Workload for Sensor Nodes

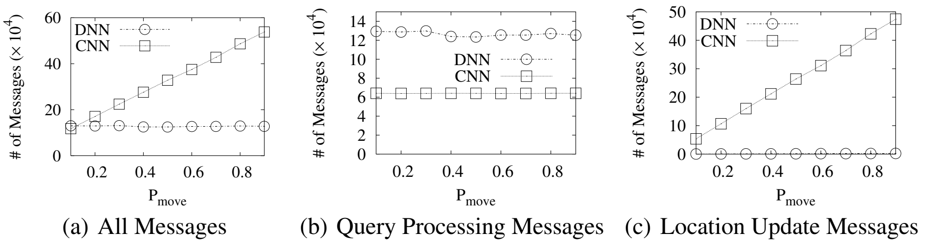

Fig. 10. Number of Messages vs. P_{move}

of energy [24], DNN would prolong the network lifetime by a factor of more than 100 over CNN.

5.3 Impact of Object Movement

Figure 10 shows the performance results for different P_{move} values. It is intuitive that the objects move faster and hence incur more location updates at larger P_{move} values. Since the number of location update messages in DNN is much lower than the number of query processing messages (recall Figures 8(b) and (c)), the total number of messages in DNN is not significantly affected by the increase in P_{move}. The overall message complexity of CNN, on the other hand, substantially increases with P_{move}. This is because the total number of messages in CNN is dominated by that of location update messages. As shown in Figure 10(a), DNN considerably outperforms CNN over a wide range of P_{move} values.

5.4 Impact of Query Rate

Figure 11 shows the performance results for different query rates. The results indicate that the overall message complexity increases with query rate for both DNN and CNN. As shown in Figure 11(b), the number of location update messages is independent of the query rate for both schemes. Figure 11(c) shows that the number of query processing messages increases with query rate, leading to an increase in the overall message complexity. DNN outperforms CNN over a wide range of query rates. In general, the improvement of DNN over CNN is smaller for larger query rate.

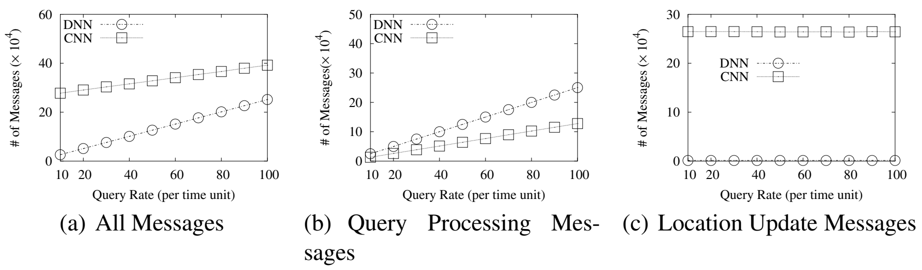

Fig. 11. Number of Messages vs. Query Rate

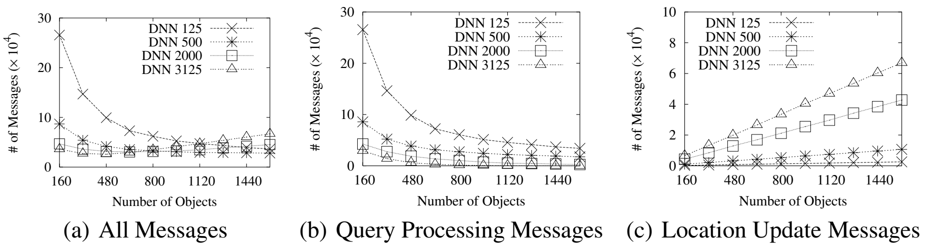

Fig. 12. Number of Messages vs. Grid Cell Size

5.5 Impact of Grid Cell Size

We also investigate the impact of grid cell size. Figure 12 shows the message complexity for different α values $125m$, $500m$, $2000m$ and $3125m$ when the number of objects increases from 160 to 1600. In general, the number of query processing messages in DNN decreases with increasing grid cell size (see Figure 12(b)). On the other hand, the number of location update messages increases with grid cell size (see Figure 12(c)). When the number of objects is small, the location update messages take up a negligible portion of the total number of messages. Therefore, the overall message complexity of DNN decreases with increasing grid cell size. When the number of objects is large, the location update messages take up a larger portion of the total number of messages. As a result, the overall message complexity of DNN may increase with grid cell size beyond certain value. For example, when there are more than 800 objects being tracked, the cost for $\alpha = 3125$ is higher than that for $\alpha = 2000$.

6 Conclusion

In this paper, we have proposed a distributed scheme called DNN for in-network processing of nearest neighbor queries in wireless sensor networks. To avoid sending data to a central repository, a grid structure is constructed for in-network storage of the collected data. By localizing the location updates, DNN eliminates hotspots in the system. Query processing in DNN proceeds in four steps: query routing, preliminary search, expanded search and result routing. Experimental results show that DNN can significantly reduce and balance network-wide energy consumption compared to the centralized scheme.

References

1. G. Pottie. Wireless integrated network sensors. *Communications of the ACM* **43** (2000), pages 51–58.
2. P. Bonnet, J. Gehrke, P. Seshadri. Towards sensor database systems. In *Proceedings of IEEE MDM'01*, Hong Kong, China (2001), pages 3–14
3. R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin. Shenker, S.: The sensor network as a database. Technical Report 02-771, Computer Science Department, University of Southern California (2002).

4. X. Tang, J. Xu. Extending network lifetime for precision-constrained data aggregation in wireless sensor networks. Accepted to appear in *Proceedings of IEEE INFOCOM'06*, Barcelona, Spain (2006).
5. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. A survey on sensor networks. In *IEEE Communication Magazine* **40** (2002), pages 102–114.
6. c. Intanagonwiwat, R. Govindan, D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual ACM/IEEE Mobicom'00*, Boston, MA, USA (2000).
7. S. Madden, M. Franklin, J. Hellerstein, W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of ACM SIGMOD'03*, San Diego, CA, USA (2003).
8. N. Chowdhary, H. Gupta. Communication-efficient implementation of join in sensor networks. In *Proceedings of DASFAA'05*, Beijin, China (2005).
9. M. Sharaf, J. Beaver, A. Labrinidis, P. Chrysanthis. TiNA: A scheme for temporal coherency-aware in-network aggregation. In *Proceedings of the 3rd International ACM Workshop on Data Engineering for Wireless and Mobile Access*, San Diego, CA, USA (2003), pages 69–76.
10. Y. Yao, J. Gehrke. Query processing for sensor networks. In *Proceedings of the 1st Conference on Innovative Data System Research*, Asilomar, CA (2003).
11. N. Roussopoulos, S. Kelley, F. Vincent. Nearest neighbor queries. In *Proceedings of ACM SIGMOD'95*, San Jose, CA, USA (1995).
12. M. Demirbas, H. Ferhatosmanoglus. Peer-to-peer spatial queries in sensor networks. In *Proceedings of the 3rd IEEE International Conference on Peer-to-Peer Computing*, Linkoping, Sweden (2003).
13. J. Winter, W.-C. Lee. KPT: A dynamic KNN query processing algorithm for location-aware sensor networks. In *The 1st VLDB Workshop DMSN'04*, Toronto, Canada (2004).
14. A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of SIGMOD'84*, Boston, Massachusetts (1984).
15. S. Saltenis, C. Jensen, S. Leutenegger, M. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the ACM SIGMOD'01*, Santa Barbara, CA, USA (2001).
16. Y. Tao, D. Papadias, A. Shen. Continuous nearest neighbor search. In *Proceedings of VLDB'02*, Hong Kong, China (2002).
17. B.H. Wellenhof, H. Lichtenegger and J. Collins. Gps theory and practice. 2nd Ed, Springer-Verlag, New York.
18. D. Niculescu, B. Nathi. Ad hoc positioning system. In *Proceedings of INFOCOM'03*, San Francisco, CA (2003).
19. D. Li, K. Wong, Y. Hu, A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine* **19** (2002).
20. J. Xu, X. Tang, W.-C. Lee. Ease: An energy-efficient in-network storage scheme for object tracking in sensor networks. In *Proceedings of IEEE SECON'05*, Santa Clara, CA, USA (2005).
21. Y. Xu, J. Winter, W.-C. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceedings of IEEE MDM'04*, Berkeley, CA, USA (2004).
22. B. Karp, H. Kung. GPSR: Greey perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual ACM/IEEE Mobicom'00*, Boston, MA, USA (2000).
23. E. Weisstein. Hypercube line picking. MathWorld–http://mathworld.wolfram.com/HypercubeLinePicking.html.
24. C. Buragohain, D. Agrawal, S. Suri. Power aware routing for sensor databases. In *Proceedings of IEEE INFOCOM'05*.