Range Queries for Mobile Objects in Wireless Sensor Networks

Goce Trajcevski[*] Zachary Bischof Peter Scheuermann[†]
Department of EECS
Northwestern University
Evanston, Il 60208

goce,zach5,peters@eecs.northwestern.edu

ABSTRACT

This work addresses the problem of processing spatio-temporal range queries when the mobile entities are tracked in Wireless Sensor Network (WSN). We demonstrate that in many realistic settings, depending on the parameters of a given range query, the tracking of a particular moving object may not be needed past certain thresholds in space and/or time. We also propose and analyze distributed data-reduction techniques for the purpose of reducing the energy consumption due to communication.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]

General Terms

Algorithms, Management

1. INTRODUCTION AND MOTIVATION

The problem of tracking in WSN settings has received considerable attention in the recent years [10, 11, 17]. Different aspects have been addressed, focusing on the inherent quests: (1) energy-efficient tasks management; (2) imprecisions due to occlusion and calibration; (3) coverage properties, due to deployment and lifetime of individual sensors, etc. For example, [15] addressed the problem of maintaining the identity of the objects, while [20] has proposed convoy-trees in order to minimize the energy overheads associated with keeping the sink up-to-date with the location information of the tracked objects. The real-time quality of the tracking was tackled in [10], and works have also tackled spatial shapes [2].

However, the efficient management of spatio-temporal *range* queries WSN settings has not been addressed in a manner that would capitalize on the studies in Moving Objects Databases (MOD) [9], where a plethora of works have tackled such queries under a variety of model/system assumptions. The SOLE project [13], for example, has investigated the efficient management of spatio-temporal queries assuming that the *(location,time)* data arrives in a stream-like manner. As another example [7] addressed the trade-off between the communication and correctness of the queries' answers by delegating some of the responsibilities for maintaining the queries' answers to the participating mobile entities themselves. Works have also addressed processing of such queries under uncertainty [4, 16].

[*]Research supported by the NSF grant CNS-0910952
[†]Research supported by the NSF grant IIS-0325144/003

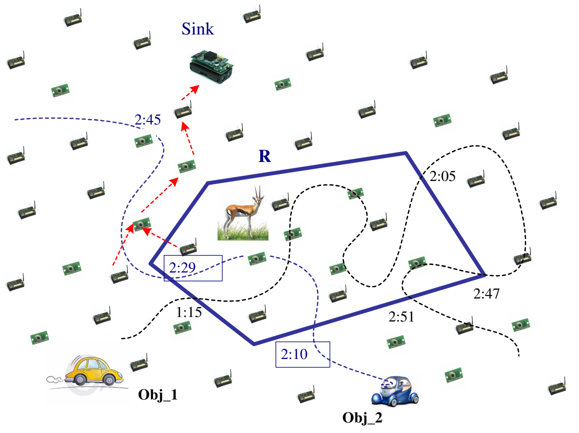

Figure 1: Spatio-Temporal Range Query

Spatio-temporal range queries are important in a variety of GIS-related application domains e.g., traffic management, battlefield, habitat and environmental health monitoring – both for the purpose of real-time decisions making based on the answer of a given query (event detection), as well as for data mining and learning purposes. In its simplest form, a range query can be stated as:
Q1: *"How many objects were inside the region R between 1PM and 3PM"*
The parameter R denotes a spatial region, often represented as a polygon [9], whereas the temporal values (1PM and 3PM) denote the bounds on the time-interval of interest for the query. Note that the syntax of **Q1** makes it a *count*-related query.

An illustration of processing a range query in typical WSN settings is given in Figure 1. The sensors (acoustic, vibration etc.) inside and on the boundary of the region of interest R, detect their distances from a given object at a particular time-instant, and collaboratively determine the location of the object at that time. For example, the sensors on the boundary of R have determined that a new object (Obj_2) has entered the region at 2:10PM, and *that* particular object has exited the region at 2:29PM. Typically, the information of interest (depending on the actual query-syntax) needs to be routed to a dedicated *sink* (cf. [20]), as illustrated by the three sensor detecting that Obj_2 is exiting the region R in Figure 1 and forwarding the information to the *Sink* node. We address the efficient management of two kinds of spatio-temporal range queries in WSN settings – one as illustrated by **Q1** above, and the other stated as:
Q2: *"Retrieve the trajectories of the objects that are inside the region R between 1PM and 3PM".*
To ensure correctness of the (continuous) range queries, one needs to track certain objects even *after* they have exited the region of interest for the query. For example, Obj_1 in Figure 1 exits R at 2:05, however, the same object re-enters at 2:47 and, in the context of **Q1** it should not be re-counted in the answer. We postulate that some *query-awareness* is needed for the nodes that are not inside or on the boundary of R, and we argue that local decisions can be brought regarding the termination of the tracking process. We also demonstrate that some of the existing MOD-techniques for balancing the (im)precision vs. communication trade-offs [18] can be readily applied.

In the rest of the paper, Section 2 presents the preliminaries and Section 3 describes our main contributions. In Section 4, we summarize and position our work, and outline directions for future work.

2. PRELIMINARIES

We assume a sensor network consisting of N nodes, $SN = \{S_1, S_2, \ldots, S_N\}$, and each node is capable of detecting an object within its range of sensing, e.g., based on vibration, acoustics or otherwise [6, 10]. Nodes are aware of their locations $S_k = (x_{Sk}, y_{Sk})$ via a GPS or other techniques e.g., trilateration [19]. They also know the locations of their one-hop neighbors, and are assumed to be static. We assume that the network is *dense* enough to ensure coverage for the purpose of detecting and localization via trilateration [11, 19].

A spatio-temporal query, **Q** is specified as a quadruple $(Sink, R, t_{begin}, t_{end}, Type)$, where the semantics of the parameters is as follows:

1. *Sink* comprises the ID and the location of the sink-node (x_S, y_S) – the final destination of the packets of relevance.
2. R is the geographic region of interest for the range query. We assume that it is bounded by a polygon, represented as a sequence $\{(x_{v1}, y_{v1}), (x_{v2}, y_{v2}), \ldots, (x_{vn}, y_{vn})\}$ of its vertices in a counter-clockwise order.
3. t_{begin} and t_{end} denote the start-time and end-time of interest for the query.
4. The last parameter (*Type*) is a description of the intended semantics of the query – e.g., `COUNT_DISTINCT` [12], would correspond to a **Q1**-type of query (cf. Section 1).

To initiate the query, the sink needs to disseminate the information through the network via a Q-REQ packet. In addition to the query-quadruple, Q-REQ specifies the location of the point $B_C = (b_{cx}, b_{cy})$ on ∂R, the boundary of R, which is closest to the sink, and the Q-REQ packet is forwarded along the trajectory defined by the line-segment $\overline{(x_S, y_S), (b_{cx}, b_{cy})}$ in a TBF-like manner [14]. Each node along the route, first needs to check whether it is inside R (or $\in \partial R$), before transmitting the Q-REQ further. The first node inside $R \cup \partial R$, will begin a *selective-flooding* of the Q-REQ data to all its neighbors, and each neighbor will recursively repeat the *selective-flooding*. The *select*-ivity means that, based on the location of the nodes, a given node will decide which one of its neighbors (or, itself) should participate in the processing [2].

3. RANGE QUERIES PROCESSING

We now present the details of the processing the spatio-temporal range queries **Q1** and **Q2**. First we introduce the concept of the network-wide *query-awareness* when processing (**Q1**). Subsequently, we demonstrate that the *dead-reckoning* policy (cf. [18]) can be adapted in distributed tracking settings for processing **Q2**-type of queries, and we how the concepts of data reduction can be used to reduce the communication cost.

3.1 Query-Awareness

After the Q-REQ has been received by the point on ∂R and propagated throughout $R \cup \partial R$ (cf. [2]), the typical processing of **Q1**-type of query proceeds as follows:

Algorithm 1:

```
(1) When an object $o_i$ is detected by the sensors near $\partial R$
(2) IF none of the sensor nodes participating in its localization has received any message regarding $o_i$'s prior participation in Q1
(3)     Assign an ID to $o_i$, which is a pair $(sn_{k3}, t_{di})$ where:
        – $sn_{k3}$ is the triplet of IDs of the sensors participating in $o_i$'s initial location-detection, that is furthest from $\partial R$
        – $t_{di}$ is the time-stamp of initial detection of $o_i$.
(4) Determine the velocity of $o_i(sn_k, t_{di})$
(5) Notify the neighbors that are expected to be handed-off the task of further tracking $o_i$
```

The determination of the neighbors that are expected to take over the tracking task can be done based on the history of the velocity-information of the given object (cf. [19]). In case no such information has been handed off (cf. step (3) above), the participating sensors are coordinating the location-sampling together with their neighbors, obtaining the first estimate of o_i's velocity.

Two important observations regarding Algorithm 1 are:

1. The sensors "near ∂R" should also include the sensors on the exterior of R, because the interior ones may yield inaccurate time-value o_i's entrance in R.
2. The task of "further tracking" is needed to avoid the multiple-counts of the same object as part of the answer to a **Q1**-type of query (cf. Obj_1 in Figure 1).

To handle the task related to the first observation and to minimize the overheads imposed by the second one, we pos-

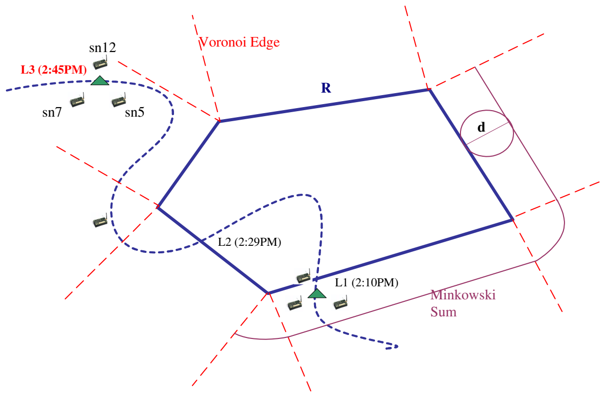

Figure 2: Query Awareness in WSN

tulate two parameters of *query awareness*, needed by the sensors that are not necessarily close to R:

• The first parameter is a value of a Euclidian distance d. Namely, the sensors that are "alert" to detecting a new object possibly affecting the answer to **Q1** are the ones that are in the zone defined[1] by the *Minkowski sum* of R and a disk of diameter d [5]. d is determined as the distance which ensures that an object with the highest expected speed moving towards ∂R from the outside, is guaranteed to be located by at least one trilateration. Clearly, d may be adjusted dynamically, based on the real-time observations. Figure 2 illustrates part of the Minkowski sum $R \oplus d$, used to determine the sensors that participate in detecting the objects approaching R.

• The second parameter is actually a dynamically adjustable spatio-temporal-threshold ST_{th}. Namely, there may be cases that are opposite to the scenario of Obj_1 in Figure 1, as illustrated by Figure 2. The history of the motion of Obj_2 illustrates that:

1. enters R at location L_1 at 2:10PM;
2. exits R at location L_2 at 2:29PM;
3. is detected at location L_3 at 2:45PM

In such settings, the sensor node sn_5 (collaborating with sn_{12} and sn_7 for localization), since it knows its own location, if it is provided with some knowledge regarding the query region R, it can make a local inference that may affect future tracking of Obj_2. Given the relative position of sn_5 with respect to R and the time-interval of interest for the query $[1PM, 3PM]$, sn_5 can infer that Obj_2 need not be tracked any further, because the likelihood of it travelling back to R before 3PM is minimal. To determine the value of ST_{th}, sn_5 does not need the entire shape of R – all it needs to know is to which cell of the Voronoi diagram of R's exterior [5] it belongs to. Each cell is associated either with an edge, or a vertex of ∂R and determining to which cell a given sensor node belongs to can be achieved locally [1].

3.2 Trajectories and Data Reduction

An important aspect of processing a **Q2**-type of queries is *how often* does the sink need the *(location,time)* values. In case a hard real-time value is required, the brute-force approach would be to transmit every location detected by an individual trilateration (subject to the sampling frequency). However, if some imprecision is acceptable, then the *dead-reckoning* policy presented in [18] can readily be applied in WSN settings.

[1]Strictly speaking, the zone of the locations of the "alert" nodes is defined by $(R \oplus d) \setminus R$.

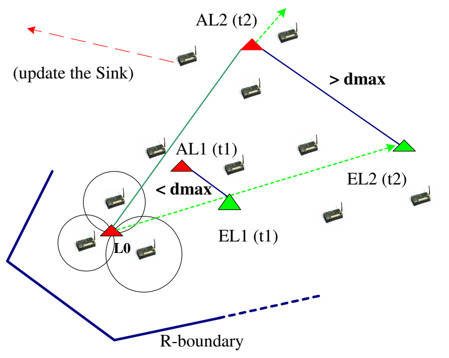

Figure 3: Dead-Reckoning and Data Reduction

The essence of the dead-reckoning policy is:
(1) The sink specifies a parameter d_{max} describing its maximum imprecision tolerance with respect to the location of the given object at a given time-instant.
(2) At any update to the sink, along with the *(location,time)* pair, the *estimated velocity* of the object is also transmitted.
(3) An update at time t is transmitted only if the *detected location* at t is further than d_{max} from the *expected location* (based on the last update).

An illustration of the dead-reckoning policy is provided in Figure 3. Upon localization at L_1, the expected velocity was transmitted to the sink, *except* now it is also handed-off as a tracking-parameter to the neighboring nodes. At time t_1, the actual location (AL_1) is closer then d_{max} to the expected location (EL_1). At t_2, the actual location AL_2 is further than d_{max} from EL_2. Hence, an update is transmitted to the sink node, containing the information $(AL_2, t_2, \overline{v}_2)$ – and $\overline{v}_2$ is handed-off to the neighbors that are expected to detect the next location of that object. As far as the sink is concerned, the trajectory of the moving object between t_0 and t_2 consisted of the line-segment $\overline{L_0, AL_2}$. The problem of calculating the expected velocity of a tracked moving object has already been addressed in (cf. [19]).

When processing **Q2**-type of a query, once again, the tracked object may exit and re-enter the query region R throughout the time-interval of interest (cf. Obj_1 in Figure 1). In that case, the dead-reckoning policy needs to be modified, so that the sink receives an update containing the last detected location inside R, prior to object exiting it. The particular object may still need to be tracked outside R, and the the query awareness issues (cf. Section 3.1) still apply.

We conclude this section with observing that if the sink does not have hard real-time constraints, further savings in communication may be achieved. Assuming that the sink requires the trajectory to be reported periodically, e.g., at fixed time-intervals, the sensors can locally perform a data reduction of the partial history of the trajectory (in-between updates) with a given error-bound ε – a parameter specified by the user [3]. Such savings are two-fold: (1) Not every

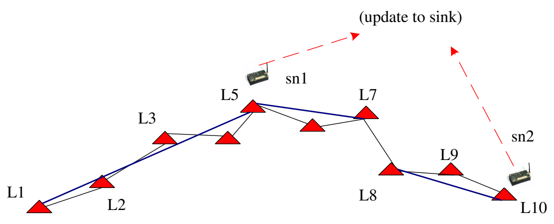

Figure 4: In-Network Simplification

node that participates in the localization/tracking needs to communicate with the sink; (2) The size of the packets transmitted to the sink will be smaller. An illustration is provided in Figure 4. Instead of every location being transmitted, at the pre-specified times, the sensor sn_1 will route of the packet containing the segment $\overline{L_1L_5}$ and, subsequently, sn_2 will do so for the sequence $\{\overline{L_5, L_7}, \overline{L_7, L_8}, \overline{L_8, L_{10}}\}$.

4. CONCLUSIONS AND FUTURE WORK

We addressed the problem processing two kinds of spatio-temporal range queries over moving objects in WSN settings, where the location information of the tracked object is obtained collaboratively by static sensors. We proposed the concept of *query-awareness* for the purpose of eliminating unnecessary tracking of objects. We demonstrated that such awareness can be obtained by utilizing some of the existing concepts in computational geometry (Minkowski sum, Voronoi diagrams), that can be constructed in a distributed manner. When it comes to reporting the trajectories of the moving objects, we demonstrated that some of the concepts studied in the MOD literature [3, 18] can be readily adapted in WSN settings.

As we mentioned, a plethora of different aspects of the tracking problem in WSN have been addressed [10, 11, 17, 20]. However, the existing approaches have not utilized the existing body of MOD-related works [4, 7, 9, 13, 16], where many of the efforts have considered different aspects in distributed and streaming-data settings. We believe that a convergence of the results is not only possible but also desirable, as it can enable implementation of various location-based applications in WSN settings.

Currently, we are implementing the techniques proposed in this poster paper in our SID-net SWANS simulator [8] for the purpose of experimentally quantifying the potential benefits. In the near future, one of our tasks is to see how the proposed techniques need to be modified to better capture some realistic settings (e.g., coverage/density, sampling). Our other two goals in this context are: – more explicit consideration of the uncertainty, both as part of the query's answer, as well as the management of the identity of the tracked objects (cf. [15]); – incorporating of the sleeping-schedule of the nodes for the purpose of further energy conserving, while still providing some quality of data/service assurances. Lastly, we are planning to extend our work towards more dynamic settings, where both the sink and (some of) the sensor nodes are mobile [20].

5. REFERENCES

[1] B. A. Bash and P. Desnoyers. Exact distributed voronoi cell computation in sensor networks. In *IPSN*, 2007.

[2] C. Buragohain, S. Gandhi, J. Hershberger, and S. Suri. Contour approximation in sensor networks. In *DCOSS*, 2006.

[3] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB Journal*, 15(3), 2006.

[4] R. Cheng, D. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving objects environments. *IEEE-TKDE*, 16(9), 2003.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., 2001.

[6] S. Gandhi, R. Kumar, and S. Suri. Target counting under minimal sensing: Complexity and approximations. In *ALGOSENSORS*, 2008.

[7] B. Gedik and L. Liu. Mobieyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10), 2006.

[8] O. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischoff, and N. Valtchanov. Sidnet-swans: A simulator and integrated development platform for sensor networks applications. In *SenSys*, 2008.

[9] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[10] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. F. Abdelzaher. Achieving real-time target tracking usingwireless sensor networks. In *IEEE Real Time Technology and Applications Symposium*, 2006.

[11] L. Lazos, R. Poovendran, and J. A. Ritcey. Analytic evaluation of target detection in heterogeneous wireless sensor networks. *TOSN*, 5(2), 2009.

[12] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM TODS*, 30(1), 2005.

[13] M. F. Mokbel and W. G. Aref. Sole: scalable on-line execution of continuous queries on spatio-temporal data streams. *VLDB J.*, 17(5), 2008.

[14] D. Niculesu and B. Nath. Trajectory based forwarding and its applications. In *MOBICOM*, 2003.

[15] J. Shin, L. Guibas, and F. Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *IPSN*, 2003.

[16] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM TODS*, 29(3), 2004.

[17] W. Wang, V. Srinivasan, K. C. Chua, and B. Wang. Energy-efficient coverage for target detection in wireless sensor networks. In *IPSN*, 2007.

[18] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, 7, 1999.

[19] L. Yang, C. Feng, J. W. Rozenblit, and H. Qiao. Adaptive tracking in distributed wireless sensor networks. In *ECBS*, 2006.

[20] W. Zhang and G. Cao. Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transcations on Wireless Communication*, 2004.