FP-CPNNQ: A Filter-Based Protocol for Continuous Probabilistic Nearest Neighbor Query

Yinuo Zhang[1], Anand Panangadan[2], Viktor K. Prasanna[2]

[1] Department of Computer Science, University of Southern California
[2] Ming Hsieh Department of Electrical Engineering, University of Southern California
{yinuozha,anandvp,prasanna}@usc.edu

Abstract. An increasing number of applications in environmental monitoring and location-based services make use of large-scale distributed sensing provided by wireless sensor networks. In such applications, a large number of sensor devices are deployed to collect useful information such as temperature readings and vehicle positions. However, these distributed sensors usually have limited computational and communication power and thus the amount of sensor queries should be reduced to conserve system resources. At the same time, data captured by such sensors is inherently imprecise due to sensor limitations. We propose an efficient probabilistic filter-based protocol for answering continuous nearest neighbor queries over uncertain sensor data. Experimental evaluation on real-world temperature sensing data and synthetic location data showed a significant reduction in the number of update messages.

1 Introduction

A *range query* is a test of whether a variable has its value within a specified range. When the variable is to be monitored so that a result is returned whenever its value enters the specified range, the range query is typically registered at a server and the process is called a *continuous range query*. Continuous queries, in particular continuous range queries, has attracted significant research interest with the development of wireless sensor networks and moving object databases. For example, a query to an environment monitoring sensor network could be to continuously return the identity of all temperature sensors with their readings above a certain threshold (if the temperature reading is above this limit, it could indicate a fire at the sensor location). A range query is a special case of a *non-aggregate* query as the answer only depends on the value of the object being queried (for instance, a sensor measurement or location). On the other hand, an *aggregate* query is one where accessing a single object does not provide enough information to answer the query. For example, whether vehicle v is in a specific region only depends on its own position (non-aggregate query), while whether v is the nearest vehicle to building b also depends on other vehicles (aggregate query). Such *continuous nearest neighbor queries* are important in

location-based services. For example, a query that can continuously report the nearest ambulance to an accident site can enable rapid response to accidents.

An additional complication in practical sensor network deployments is the inherent uncertainty in the measurement process due to reasons such as sensor inaccuracy, discrete sampling intervals, and network latency. In previous work [31], we adopted the *attribute uncertainty* model to formalize the querying of uncertain data sources. The attribute uncertainty model assumes that the true value of the object being queried is within a closed region with a non-zero probability density function (PDF) for the value of interest. This region of uncertainty is an interval for the one-dimensional case, while it is a closed 2D region (e.g., circle or rectangle) for the two-dimensional case. The PDF can take on any distribution such as uniform distribution or Gaussian distribution. Figure 1 illustrates an example of attribute uncertainty for one-dimensional and two-dimensional data.

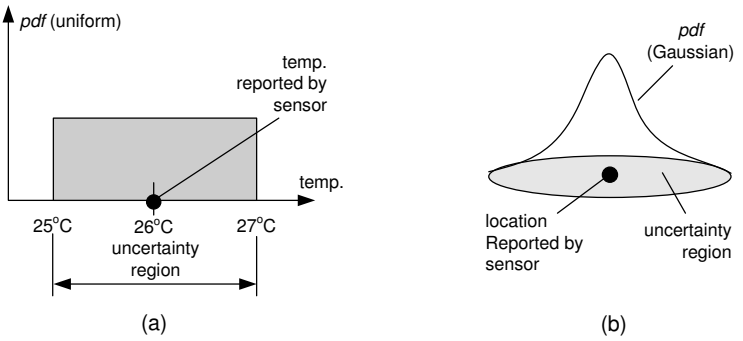

Fig. 1. Uncertainty of (a) temperature and (b) location

Queries can be issued over such uncertain data sources. Incorporating attribute uncertainty in continuous query processing results in *continuous probabilistic queries* (CPQ). In this context, "probabilistic" refers to a threshold condition defined as a probability bound that is provided with the query. For example, instead of a query requesting the identity of all temperature sensors having their readings within a specific range, a *probabilistic* query requests the identity of all sensors that are likely to have readings within that range with a probability higher than some pre-defined threshold.

As previously described, continuous queries over uncertain data can be object-independent as in the case of *continuous probabilistic range queries* (CPRQ) or be of an aggregate type, as in *continuous probabilistic nearest neighbor queries* (CPNNQ).

In [31], we comprehensively investigated object-independent queries with emphasis on range queries. We proposed a probabilistic filter protocol that reduces both communication and computational cost during query execution. Efficient concurrent query execution is also enabled with a multiple query protocol. In

this paper, we extend the filter-based approach to the case of object-dependent queries over uncertain data. Such aggregate queries are more difficult to answer compared to object-independent queries since it requires accessing more information. We evaluate our approach for the special case of CPNNQ. Our main contributions in this paper are:

- Formalization of continuous probabilistic nearest-neighbor query (CPNNQ)
- A filter-based protocol to efficiently answer CPNNQs
- Evaluation of the proposed approach with comprehensive experiments on real sensor datasets

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes the proposed data system and query model. A novel probabilistic filter protocol for CPNNQ is proposed in Section 4. In Section 5, we describe our experimental setup for evaluation and discuss the results. We conclude this paper in Section 6.

2 Related Work

In this section, we first summarize current research in efficient continuous query processing followed by progress in probabilistic query execution.

2.1 Continuous Query Processing

Efficient continuous query execution has been widely studied in the database community. Most of this work focuses on reducing the update frequency and computation load during query execution. In wireless sensor networks (WSNs), in-network processing is a common strategy to provide energy-efficient query execution in terms of communication cost [13,1]. Research has resulted in efficient protocols for a single type of query execution including aggregate query [18], top-k query [29] and spatio-temporal query [8]). In [15], a general framework was proposed to find the maximum lifetime for continuous innetwork evaluation of expression trees and can efficiently execute continuous queries in WSNs. However, in-network processing demands a large query processing capability at all sensors. In order to alleviate this requirement, [24] proposed a technique which does not require that a sensor be able to resolve a query and also supports multiple types of queries. In order to reduce transmission power, [2] also presented an optimized query routing tree which can provide a path to transmit query results to the querying node.

Most of the above works focus on the resolution of a single query. Prabhakar et al. introduced an efficient indexing framework to handle query arrival and removal for multiple query execution [23]. Xiong et al. [30] proposed an incremental algorithm for reducing query re-evaluation cost by sharing execution effort among concurrently-executing queries. Muller et al. proposed a network query approach which combines multiple requests [20]. The results for a specific

user are then extracted from the results for the corresponding network query. Li et al. developed an algorithm for evaluating multiple queries, which exploit the sharing of data movement among different queries [16]. [17] provided a scalable energy-efficient multi-query processing framework which enables sharing information among different queries.

Another technique for reducing system load is *stream filter* [22,7,29,6,10], in which some query answering tasks are deployed to remote streaming sources (e.g., sensors and mobile devices). Each remote source is associated with *filter constraints* derived from a given continuous query. These constraints are used to decide whether an object needs to report its newest value to the server. Since the filter prevents all values from being sent to the server, a substantial amount of communication effort can be saved. However, data uncertainty is wide-spread in real-world applications and this issue is not addressed in existing work. In this paper, we investigate the problem of efficiently executing attribute-uncertain queries. Specifically, we develop *probabilistic filters* for continuous probabilistic nearest-neighbor queries which utilize the uncertainty information associated with sensor measurement.

2.2 Uncertainty Management in Query Execution

Chen et al. [4] studies the problem of *updating* answers for continuous probabilistic nearest neighbor queries in the server. They developed an efficient algorithm to update the answers without re-evaluating the whole query. [28,27] investigated the problem of efficiently executing continuous nearest neighbor (NN) queries for uncertain moving objects trajectories. [21] addressed probabilistic nearest neighbor queries in uncertain trajectories databases using a Markov chain model. Note that these works only handle continuous queries on the server and do not use filters to reduce communication and energy costs. Farrell et al. [11,12] proposed the notion of spatial and temporal tolerance, and examined the use of these semantics to support energy-efficient sensing of location data. The uncertainty in location value is modeled by a uniform uncertainty region; the possibility of a non-uniform PDF representing the uncertainty within the region is not considered. Moreover, the results of the queries studied in those works are not probabilistic. In this paper, we consider attribute-uncertain data and probabilistic queries.

In our preliminary work [32], we developed a filter protocol for single continuous non-aggregate query execution. In [31], we examined how to handle multiple queries efficiently. We also proposed slack filters to approximate filter regions that are hard to represent. We used range query, a typical aggregate query, as a case study in these works. In [14], we investigated efficient continuous aggregate query execution over uncertain data. However, this work mainly focus on *possible* instead of probabilistic queries (this is the special case where the probability threshold specified in the query condition is either 0 or 1). In this paper, we explore the problem of probabilistic aggregate query execution. Specifically, we will develop a filter-based protocol for continuous nearest neighbor queries over uncertain data.

3 System Architecture

In this section, we present the system model for query execution and the probabilistic query definition.

3.1 System Model

We adopt the system model defined in [31]. Figure 2 shows the system framework which has following components.

- **Uncertainty Database** stores an error model (e.g., attribute uncertainty model with region and distribution) for each type of sensor and the most recent value reported by each sensor.
- **Query Manager** receives query requests from users and evaluates them based on the data in the *uncertainty database* (e.g., [5]).
- **Filter Manager** derives *filter constraints*: the query information and data uncertainty is sent to the sensors which use this information to decide if they should report any updated value. This step reduces the energy and network bandwidth consumption.
- Each **sensor** has a *data collector* which retrieves data values (e.g., temperature or position coordinates) from external environments and a set of *filter constraints*, which are boolean expressions for determining whether the value obtained from the data collector is to be sent to the server.

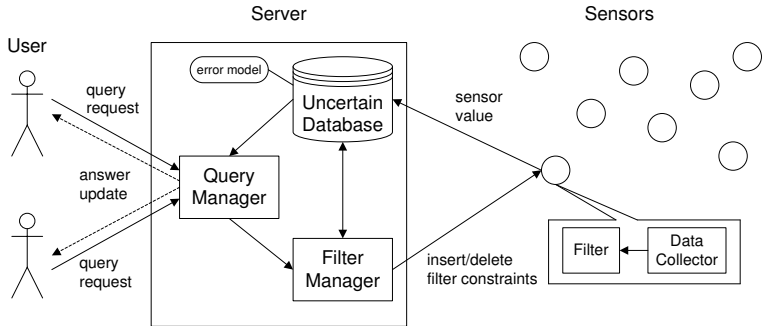

Fig. 2. System Architecture

Table 1 summarizes the symbols used throughout this paper. We describe a one-dimensional data uncertainty model (e.g., Figure 1(a)) to illustrate our techniques. However, the techniques can be directly applied to the multi-dimensional case since we can project the distance between the sensed value of a sensor and the query position to one dimension.

Table 1. Symbols used in the paper

Symbol	Description
o_i	ID of the i-th sensor, where $1 \leq i \leq n$
$v_i(t)$	Sensed value of o_i at time t
P	Probability threshold for q
$p_i(t)$	Qualification probability of o_i at time t for q
$b_i = [l_i, u_i]$	1D probabilistic filter of o_i for q_j

3.2 Continuous Probabilistic Queries

Let $o_1, \ldots, o_n$ be the IDs of n sensing devices monitored by the system. A Continuous Probabilistic Query (CPQ) [31] is defined as:

Definition 1. *Given a time interval $[t_1, t_2]$, a real value $P \in (0, 1]$, a* **CPQ** *q returns a set of IDs $\{o_i | p_i(t) \geq P\}$ at every time instant t, where $t \in [t_1, t_2]$, and $p_i(t)$ is the probability that the value of o_i satisfies query q at time t.*

We also call $[t_1, t_2]$ the *lifetime* of q. In this paper, we focus on a special case of CPQ called *continuous probabilistic nearest neighbor query* (**CPNNQ**), as defined below:

Definition 2. *Given a time interval $[t_1, t_2]$, a real value $P \in (0, 1]$, a* **CPNNQ** *q returns a set of IDs $\{o_i | p_i^{NN}(t) \geq P\}$ at every time instant t, where $t \in [t_1, t_2]$, and $p_i^{NN}(t)$ is the probability that o_i is the nearest neighbor of q at time t.*

An example of such a query is: "During the time interval $[1PM, 2PM]$, what are the IDs of sensors, whose temperature values are closest to (nearest neighbor of) 13^oC with probability $p > 0.2$, at each point of time?" Another example in 2D case is: "During the next one hour, what are the IDs of vehicles, whose probabilities of being the nearest neighbor of Staples Center are more than $P = 0.3$, at each point of time?" Notice that the answer can be changed whenever a new value is reported.

At any time t, the qualification probability of a sensor o_i for a query q can be computed by performing the following operation:

$$p_i^{NN}(t) = Pr(\forall o_j \in O, o_j \neq o_i, |o_i(t) - q| \leq |o_j(t) - q|) \tag{1}$$

where $|q - o(t)|$ is the distance between query q and sensor o's value at time t. Note that $o(t)$ represents sensor o's most recently updated value instead of sensed value at time t. However, we know that the value of sensor o is modeled using attribute uncertainty. So Equation 1 can be rewritten as:

$$p_i^{NN}(t) = \int_{n_i}^{f} Pr(|o_i(t) - q| = r) Pr(|o_j(t) - q| > r) dr \tag{2}$$

$$= \int_{n_i}^{f} pdf_i(r) \prod_{k=1 \wedge k \neq i}^{n} (1 - cdf_k(r)) dr \tag{3}$$

where r is a variable denoting the distance to q, f is the distance between the nearest far-point among all sensor values to q, n_i is the distance between the nearest point of sensor o_i's value to q, pdf_i and cdf_i are the probability density function and the cumulative density function of o_i's value respectively. For example,

$$cdf_k(r) = \int_{n_k}^{r} pdf_k(r)dr \tag{4}$$

Basic CPNNQ Execution. A naive approach for answering a CPNNQ is to assume that each sensor's filter has no constraints. When a sensor's value is updated at time t', its new value is immediately sent to the server, and the qualification probabilities of all sensors are re-evaluated. Then, after all $p_i^{NN}(t')$ have been computed, the IDs of devices whose qualification probabilities are not smaller than P are returned to the user. The query answer is recomputed during t_1 and t_2, whenever a new value is received by the server.

However, this approach is inefficient because:

- Every sensor has to report its sensed value to the server periodically, which wastes a lot of energy and network bandwidth;
- Whenever an update is received, the server has to compute the qualification probability of each sensor with Equation 1, which can be slow.

In the next section, we will present our efficient probabilistic filter-based solution for handling continuous nearest neighbor queries.

4 The Probabilistic Filter Protocol

In this section, we present a filter-based protocol for answering continuous nearest neighbor query.

4.1 Protocol Design

Let us discuss the protocol for server side and sensor side separately.

In the server side (Algorithm 1), once a query q is registered, it comes to the initialization phase. The server requests the latest value (i.e., temperature reading) from each sensor. Based on the values and uncertainty model, the server computes a probabilistic filter for each sensor and deploy on the sensor side. How the filter is derived will be elaborated in Section 4.2. The server initializes the answer set using the uncertain database. In the maintenance phase, once the server receives an update from a sensor, it requests the latest value from each sensor. The probabilistic filters are recomputed in the server side and sent back to each sensor. Based on the latest values in the uncertain database, the server refreshes the answer set.

In the sensor side, it is quite similar to [31]. The only task is to check whether the sensed value violates its filter constraint or not. In order to do this, each sensor needs to continuously sense its value and compare with the filter constraint.

```
1  Initialization:
2  Request data from sensors o_1, ..., o_n;
3  for each sensor o_i do
4      UpdateDB(o_i);
5      Compute filter region b_i;
6      Send(addFilterConstraint, b_i, o_i);
7  Initialize the answer set;
8  Maintenance:
9  while t_1 ≤ currentTime ≤ t_2 do
10     Wait for update from o_i;
11     Request data from sensors o_1, ..., o_n;
12     for each sensor o_i do
13         UpdateDB(o_i);
14         Compute filter region b_i;
15         Send(addFilterConstraint, b_i, o_i);
16     Update the answer set;
17 for each sensor o_i do
18     Send(deleteFilterConstraint, o_i);
```

Algorithm 1: Probabilistic filter protocol for CPNN query(server side).

The filter constraint is in the form of an interval in one dimensional case and an annulus in two dimensional case. If the sensed value is within that interval or annulus, the sensor does not need to send an update to the server. Otherwise, update is sent. Intuitively, this protocol can reduce the number of communication messages sent from sensors to server, but still provides a correct query result set.

4.2 Filter Derivation

In this section, we will present how the probabilistic filter is derived. We use one dimensional filter constraint as a case study.

For the filter constraint of o_i, it consists of two boundaries $b_i = [l_i, u_i]$. For convenience, l_i denotes the boundary near to the query point, while u_i denotes the boundary far from the query point.

Preprocessing Phase. Once the server receives the latest values and uncertainty model from all sensors, it computes the qualification probability $p_i^{NN}(t_0)$ (p_i for short in the following discussions) for each sensor o_i. Based on the qualification probabilities, the server classifies the sensors into three sets S, L (answer set) and Z.

$$S \leftarrow \{o_i | 0 < p_i < P\} \tag{5}$$

$$L \leftarrow \{o_i | p_i \geq P\} \tag{6}$$

$$Z \leftarrow \{o_i | p_i = 0\} \tag{7}$$

We now describe how to derive b_i for the sensors in each set.

Deriving u_i for S. Algorithm 2 illustrates the derivation of far boundary for sensors in set S.

```
1 p_diff = min_{o_i ∈ S}(P − p_i);
2 for each sensor o_i ∈ S do
3 |   s_i = max(p_i − p_diff / (|S|−1), 0);
4 |   u_i = z_i(s_i);
```

Algorithm 2: Deriving u_i for S.

In Algorithm 2, $z_i(x)$ is a function defined as: given that all other sensor values do not change, the qualification probability of o_i will be exactly p, when the distance from o_i's sensed value to the query point is $z_i(p)$, where $0 \leq p \leq 1$. Notice that sometimes $z_i(p)$ may not exist when p is large (i.e. if another sensor value's uncertainty region overlaps with q, o_i cannot have its qualification probability to be 1).

The intuition behind Algorithm 2 is as follows: if o_i's sensed value moves far away from query q, p_i becomes smaller. This may increase the qualification probability of other sensors in S or L since the sum of p is equal to 1. The filter boundary u_i ensures that the qualification probability of any sensor in S will not increase to a value above P. It means that no sensor in S changes its status from non-answer to answer if the filter constraint is not violated.

Deriving l_i for L. Algorithm 3 illustrates the derivation of near boundary for sensors in set L.

```
1 p_diff = min_{o_i ∈ L}(p_i − P);
2 for each sensor o_i ∈ L do
3 |   s_i = p_diff / (|L|−1);
4 |   p'_i = {p_i | c_i = 0};
5 |   y_i = min(p_i + s_i, p'_i);
6 |   l_i = z_i(y_i);
```

Algorithm 3: Deriving l_i for L.

Similar to Algorithm 2, the filter boundary derived from Algorithm 3 guarantees that if o_i's value moves close to query q, all other sensors in L have no chance to switch their status from answer to non-answer. In other words, the filter constraint limits the increase on p_i so that others' qualification probability cannot decrease to a value below P.

Deriving l_i for S. The next two cases (l_i for S and u_i for L) are more complicated than the previous two. Let us imagine, if the value of a sensor $o_i \in S$ moves towards query q, it may affect the status of the sensors in L since the increase in p_i causes the decrease in the qualification probability for sensors in L. It is possible that some L sensors have their qualification probability less than P. But this never happens in the first two cases.

For convenience, we first sort all sensors. We have sensors in L as o_1,...o_h in descending order of p, and sensors in S as o_{h+1},...,o_m in descending order of p. c_i denotes the distance from the latest sensed value v_i to q. In order to derive l_i for S, we define two functions $Z_i(w, C[1...m])$ and $CalQP_h(C[1...m])$ as follows,

Given a sensor o_i, the distance from its sensed value to the query point is $Z_i(w, C[1...m])$ where $w = p_i \in [0, 1]$ and $C[k] = |v_k - q|$.

Given a sensor o_h, its qualification probability is $CalQP_h(C[1...m])$ where $C[k] = |v_k - q|$.

Let us design the algorithm by considering the worst case. Suppose we allow p_i increase by at most δ_i. To ensure the correctness, δ_i must satisfy following conditions,

$$p'_h - \sum_{i=h+1}^{m} \delta_i \geq P \tag{8}$$

where $p'_h = CalQP_h(C[1...m])$ and $C[k] = l_i$, $k = 1...h-1$ and $C[k] = c_k$, $k = h...m$.

$$p_i + \sum_{k=h+1, k\neq i}^{m} (p'_k - p_k) + \delta_i < P \tag{9}$$

where $p'_k = CalQP_k(C[1...m])$ and $C[t] = c_t$, $t = 1...h$, i and $C[t] = u_t$, $t = h+1...m$, $t \neq i$.

Equation 8 guarantees that if all sensor values move to their inner filter boundary except o_h, then o_h's p_h is still larger than or equal to P. Equation 9 guarantees that if all S sensors values move to their outer filter boundary except o_i and o_i moves to its inner boundary, then o_i's p_i is still less than P.

As a result, the increase for p_i should satisfy

$$\delta_i \leq \frac{p'_h - P}{|S|} \tag{10}$$

$$\delta_i < P - P_i - \sum_{k=h+1, k\neq i}^{m} (p'_k - p_k) \tag{11}$$

If $P - P_i - \sum_{k=h+1, k\neq i}^{m} (p'_k - p_k) \geq \frac{p'_h - P}{|S|}$, the maximum allowed qualification probability $p_i^{max} = p_i + \delta_i$, where $\delta_i = \frac{p'_h - P}{|S|}$, and $l_i = Z_i(p_i^{max}, C[1...m])$, where $C[k] = l_k$ and $k = 1...h-1$ and $C[k] = c_k$, $k = h...m$.

If $P - P_i - \sum_{k=h+1,k\neq i}^{m}(p'_k - p_k) < \frac{p'_h - P}{|S|}$, the maximum allowed qualification probability $p_i^{max} = p_i + \delta_i$, where $\delta_i = P - P_i - \sum_{k=h+1,k\neq i}^{m}(p'_k - p_k)$, and $l_i = Z_i(p_i^{max}, C[1...m])$, where $C[k] = c_k$ and $k = 1...h, i$ and $C[k] = u_k$, $k = h + 1...m, k \neq i$.

According to Equation 10 and 11, the qualification probability of o_i can reach $p_i^{max} = p_i + \delta_i$, but no larger than p_i^{max}. Then the filter boundary can be safely set as $l_i = Z_i(p_i^{max}, C[1...m])$. The derivation is formalized in Algorithm 4.

1 Compute $p'_h = CalQP_h(C[1...m])$ and $C[k] = l_i$, $k = 1...h-1$ and $C[k] = c_k$, $k = h...m$;
2 Compute $p'_k = CalQP_k(C[1...m])$ and $C[t] = c_t$, $t = 1...h$, i and $C[t] = u_t$, $t = h+1...m$, $t \neq i$;
3 **for** *each sensor* $o_i \in S$ **do**
4 $\quad p_i^{max} = p_i + min(\frac{p'_h - P}{|S|}, P - P_i - \sum_{k=h+1,k\neq i}^{m}(p'_k - p_k))$;
5 $\quad l_i = Z_i(p_i^{max}, C[1...m])$;

Algorithm 4: Deriving l_i for S.

Deriving u_i for L. We omit the detailed derivation here since it is similar to the previous case (l_i for S). Algorithm 5 illustrates the derivation steps.

1 Compute $p'_{h+1} = CalQP_{h+1}(C[1...m])$ and $C[k] = c_k$, $k = 1...h$ and $C[k] = l_k$, $k = h+2...m$ and $C[k] = l_k$, $k = h+1$;
2 Compute $p'_k = CalQP_k(C[1...m])$ and $C[t] = u_t$, $t = 1...h$, $t \neq i$ and $C[t] = l_t$, $t = h+1...m$;
3 **for** *each sensor* $o_i \in L$ **do**
4 $\quad p_i^{min} = p_i$ - $min(\frac{p'_{h+1} - P}{|L|}, P - P_i - \sum_{k=1,k\neq i}^{m}(p'_k - p_k))$;
5 $\quad u_i = Z_i(p_i^{min}, C[1...m])$;

Algorithm 5: Deriving u_i for L.

Deriving Filter for Z. So far, we have derived the filters for sensors in S and L. Now we investigate Z set, which is the largest set among all three in most cases. However, the filter derivation for this set is much simpler than the previous two. We first define a cut off value $cutoff = \frac{f+n}{2}$ where f is the farthest uncertainty boundary of non-zero qualification probability sensors (minimum

maximum distance) and n is the nearest uncertainty boundary of zero qualification probability sensors (maximum minimum distance). The filter boundary for $o_i \in Z$ can be set as $[l_i, u_i] = [cutoff + \frac{r_i}{2}, +\infty]$ where r_i is the length of o_i's uncertain region.

The one dimensional filter constraint can be easily extended to support two dimensional data. The filter constraint region is an annulus centered at the query point with two radii as l_i and u_i.

Intuitively, with the deployment of the probabilistic filters, the updates between sensors and server can be saved compared with basic CPNNQ execution protocol.

5 Experiments

In this section, we describe the results of evaluating our protocol using a set of real temperature sensor data (Section 5.1) and a location database (Section 5.2).

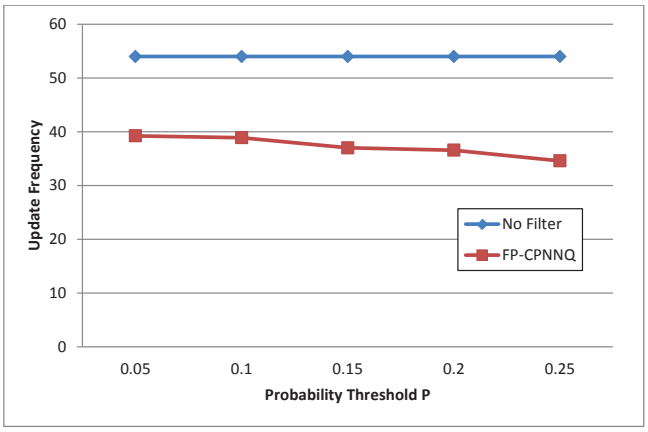

Fig. 3. Update Frequency on Temperature Data

5.1 Temperature Data

Experiment Setting. The same set of data in [31] is used for experimental evaluation. Specifically, we have 155,520 one-dimensional temperature readings captured by 54 sensors on 1st March 2004, provided by the Intel Berkeley Research lab. The temperature values are collected every 30 seconds. The lowest and the highest temperature values are 13^oC and 35^oC respectively. The domain space is $[10^oC, 40^oC]$. The uncertainty region of a sensor value is in the range of $\pm$ 1^{o}C [19]. By default, the uncertainty PDF is a normal distribution, with

the mean as the sensed value, and the variance as 1. Also, the energy for sending an uplink message is 77.4mJ, while that for receiving a downlink message is 25.2mJ [9]. Each data point is obtained by averaging over the results of 100 random queries. Each query point is generated randomly within the domain. A query has a lifetime uniformly distributed between [0, 24] hours. A query's probability threshold value, P, varies from 0.05 to 0.25. We compare our protocol with the basic protocol in which no filter is deployed. The reason we only consider no filter case is that no other work focuses on communication cost for continuous probabilistic nearest neighbor query.

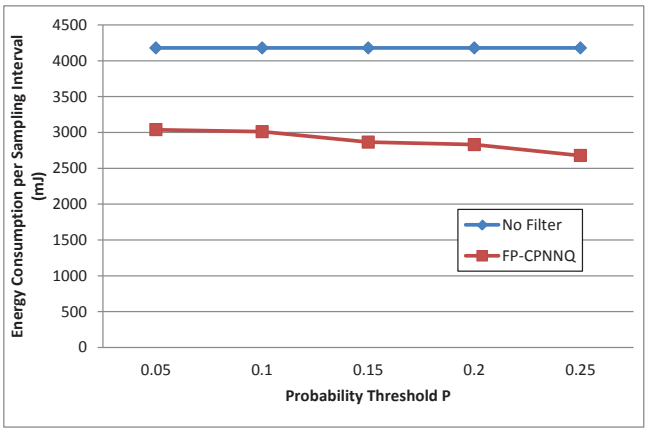

Fig. 4. Energy Consumption on Temperature Data

Experiment Results. We evaluate the probabilistic filter protocol for answering nearest neighbor query in terms of update frequency and energy consumption. As shown in Figure 3, the probabilistic filters reduce update frequency by 31% on average. When the probability threshold is 0.25, the reduction on update frequency is 36%. Similar results can be observed while evaluating energy consumption (Figure 4). The probabilistic filters reduce energy consumption on sensors by 1295 mJ per sampling interval (30 seconds) on average. The reason we have reduction in both the update frequency and energy consumption is that in some sampling periods, all sensor values stay within their corresponding filter region. As a result, no update is generated. We also observe that as the probability threshold increases, the probabilistic filters decrease the number of update messages and thus reduce energy. This is because the answer set for the queries with larger probability thresholds often has fewer qualified objects. Therefore, these answer sets are updated less frequently than that of queries with a smaller probability threshold. For example, when the probability threshold is 1.0, the answer set has at most one object.

5.2 Location Data

Experiment Setting. In this experiment, we also use the same set of location data in [31]. We simulate the movement of vehicles in a 2.0 x 2.0 km^2 European city. We use the CanuMobiSim simulator [26] to generate 5000 vehicles, which follow a smooth motion model in the streets of the city [3]. The following error model is applied to a position obtained with a (simulated) GPS device: the sensing uncertainty is obtained from a statistical error model with imprecision of 6.3m, with 95% probability [25]. The vehicles have a maximal velocity of v_{max} = 30m/s. The maximal sensing uncertainty is 10m and the sampling time interval is 1s. Thus, the radius of the uncertainty region of the vehicle is 40m. We simulate the movement of 5,000 objects over 90s, or 450,000 records. Each query point is generated randomly within the map. Each query has a lifetime uniformly distributed between [0, 90] seconds. Next, we present the results for expriments using this location dataset.

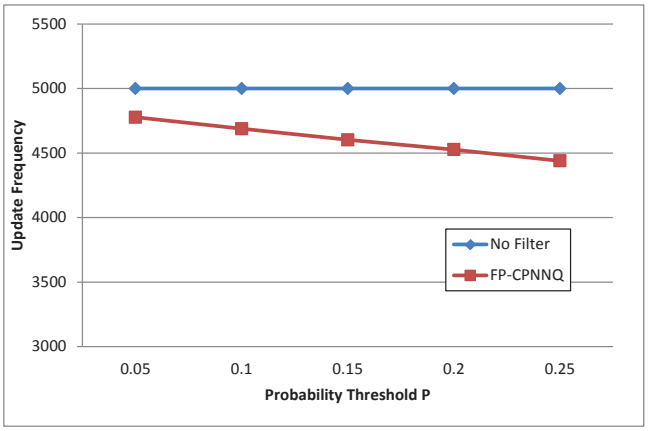

Fig. 5. Update Frequency on Location Data

Experiment Result. We also use update frequency and energy consumption as the metrics to evaluate the probabilistic filter protocol for answering nearest neighbor query over location data. As shown in Figure 5, the probabilistic filters reduce update frequency by 8% on average. For energy consumption (Figure 6), it is reduced by 30.4 J per sampling interval (1s) on average. When the probability threshold is 0.25, the reduction in both update frequency and energy consumption approaches 11%. Similar to the results on temperature data, in some sampling periods, no moving object crosses the boundary of their filter region. Update messages are not sent in these cases. However, the improvement is not as much as that for temperature data. The reason is that temperature readings, compared with vehicle locations, are relatively steady so that the filter constraints are not typically violated.

However, the performance of probabilistic filter for CPRQ [31] is much better than that for CPNNQ. This is because of the differences in their filter protocols. For CPRQ, only the sensor whose filter constraint is violated needs to send an update to the server. For CPNNQ, once a filter constraint is violated, all sensors need to send a message to the server. In the future, we will consider locality in value in order to send updates to only a subset of sensors.

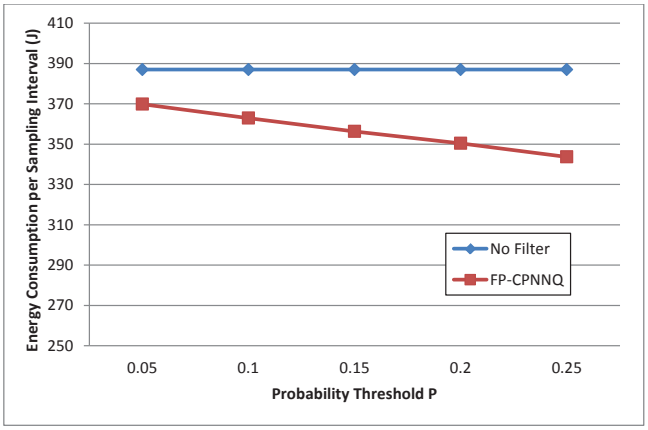

Fig. 6. Energy Consumption on Location Data

6 Conclusions

We investigated continuous nearest neighbor query execution over uncertain data. The proposed probabilistic filter protocol for processing such queries reduced communication cost and energy consumption of wireless sensors. Experimental evaluation on real-world temperature sensing data showed a reduction in the number of update messages by upto 36%. Evaluation on querying of synthetic 2D location dataset showed a reduction in both update frequency and energy consumption of 11%. The reduction in updates for continuous nearest neighbor queries is also significantly greater than that of continuous range queries.

One future direction is to extend our protocol to efficiently support multiple aggregate query execution. Another direction is to introduce tolerance in query answers in order to further reduce communication cost at sensor side.

Acknowledgment

We would like to thank Prof. Reynold Cheng (University of Hong Kong) for providing support in the early stage of this research.

References

1. Maaz Bin Ahmad, Muhammad Asif, M Hasan Islam, and Dr Sadia Aziz. A short survey on distributed in-network query processing in wireless sensor networks. In *Networked Digital Technologies, 2009. NDT '09. First International Conference on*, pages 541 –543, july 2009.
2. Panayiotis Andreou, Demetrios Zeinalipour-Yazti, Andreas Pamboris, Panos K. Chrysanthis, and George Samaras. Optimized query routing trees for wireless sensor networks. *Inf. Syst.*, 36(2):267–291, April 2011.
3. Christian Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *Mobile Computing and Communications Review*, 5(3):55–66, 2001.
4. Jinchuan Chen, Reynold Cheng, Mohamed F. Mokbel, and Chi-Yin Chow. Scalable processing of snapshot and continuous nearest-neighbor queries over one-dimensional uncertain data. volume 18, pages 1219–1240, 2009.
5. Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.
6. Reynold Cheng, Ben Kao, Alan Kwan, Sunil Prabhakar, and Y. Tu. Filtering data streams for entity-based continuous queries. In *IEEE TKDE*, volume 22, pages 234–248, 2010.
7. Reynold Cheng, Ben Kao, Sunil Prabhakar, Alan Kwan, and Yi-Cheng Tu. Adaptive stream filters for entity-based queries with non-value tolerance. In *VLDB*, 2005.
8. Alexandru Coman, Mario A. Nascimento, and Jörg Sander. A framework for spatio-temporal query processing over wireless sensor networks. In *Proceedings of the 1st Workshop on Data Management for Sensor Networks, in conjunction with VLDB, DMSN 2004, Toronto, Canada, August 30, 2004*, pages 104–110, 2004.
9. Crossbow Inc. *MPR-Mote Processor Radio Board User's Manual.*
10. Hazem Elmeleegy, Ahmed K. Elmagarmid, Emmanuel Cecchet, Walid G. Aref, and Willy Zwaenepoel. Online piece-wise linear approximation of numerical streams with precision guarantees. *PVLDB*, 2(1):145–156, 2009.
11. Tobias Farrell, Reynold Cheng, and Kurt Rothermel. Energy-efficient monitoring of mobile objects with uncertainty-aware tolerances. In *IDEAS*, 2007.
12. Tobias Farrell, Kurt Rothermel, and Reynold Cheng. Processing continuous range queries with spatio-temporal tolerance. In *IEEE TMC*, volume 10, pages 320–334, 2010.
13. Johannes Gehrke and Samuel Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 3(1):46–55, 2004.
14. Yifan Jin, Reynold Cheng, Ben Kao, Kam yiu Lam, and Yinuo Zhang. A filter-based protocol for continuous queries over imprecise location data. In *CIKM*, pages 365–374, 2012.
15. Konstantinos Kalpakis and Shilang Tang. Maximum lifetime continuous query processing in wireless sensor networks. *Ad Hoc Netw.*, 8(7):723–741, September 2010.
16. Jian Li, Amol Deshpande, and Samir Khuller. Minimizing communication cost in distributed multi-query processing. In *ICDE*, 2009.
17. Juzheng Li and Sol M. Shatz. Remote query processing in wireless sensor networks using coordinated mobile objects. In *DMS*, pages 82–87. Knowledge Systems Institute, 2010.

18. Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David E. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA*, pages 49–58, 2002.
19. Microchip Technology Inc. *MCP9800/1/2/3 Data Sheet.*
20. Rene Muller and Gustavo Alonso. Efficient sharing of sensor networks. In *MASS*, 2006.
21. Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *PVLDB*, 7(3):205–216, 2013.
22. Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*, 2003.
23. Sunil Prabhakar, Yuni Xia, Dmitri V. Kalashnikov, Walid G. Aref, and Susanne E. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. In *IEEE Trans. Comput.*, volume 51, pages 1124–1140, 2002.
24. Md. Ashiqur Rahman and Sajid Hussain. Energy efficient query processing in wireless sensor network. In *AINA Workshops (2)*, pages 696–700, 2007.
25. James Rankin. Gps and differential gps: An error model for sensor simulation. In *PLANS*, pages 260–266, 1994.
26. Illya Stepanov, Pedro José Marrón, and Kurt Rothermel. Mobility modeling of outdoor scenarios for manets. In *Annual Simulation Symposium*, pages 312–322, 2005.
27. Goce Trajcevski, Roberto Tamassia, Isabel F. Cruz, Peter Scheuermann, David Hartglass, and Christopher Zamierowski. Ranking continuous nearest neighbors for uncertain trajectories. *VLDB J.*, 20(5):767–791, 2011.
28. Goce Trajcevski, Roberto Tamassia, Hui Ding, Peter Scheuermann, and Isabel F. Cruz. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *EDBT*, pages 874–885, 2009.
29. Minji Wu, Jianliang Xu, Xueyan Tang, and Wang-Chien Lee. Top-k monitoring in wireless sensor networks. *IEEE Trans. Knowl. Data Eng.*, 19(7):962–976, 2007.
30. Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *ICDE*, 2005.
31. Yinuo Zhang and Reynold Cheng. Probabilistic filters: A stream protocol for continuous probabilistic queries. *Information Systems*, 38(1):132 – 154, 2013.
32. Yinuo Zhang, Reynold Cheng, and Jinchuan Chen. Evaluating continuous probabilistic queries over imprecise sensor data. In *DASFAA*, 2010.