Trajectory Data Reduction in Wireless Sensor Networks

OLIVIU GHICA and GOCE TRAJCEVSKI
Northwestern University
OURI WOLFSON and UGO BUY
University of Illinois at Chicago
PETER SCHEUERMANN and FAN ZHOU
Northwestern University
DENNIS VACCARO
Northrop Grumman Corp.

This work addresses the problem of balancing the trade-off between the energy cost due to communication and the accuracy of the tracking-based trajectories' detection and representation in Wireless Sensor Networks (WSNs) settings. We consider some of the approaches used by the Moving Objects Databases (MOD) and Computational Geometry (CG) communities, and we demonstrate that with appropriate adaptation, they can yield significant benefits in terms of energy savings and, consequently, lifetime of a given WSN. Towards that, we developed distributed variations of three approaches for spatio-temporal data reduction – two heuristics (Dead-Reckoning and the Douglas-Peuker algorithm), and a variant of a CG-based optimal algorithm for polyline reduction. In addition, we examine different policies for managing the buffer used by the individual tracking nodes for storing the partial trajectory data. Lastly, we investigated the potential benefits of combining the different data-reduction approaches into "hybrid" ones during tracking of a particular object's trajectory. Our experiments demonstrate that the proposed methodologies can significantly reduce the network-wide energy expenses due to communication and increase the network lifetime.
Research supported by the NSF-CNS 0910952.
Authors addresses: Oliviu Ghica, Goce Trajcevski, Peter Scheuermann and Fan Zhou, Dept. of EECS, Northwestern University, Evanston, Il 60208;
email {ocg474, goce, peters, fanz }@eecs.northwestern.edu; Ouri Wolfson and Ugo Buy, Dept. of CS, University of Illinois at Chicago, 851 S. Morgan St., Chicago, Il 60607; email: {wolfson, buy }@cs.uic.edu; Dennis Vaccaro, Defense Systems Division, Northrop Grumman Corp., Rolling Meadows, Il 60008; email:dennis.vaccaro@ngc.com

General Terms: Algorithms, Design

Key Words: Data Reduction, Tracking, Sensor Networks

1. INTRODUCTION

In recent years, wireless sensor networks (WSN) have permeated a plethora of application domains, due to the ability of the constituent nodes to self-organize in a wireless network in addition to simply sensing and performing local calculations. This, in turn, enables their deployment in various environments, where they can observe and gather data of interest for scientific, traffic management, environmental safety/hazardz, infrastructure, health-care and military applications [Hartung et al. 2006; Kim et al. 2007; Szewczyk et al. 2004; Werner-Allen et al. 2006].

One of the most stringent constraints of WSNs is the energy, especially its con-

sumption due to communication, which can be orders of magnitude higher than the energy spent on sensing and performing local calculations [Madden et al. 2005]. Towards that end, research works have addressed various facets of the problem of energy-efficient operation of WSNs: topology/connectivity maintenance [Bhattacharya et al. 2005; Poduri et al. 2009; Santi 2005; Song et al. 2004], routing [Akkaya and Younis 2005; Pattem et al. 2008; Wu and Candan 2007; Singh et al. 1998], and in-network aggregation techniques [Krishnamachari et al. 2002; Madden et al. 2002; Manjhi et al. 2005; Shrivastava et al. 2004].

Among the canonical problems in WSN settings is the one of *tracking* of mobile objects in an area of interest. A large body of existing work has tackled problems such as improving the accuracy of the tracking process, balancing the trade-offs among the tracking accuracy, the energy consumptions, and adjusting the routing structures that convey the location-in-time information to a given sink [Alaybeyogly et al. 2010; Cao et al. 2005; Chen et al. 2003; Jeong et al. 2007; Pattem et al. 2003; Wang et al. 2005; Tanin et al. 2008; Zhang and Cao 2004; Zhong et al. 2009]

Our work also targets balancing the trade-off between the accuracy and energy-savings in tracking scenarios; however, we focus on a slightly different aspect of the problem. To better illustrate the motivation and settings, consider the following requests:

R1: *"Notify me when the tracked enemy objects have been continuously moving towards the armored units for at least 15 minutes"*

R2: *"What is the difference in the current motion of the gazelle herd around the water reservoir with respect to the last-recorded one?"*

R3: *"To which of the trajectories observed last week, is the currently tracked one most similar?"*

One commonality in all of the above requests is that—in addition to the need for coordinating the *localization* [Jeong et al. 2008; Mao and (ed.s) 2009; Zhang et al. 2009] at given time instances for the purpose of tracking—in order to process them, the dedicated sink node actually needs the whole *trajectory* of the motion, represented as a sequence of *(location,time)* data. The problem of trajectory data reduction has been studied in the MOD (Moving Objects Databases) context from the perspective of saving the storage space required and reducing the communication overheads [Cao et al. 2006; **?**; Wolfson et al. 1999]. However, in WSN the network-wide energy consumption is affected by the transmissions executed by the many relay-nodes, in addition to the ones that participate in tracking the moving objects.

The motivation and objectives of this work are illustrated in Figure 0**??**. It shows a scenario in which three sensors (S_1, S_2 and S_3) have detected the location of a tracked animal at some time, say, t_1, at which point S_1 forwards the location-data towards the sink. At the next instant t_2, when S_1, S_4 and S_5 detect the current location of that animal, S_5 transmits the location-data to the sink, possibly using a different route. Subsequently, the process is repeated with S_7 initiating another routing of the location-data towards the sink. We observe that, if the sink is willing to tolerate some imprecision about the tracked object's location, then significant energy savings in the network can be achieved. Namely, along with the animal's location, S_1 can forward its velocity to the sink. Provided that the subsequently-

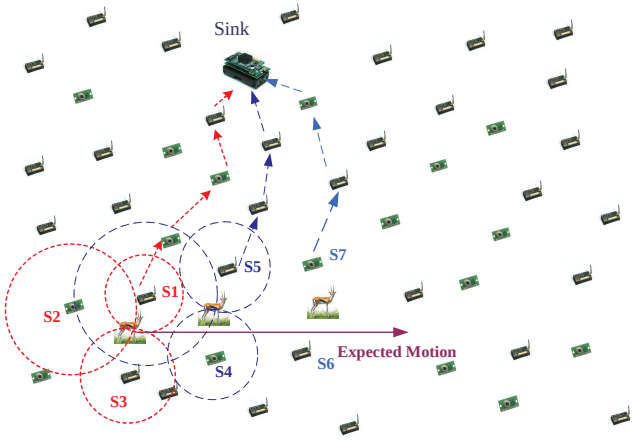


Figure. 1. Reducing multihop transmissions in trajectory tracking

detected locations stay within the error-tolerance from their expected values (which is what the sink believes to be the case), then the transmissions by S_5 and S_7 could have been avoided.

The main contributions of this work can be summarized as follows:

—We propose and implement in-network tracking based versions, of

(1) Two heuristics for data reduction:
– DDR: *Distributed* variant of the *Dead-Reckoning* policy (cf. [Wolfson et al. 1999]);
– DDP: *Distributed* variant of the popular *Douglas-Peuker* heuristic for polyline simplification used in cartography and CG (Computational Geometry) [Douglas and Peuker 1973].

(2) A *Distributed variant* (DOpt) of the optimal algorithm for polyline reduction [Chan and Chin 1996].

—We also propose and analyze the impact of different *buffer management* policies to be used by the collaborative tracking nodes for the purpose of local data reduction.

—We present extensive experimental evaluations of the benefits of the proposed methodologies using both real and synthetic motion traces datasets. We also conducted an experimental analysis of different possible "hybrid" approaches, obtained by combining the proposed methodologies. Although some results may seem counter-intuitive, they can be readily explained by carefully considering the semantics of the properties of each of the approaches.

The rest of this article is structured as follows. In Section 2 we recollect the necessary background, followed by the detailed presentation of each of the proposed approaches in Section 3. Experimental evaluations are presented in Section 4. In Section 5 we position our work with respect to the related literature, and in Section 6 we summarize and outline directions for future work.

2. PRELIMINARIES

We now discuss the basic assumptions regarding the WSN settings, and present the intuition behind the dead-reckoning policy, Douglas-Peuker heuristic for polyline data reduction, and the optimal polyline reduction algorithm. Since the last two approaches were motivated by the needs from cartography and CG, we discuss the need for adding *temporal awareness* in each of them, so that they can be applied to spatio-temporal trajectories.

We assume a sensor network consisting of N nodes, $SN = \{S_1, S_2, \ldots, S_N\}$, where each node is capable of detecting an object within its range of sensing, e.g., based on vibration, acoustics or otherwise [He et al. 2006]. Each node is aware of its locations $S_k = (x_{Sk}, y_{Sk})$ via a GPS or other techniques e.g., beacons [Mao and (ed.s) 2009; Yang et al. 2006]. Nodes are also assumed to be static and know the locations of their one-hop neighbors. We further assume that the network is *dense* enough to ensure coverage for the purpose of detection and localization via trilateration using some standard ranging method, e.g., acoustic/echo-based, RSS(Received Signal Strength) or TDOA (Time Difference of Arrival) and to ensure the selection of a neighbor as a tracking leader to whom the task of tracking can be handed-off [Jeong et al. 2007; Lazos et al. 2009; Lee et al. 2007; Pattem et al. 2003; Wang et al. 2005].

2.1 Dead-Reckoning Policy

The dead-reckoning (DR) policy, introduced in [Wolfson et al. 1999] for managing a trajectory data in MOD settings, can be viewed as a *contract* between the MOD server and the GPS-equipped localization devices on-board moving vehicles, as clients.

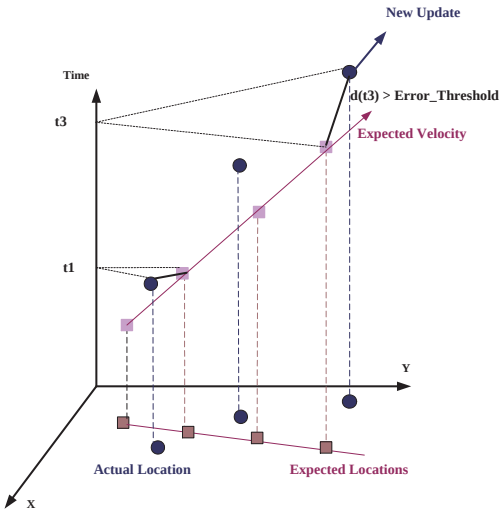


Figure. 2. Dead-reckoning location updates

Essentially, the server specifies an acceptable error-threshold for the moving ob-

jects whereabouts. In response, the mobile unit transmits its: (1) current location; (2) current time; and (3) the current estimate of the expected velocity to the MOD server. For as long as the actual locations detected at any subsequent time instant are no further than the error-threshold, say, ε, the location update is *not* transmitted to the server. Whenever the actual location deviates by $> \varepsilon$ from the expected one, the mobile unit will send an update with the new values for the *(location, time, expected_velocity)*. An illustration of the dead-reckoning policy is provided in Figure 2, and we note that, as demonstrated in [Trajcevski et al. 2006], when DR policy is used, the simplified trajectory thus generated actually corresponds to a simplification of the entire trajectory (had every location been transmitted to the MOD server) with an error-bound of $2 \cdot \varepsilon$.

2.2 Douglas-Peuker Polyline Simplification Algorithm

The problem of *polyline simplification* (equivalently, *reduction*), can be specified as follows:
Given a polyline (a sequence of vertices and the line segments in-between consecutive vertices) $P = \{V_1, V_2, \ldots, V_n\}$, generate a polyline $P' \subseteq P$ such that for every point of $p \in P$ (vertex V_i and/or a point along a line segment $\overline{V_i V_{i+1}}$) there exists a point on $p' \in P'$ for which the Euclidian distance $\overline{pp'}$ is no greater than ε.

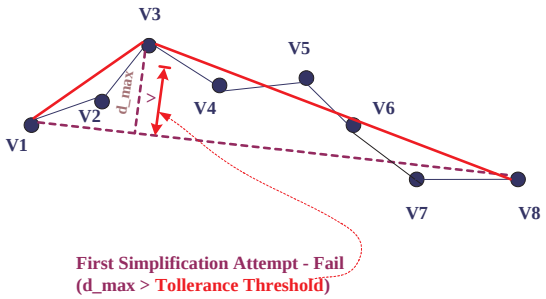


Figure. 3. Douglas-Peuker Polyline Reduction

One of the most popular methods for polyline simplification is the Douglas-Peuker (DP) heuristic algorithm [Douglas and Peuker 1973], and the pseudo-code of the algorithm that implements it can be specified as follows:
1. If every point in P is closer than ε to the $\overline{V_1 V_n}$ line segment, return $\overline{V_1 V_n}$.
2. Else
3. Find the vertex V_i with largest distance from $\overline{V_1 V_n}$
// In case of a tie, pick the vertex with smallest index i.
4. Recursively apply DP on $\{V_1, \ldots, V_i\}$ and $\{V_i, \ldots, V_n\}$ and merge the results.

Figure 3 illustrates the DP algorithm on a polyline with 8 vertices. The initial attempt to simplify it with a single line segment $\overline{V_1 V_8}$ fails, as the the distance of the vertex V_3 to $\overline{V_1 V_8}$ exceeds the acceptable error. Subsequently, the procedure is applied to the segments $\overline{V_1 V_3}$ and $\overline{V_3, V_8}$ and the simplified polyline $\{V_1, V_3, V_8\}$ is returned. As can be seen, due to the recursive invocation in line 4, the time-complexity of the above algorithm is $O(n^2)$, for a polyline with n vertices. Subse-

quently, a version of the DP algorithm with complexity $O(n \log n)$ was presented in [Hershberger and Snoeyink 1992].

2.3 Optimal Polyline Simplification

We re-iterate that the DP algorithm is a heuristic in the sense that it does not guarantee to return a polyline with a minimal number of points for a given tolerance threshold ε. Looking at the respective pseudo-code, this is reflected in the fact that the index–i of the anchor-point for the recursive invocations is chosen non-deterministically. Works from the CG community have addressed the problem of *optimal* polyline reduction [Chan and Chin 1996; Imai and Iri 1988], and pointed out that there are two distinct facets to the problem:
(1) Given a tolerance threshold ε, determine the subset of the vertices of the original polyline with minimum cardinality.
(2) Given a "subset-budget" $k \leq n$, i.e., the maximal acceptable cardinality of the reduced polyline, generate a reduction with a smallest error μ.

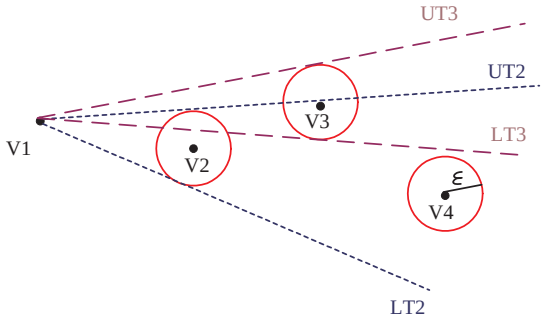


Figure. 4. Chan and Chin Optimal Polyline Reduction

The intuition behind the optimal algorithm for polyline reduction (cf. [Chan and Chin 1996]) is illustrated in Figure 4. Given the tolerance threshold ε, a collection of circles is generated, each centered at V_i ($i \in \{1, 2, \ldots, n\}$) and with radius ε. Subsequently, each vertex (e.g., V_1 in Figure 4) generates a sequence of pairs of rays corresponding to the tangents to the circles around the prior (when $i \geq 2$) and subsequent vertices. In Figure 4, the first such pair is (UT_2,LT_2), defining the boundaries of the slab between the Upper Tangent and Lower Tangent to the circle centered at V_2. The subsequent pair is (UT_3,LT_3) corresponding to the disk centered at node V_3. The key observation is that any ray emanating from V_1 that is inside the slab bounded by (UT_i,LT_i), is guaranteed not to be further than ε from V_i. At each subsequent vertex, the boundaries of intersection of the slabs are maintained—for instance, after processing V_3, the boundaries of the intersection are (UT_2,LT_3). The process terminates when the intersection between the incrementally-maintained slab and the subsequent slab, defined by the tangents to the circle centered in the next vertex in the sequence, becomes empty. Although not explicitly shown in Figure 4, it can be inferred that the intersection of the slab bounded by tangents from V_1 to the circle centered at V_4, and the slab bounded by (UT_2,LT_3) will be

empty. Hence, the segment $\overline{V_1, V_4}$ cannot be a simplified representation of the partial polyline $\{V_1, V_2, V_3, V_4\}$, with an error tolerance ε. The best that can be achieved is the simplified trajectory vertex $\overline{V_1, V_3}$, replacing $\overline{V_1, V_2}$ and $\overline{V_2, V_3}$, after which the procedure continues from V_3. A detailed specification of the algorithm is available at [Chan and Chin 1996] and we note that, since the above procedure has to be repeated for each vertex as a starting-anchor point, and both in increasing and decreasing directions of vertices' indices, the complexity of the optimal algorithm is $O(n^2)$.

2.4 Temporal Awareness

The DR policy was cast in MOD settings in [Wolfson et al. 1999] and, by transmitting the *expected velocity* at each update, it implicitly provided a "link" between the detected vs. the expected locations and the *time*, this is not the case for the DP heuristic [Douglas and Peuker 1973], nor for the optimal algorithm [Chan and Chin 1996] – both of which are geometric in nature. As demonstrated in [Cao et al. 2006], the temporal aspect of the *trajectory polyline* needs to be properly incorporated when calculating the *distance* between points and/or segments for the purpose of simplification.

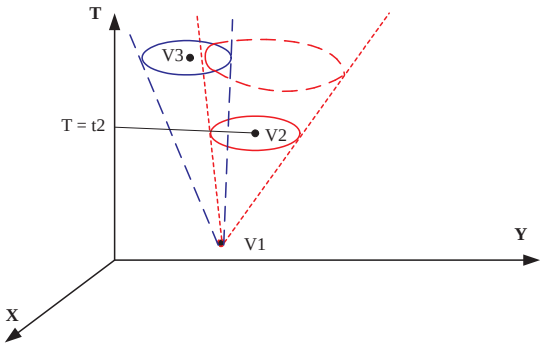


Figure. 5. Optimal Trajectories' Polyline Reduction

To explain the intuition[1], observe the spatio-temporal version of the optimal algorithm from [Chan and Chin 1996]. The distance-comparisons between the points from the original and simplified polyline need to be performed at *same time-instances*. Hence, instead of the circles around each vertex in 2D settings becoming *spheres* in 3D and the tangents-based slabs becoming sheared cones on the tangential circles of such spheres, we actually still have circles in horizontal planes for fixed values of $T = t_i$ (recall, a trajectory vertex has coordinates: $V_i = (x_i, y_i, t_i)$). For example, the bounding *sheared cone* emanating from V_1 is obtained as a union of all the tangents from V_1 to the circle centered at V_2 with radius ε and located at *the horizontal plane at time* $T = t_2$. One consequence of this is that the complexity of maintaining the bounding volumes is extended – as shown for the vertex V_3 in

[1]Detailed discussion is available in [Cao et al. 2006; Trajcevski et al. 2006].

Figure 5. Namely, instead of pair of rays, which can be updated in constant time, now the intersection of a *collection of circles* at each horizontal plane corresponding to a particular time instant needs to be maintained. Optimal versions of the 2D algorithm from [Chan and Chin 1996] extended to three (and higher) spatial dimensions has been presented in [Bose et al. 2002], and spatio-temporal variants have been presented in [Cao et al. 2006].

3. TRACKING AND IN-NETWORK TRAJECTORY DATA REDUCTION

We now proceed with presenting the distributed versions of the Dead-Reckoning (DDR) policy, followed by the distributed variants of simplification-based approaches: the Douglas-Peuker (DDP) heuristic, and the optimal algorithm (DOpt) – in spatio-temporal context *and* in WSN settings. Subsequently, we discuss the issue of managing the local buffers of the individual nodes which are used to store partial results from tracking the mobile objects' trajectories, and "hybrid" approaches that combine DDR with DDP or DOpt.

Formally, a *trajectory tracking query* **TTQ** is specified as a tuple ($Sink, t_{begin}, t_{end}$, *Th*, *Type, Buffer_Size, Buffer_Mgmt*), where the meaning of the parameters is as follows:

(1) *Sink* contains the *identification* and the *location* of the sink-node (x_S, y_S) – the final destination of the packets containing the tracking data.
(2) t_{begin} and t_{end} denote the boundaries of the time-interval of interest for the tracking query.
(3) *Th* denotes the tolerance-threshold, i.e. ε, used by the particular type of simplification.
(4) *Type* indicates whether one of the three basic types of data reduction is to be used (DDR, DDP, DOpt). In case a *Hybrid* approach is desired, it is specified as a pair (*DDR,DDP*) or (*DDR,DOpt*).
(5) *Buffer_Size* is a parameter that denotes the capacity of the buffer that is used to collect the sequence of *(location, time)* values, as they are detected.
(6) *Buffer_Mgmt* denotes the policy used when local simplification is applied by the individual nodes.

The sink node injects the **TTQ** by propagating it to its immediate one-hop neighbors and the request is subsequently spread via gossiping [Boyd et al. 2006]. We assume that the time t_{inj} at which the query is injected in the network is not later than $t_{begin} (\leq t_{end})$ parameter. Once a given node receives a **TTQ** request, it starts monitoring for objects that it needs to track and executes the corresponding algorithm based on the value of the *Type* parameter, throughout the desired time-interval of interest.

3.1 Distributed Dead-Reckoning

When a node receives a **TTQ** request with parameters $Th = \varepsilon$ and $Type = DDR$, it first checks whether it can sense an object in its range and, in collaboration with its neighbors, it tries to determine that object's location (i.e. via trilateration of range measurements obtained from neighbors) at a given time instance, say t_1 ($\forall t_1 \in [t_{begin}, t_{end}]$). The subsequent behavior of the given node depends on whether

Algorithm 1 DDR (Executed by Tracking Nodes)

Input: TTQ ($Sink$, t_{begin}, t_{end} , ε, DDR)

1: while current_time $\in [t_{begin}, t_{end}]$
2: Detect object and determine its location (e.g. via trilateration)
3: **if** no *oID* assigned to the object yet **then**
4: Assign *oID*;
5: Broadcast (*oID*, Detected_Location, current_time) to 1-hop neighbors;
// part of the tracking leader's election process
6: **else if** no Velocity Information **then**
7: Use received information for *Location* and *Time* from the neighbors
in conjunction with the *Detected_Location* and *current_time*
to determine the *Velocity*;
8: Route (*oID*, Detected_Location, current_time, Velocity) towards the sink;
// The sink is made aware of the object
9: **else if** distance(Detected_Location,Expected_location) at current_time $> \varepsilon$ **then**
10: Re-calculate the *Velocity* (via data obtained from the previous-leader);
11: Route the new (*oID*, Detected_Location, current_time, Velocity) packet
towards the sink;
12: **end if**
13: Select the next leader;
14: Transmit to it the (*oID*, Detected_Location, current_time, Velocity)
received from the previous leader;

it has received some additional information regarding the object: (1) whether the object already has an established identifier *(oID)* or not; (2) whether the sink has the awareness of *oID* being tracked on behalf of a given **TTQ**. The DDR method is formally presented by Algorithm 1.

The different behavior of the nodes executing Algorithm 1 is illustrated in Figure 6. Initially, none of the sensors S_1, S_2 and S_3 has received any data about the object that they collaboratively determine to be at location L_1 at T_1; hence, they assign the $oID = O_1$ to it. After that information has been broadcast, one of them, say S_3, declares itself as tracking leader, using some standard leader-election protocol (e.g. [He and Hou 2005]). During the next sampling epoch, S_1, S_4 and S_5 determine that O_1 is at location L_2 at time T_2. Combined with the data about the location L_1 at T_1 (transmitted from S_3), they use it to determine the *Velocity* $\overline{V}$. The complete information (O_1, L_2, T_2, $\overline{V}$) is transmitted from S_5 towards the sink using, e.g., shortest path Trajectory-Based Forwarding (TBF) [Niculesu and Nath 2003]. In addition, S_7 is selected as the next leader [He and Hou 2005; Lee et al. 2007]. Upon detecting the location of the object at time T_3 (in collaboration with S_5 and S_6), S_7 determines that O_1 is close enough to the expected location and does not send any update to the sink.

We note that, depending on the object's motion and the nodes' deployment, it may be the case that the "next leader" (cf. line 13 of Algorithm 1) may actually be the same node from the previous sampling-epoch.

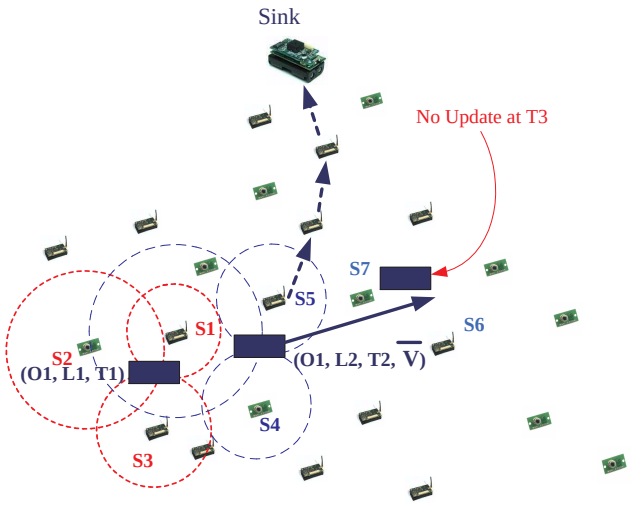


Figure. 6. Illustrating the DDR approach

3.2 Distributed Simplification Algorithms

In contrast to the DDR approach, where the sink "agrees" to have an always fresh-belief about the location of a given (tracked) object within an acceptable error-tolerance, the focus of the DDP and DOpt approaches is on locally-reducing the *number of segments that are transmitted* throughout the network for the purpose of representing the object's trajectory at the sink.

The behavior of the tracking nodes that detect the two initial locations under DDP and DOpt approaches is similar to DDR. Upon receiving a **TTQ** request with parameters $Th = \varepsilon$ and $Type = DDP$ (or $Type = DOpt$), a given node will first check if it can sense an object in its range and determine its location via trilateration. If none of the neighbors has conveyed any information regarding the sensed object, the node will assign an *oID* and, in addition, it will initialize the variable *count* to 1. The variable *count* keeps track of the number of *(location, time)* entries currently in the buffer. Otherwise, if the node receives a message stating that a particular *oID* has been located and the *count* variable is already set to 1, it will collaboratively detect the *oID*'s next location, update the counter and, using the previous location select the next leader for the subsequent localization. In addition, the *count* variable will be updated. The intuition is formally presented in Algorithm 2.

The fundamental difference between DDP and DOpt approaches is reflected in line 14 of Algorithm 2. Namely, based on the parameters of the query obtained when it is propagated in the network, the tracking nodes will select whether they will apply the heuristic or the optimal algorithm. However, regardless of the selection of a particular algorithm to be locally executed, two important observations are in order:

(1) Each algorithm operates on a *partial* representation of the moving object's trajectory—whatever is currently available in the buffer.
(2) While adding the temporal-awareness will not affect the complexity of the DDP

Algorithm 2 Distributed (Collaborative) Simplification

Input: TTQ (*Sink*, t_{begin}, t_{end} , ε, *DDP, Buffer_Size*)

1: While current_time $\in [t_{begin}, t_{end}]$
2: Detect object and determine its location (e.g. via trilateration)
3: **if** no *oID* available for the object **then**
4: Assign *oID*;
5: *count = 1*;
6: Insert (*oID*, Detected_Location, current_time, count) into *buffer*;
7: Broadcast (*oID*, Detected_Location, current_time, count, buffer);
// part of the leader election process
8: **else if** count < Buffer_Size **then**
9: Insert the *current_time* and *Detected_Location* into the buffer;
10: *count++*;
11: Use the last two values inside the buffer to determine
the velocity, next leader and transmit
(*oID*, Detected_Location, current_time, count, buffer) to it;
12: **else**
13: // count == Buffer_Size
14: Apply the corresponding *Simplification algorithm*
to the polyline in the buffer;
15: Route the simplified polyline the the sink;
16: *count = 1*;
17: Clear the buffer;
18: Select the next leader based on the last two updates in the buffer;
19: Transmit (*oID*, Detected_Location, current_time, count, buffer)
to the next leader;
20: **end if**

($O(k^2)$, where k is the size of the buffer), as illustrated in Section 2.4, it will complicate the maintenance (and update) of the partial results when applying the DOpt variant due to the need to maintain an intersection of k circular segments in the worst case. This, in turn, will increase the complexity of the DOpt to $O(k^3)$.

Figure 6 can also serve to illustrate the execution of Algorithm 2 in the context of DDP being used. Note that, unlike DDR, the DDP protocol would *not* generate any transmission from the node S_5. Instead, the buffer containing the collection $\{(L_1, T_1), (L_2, T_2)\}$ for O_1 would be sent to S_7, along with the value of *count = 2*.

3.3 Buffer Management Policies

An important observation regarding Algorithm 2 is that regardless of the choice of the particular simplification method (DDP or DOpt), there are two specific steps that impact its behavior:

(1) The simplification does not start until the local buffer – containing the previous tracking-samples plus the one(s) of its own – is completely full (line 13 of Algorithm 2).

(2) As soon as the simplification is completed, the compressed version of the trajectory polyline is transmitted to the sink (line 15 of Algorithm 2).

This, in a sense, means that the two activities (execution of the simplification algorithm and transmission to the sink) are triggered by the *same* event—local buffer filling to its capacity.

We observe that, once the local buffer has been filled up (i.e., *count* = *Buffer_Size*), the invocation of a particular simplification algorithm will reduce the current trajectory representation, thereby *freeing some of the capacity* of that same buffer. This, in turn, enables starting of another "cycle" of a sequence of sampling *(location,time)* points that can be now stored in the residual buffer space, without (and, instead of) generating a transmission to the sink. Similar observation holds once the residual free buffer space has been occupied – namely, invoking DDP or DOpt will again free some space in the same buffer.

Let $Tr(n)$ denote the set of *(location,time)* points appended to the buffer during the n^{th} iteration until the free portion of the buffer has been completely occupied. Let $Tr'(n)$ denote the compressed version of $Tr(n)$, i.e., after DDP or DOpt has been applied to it. Let C_b denote the total capacity of the buffer, and $C_{br}(n)$ denote its residual capacity after $Tr(n)$ has been substituted/represented by $Tr'(n)$. The residual capacity of the buffer can be specified with the following recurrent relation:

$$C_{br}(n) = C_{br}(n-1) - |Tr'(n)|$$

Assume that after completing the $n-$th iteration of alternating between sampling and invoking a simplification algorithm, it is no longer possible to free some residual space in the buffer by simplifying $Tr(n)$ (i.e., $C_{br}(n) = 0$). When this is the case, the node will initiate a transmission towards the sink, and clear the content of its buffer – except for the last *(location,time)* value, which is used for determining the velocity at the next sampling epoch. Clearly, the size of the data-portion of the packets that will be relayed towards the sink is $\sum_{j=1}^{n} |Tr'(j)|$, however, the important observation here is that the network, as well as the end-user application at the sink, will have a view that a "virtual buffer" of a much larger capacity (i.e. $\Sigma_{j=1}^{n} Tr(j)$) is being used during the tracking process.

The approach outlined above will trigger a transmission towards the sink when $C_{br}(n) = 0$ (i.e., $\sum_{j=1}^{n} |Tr'(j)| = C_b$). On the one hand, we observe that, it may be the case in practice that after a particular iteration k ($k < n$) of sampling and simplification, the value of the residual buffer space $C_{br}(k) > 0$ need not yield any improvements in terms of the overall data reduction. Specifically, given the nature of the object's motion, it may be the case that $C_{br}(k)$ sampled *(location,time)* points may not bring any savings after simplification. On the other hand, if the buffer had extra capacity to accommodate the subsequent s samples, the overall reduction applied to the trajectory of size $|C_{br}(k) + s|$ for a given tolerance ε could be substantial.

Let $\xi \in [0, 1]$ denote the fraction of the buffer that is already full – i.e. $\xi = 1 - (\sum_{j=1}^{j=k} C_{br}(j))/C_b$. The above discussion leads to a policy in which a threshold-value ξ_θ may be used to initiate a transmission towards the sink whenever $C_{br}(k) \leq (1 - \xi_\theta)C_b$. Investigating the problem of determining such ξ_θ value, which will depend on the history of the object's motion, is beyond the scope of this work.

However, as part of our experiments, we varied the parameter ξ_θ as part of the **TTQ** syntax and the experimental results will subsequently indicate that indeed the value of this particular parameter could improve the overall benefits of the tracking-based trajectory monitoring in a WSN settings.

We are now in the position to explain the role of the *Buffer_Mgmt* parameter of the **TTQ** syntax. Namely, the discussion presented so far assumed that each invocation of the DDP or DOpt algorithms will be applied to a particular sequence of *(location,time)* samples, i.e. $Tr(j)$, generating its reduced version $Tr'(j)$ to be stored in the local buffer. We call this policy *Partial-Scope Reduction*. In addition to this, however, the corresponding simplification algorithm can be applied to the entire buffer, including both the spatio-temporal data points due to the current iteration of samplings and the already-compressed data from the previous iterations of samplings and simplifications. For example, assume that the initial *Buffer_Size* samples represent $Tr(1)$ samples which, after applying the corresponding simplification, are replaced with $Tr'(1)$. Let $Tr(2)$ denote the trajectory consisting of the vertices that have filled in the node-buffer during the second iteration of sampling. Under the, so called, *Full-Scope Reduction*, the simplification (with the same error-tolerance ε) is applied to $Tr'(1) \cup Tr(2)$, to generate $Tr'(2)$. The *Full-Scope Reduction* as a *Buffer_Mgmt* policy is a heuristic that attempts to "link" the previously-compressed trajectory with the current-samples. As our experiments indicate, this policy is less sensitive to the variations of ξ_θ than *Partial-Scope Policy* and can further decrease the total size of the transmitted data throughout the network. However, there is another aspect of the problem that could be of interest in practical scenarios, which is affected by the chosen *Buffer_Size*, ξ_θ and *Buffer_Mgmt* combination – the "freshness" (i.e., the latency) of the data in the sink, which we address next.

3.4 Hybrid Approach

Depending on the *Buffer_Size* and ε, the DDP and DOpt approaches are likely to generate fewer routings requests with messages towards the sink through the network, when compared to DDR. This is especially true when a same tolerance-threshold is used in DDP/DOpt as in DDR (cf. [Trajcevski et al. 2006]). However, the "price" for this benefit of DDP/DOpt is that until the buffer is filled up with *(location, time)* samples, the sink has absolutely no knowledge about the tracked object's whereabouts. Once the sink received a simplified polyline, it represents the (near) past motion of the object, therefore, the sink data may not be "fresh" enough with respect to the actual data detected by the tracking sensor. Complementary to this, when using DDR, the sink can guarantee that the object's actual location is within a disk of radius ε from its expected at any time instant. Combining the two benefits (i.e., compressed near-past data while guaranteeing some error bounds on the current data) may be important in certain applications, as exemplified by the request **R1** in Section 1. To cater for such cases, we also consider the *Hybrid* heuristic which, essentially, behaves as follows:

(1) For as long as the requirements of the DDR approach for a given ε are satisfied, the tracking nodes will execute the DDP or DOpt algorithm for a given ε and *Buffer_Size*.

(2) Whenever the actual detected location is further than ε from the expected one, a new DDR cycle is started, applying the DDP or DOpt simplification to the

current (not necessarily completely filled) buffer which, upon sending the simplified trajectory to the sink, is set to the current values of the location and time, and resetting the value of *count* to 1.

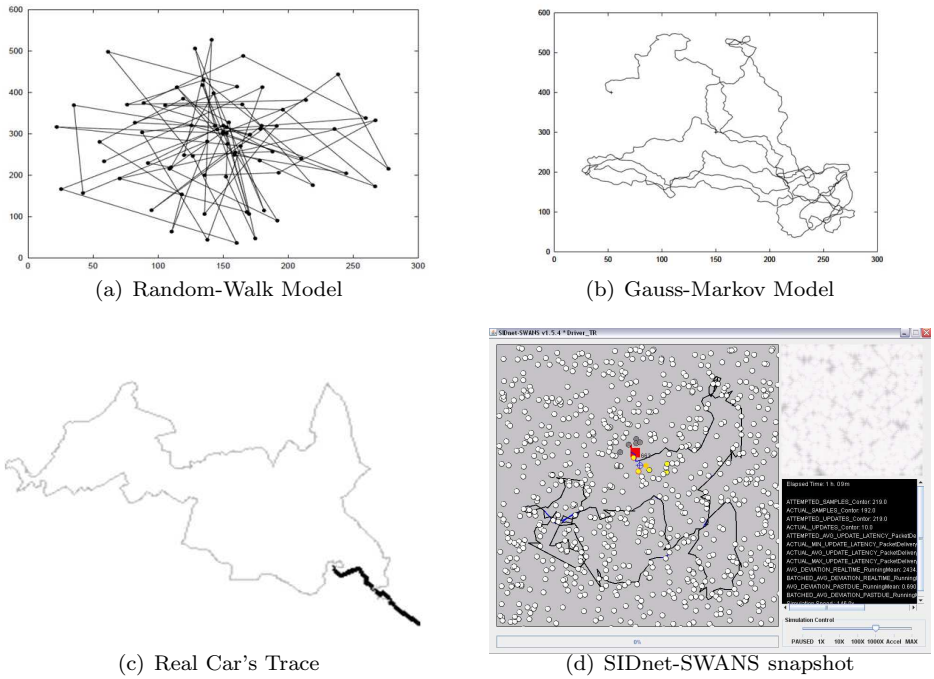
One may expect that the Hybrid policy should outperform each of the DDR, DDP and DOpt when used in isolation, in terms of the number of updates/segments transmitted to the sink and, consequently, the overall energy savings in the network. However, our experiments will demonstrate that this is not the case. Although counter-intuitive at first sight, the rationale for this behavior is that the *Hybrid* approach pays the price for the "freshness" of the data (via DDR) at the sink which, in turn, forces the simplification algorithms to be applied to a partially-full buffer.

4. EXPERIMENTAL OBSERVATIONS

Our experiments were performed using the open-source, SIDnet-SWANS simulator for WSN [Ghica et al. 2008]. The testbed consists of 750 homogeneous nodes with simulated ranging capabilities that implement the equivalent of an active ultrasonic echo ranging system, running on a standard MAC802.15.4 link layer protocol.

While mobility models such as *random walk* or *random way-point* are often used, one of their drawbacks is the lack of *spatio-temporal dependency* [Bai et al. 2003]. To address this, we used the *Gauss-Markov Mobility Model* (GMMM) [Liang and Haas 2003; Camp et al. 2002] which does exhibit spatial and temporal dependency. As illustrated in Figure 7 (based on [Ope ; Camp et al. 2002]), GMMM models yield traces that are more similar, especially in terms of sinuosity [Mueller 1968; Dodge et al. 2009] to real traces. Essentially, GMMM works on a time-slot basis where at each slot the speed and direction are computed based on the ones from the previous time-slot. Figure 7(d) illustrates a snapshot of SIDnet-SWANS Simulator performing tracking on a GMMM-based mobile object. In addition, we have performed a separate analysis using real-world prerecorded traces gathered from [Ope]. What distinguishes the real-traces is that they exhibit a *stop-and-go* behavior, which is not captured by the GMMM models. As it turns out, these differences affect the performance of our distributed algorithms, and both synthetic models and real traces have been configured to be representative of three common categories of mobile entities: humans (walk), bicyclists and automotive drivers.

The configuration space covered by our experiments is summarized in Table I, providing a total of 2,880 distinct configurations. The main parameters that we vary are the *Tracking Tolerance* ε (we note that we used the same value in the DDP and DOpt approaches), the capacity C_b of the trajectories' vertices buffer required by the DDP and DOpt components to store consecutive trajectory end-points (c.f. Section 3), the *Buffer_Size* that provides the triggering condition for the trajectory reduction algorithm, the *Buffer_Mgmt* defining the scope of applying the trajectory reduction algorithms to the buffer data and, ultimately, the specific algorithmic configuration: with simplification (DDP, DOpt) or without (DDR,RAW), and the Hybrid configurations (DDP+DDR) and (DOpt+DDR). We denote as "RAW" the naive approach, which does not apply any data reduction methodology and transmits the individual tracking samples of the form *(location,time)* to the sink node at every sampling interval.



(a) Random-Walk Model (b) Gauss-Markov Model

(c) Real Car's Trace (d) SIDnet-SWANS snapshot

Figure. 7. Synthetic and Real-life Traces

Table I. Experimental space

Simulation Area $[m]$	Sensing Range $[m]$	Tracking Tolerance $\varepsilon[m]$	DDP/DOpt Buffer Size C_b	Buffer-Flush Threshold ξ_θ	Buffer Reduction Scope	Avg. Velocity Object Type $[mph]$	Algorithmic Configuration
1,500	30	1	5	0	Partial	4 (Walk)	RAW
		10	10	.2	Full	10 (Bike)	DDR
		50	20	.4		25 (Car)	DDP
		100	40	.6			DOpt
				.8			DDP+DDR
							DOpt+DDR

We used a constant sampling-rate of 5 seconds in each of the runs of the simulator[2]. Each configuration instance from Table I has been tested against 10 distinct GMMM traces and 4 real-traces representative for *each* category of mobile objects, resulting in a total of 40,320 experiments. Each experiment spans 2 hours of simulation time and consists of two parts: (1) bootstrapping and neighbor discovery protocols in SIDnet-SWANS; and (2) the actual tracking. Although the body of experiments is extremely large, the actual reason for capping the simulation time stems from the limited duration of real-traces. Lastly, SIDnet-SWANS's nodes have been configured to meet Mica 2 Mote energy consumption specifications, briefly outlined in Table II.

Each experimental scenario consists of a single moving object traversing the net-

[2]We recognize the large body of works that have addressed the various optimizations of the sleeping schedules (e.g., [Cao et al. 2005]) when trading off accuracy for energy savings, however, this is not the objective of the current work.

Table II. Energy characteristics of *Mica2 Mote (MPR500CA)*

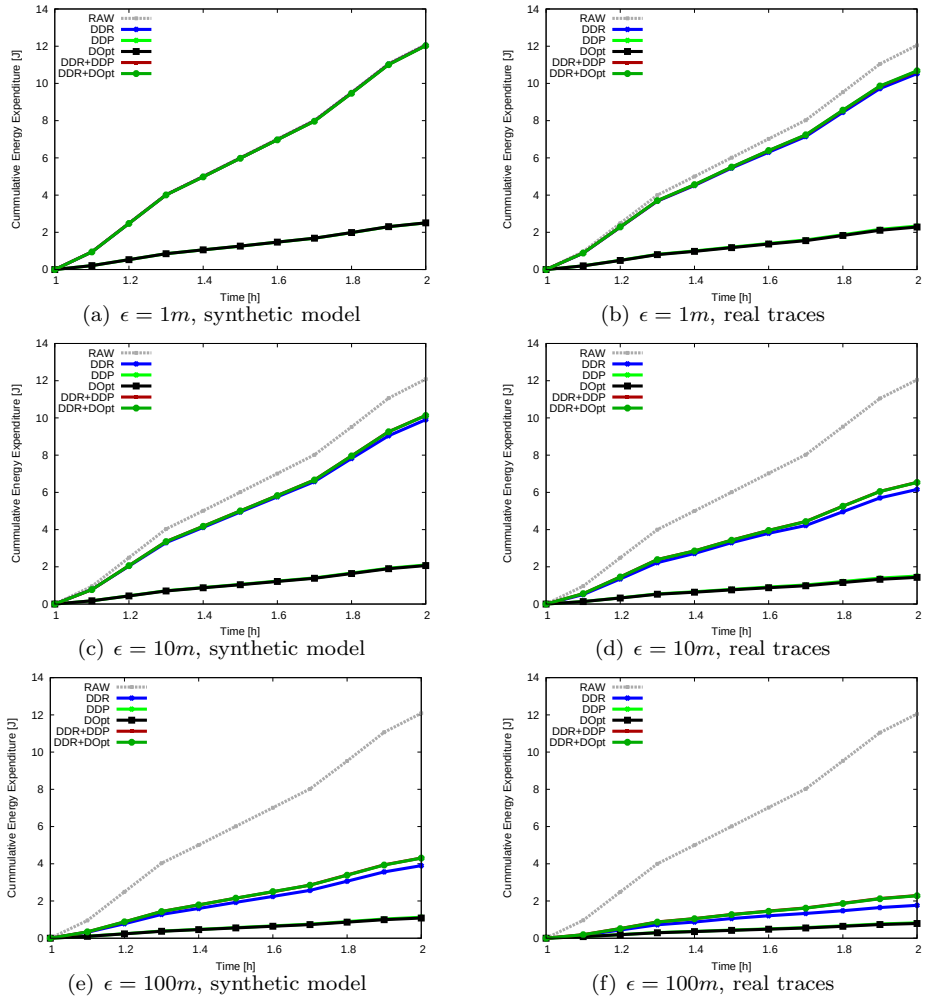
State	Based on	Energy requirement
Sensing Active	$I_s = 10mA$	$0.03mJ/ms$
Sensing Passive	$I_s = 0mA$	$0mJ/ms$
CPU Active	$I_p = 8mA$	$0.024mJ/ms$
CPU Idling	$I_i = 0.015mA$	$4.5 * 10^{-5}mJ/ms$
RADIO Transmitting	$I_t = 27mA$	$0.081mJ/ms$
RADIO Receiving	$I_r = 10mA$	$0.03mJ/ms$
RADIO Listening	$I_l = 3mA$	$0.009mJ/ms$
RADIO Off-Mode	$I_{slp} = 0.5mA$	$0.0015mJ/ms$

work field, tracked via a leader-based tracking protocol (cf. [He and Hou 2005]). Tracking information is propagated to subsequent tracking nodes which are dynamically chosen in close-proximity to the moving target as it progresses through the sensorial field.

The first results that we report concern the network-wide energy savings and consumption patterns. Figures 8 shows the run-time cumulative energy expenditure of all the nodes involved in tracking and transmitting of the location updates and/or segments of the reduced trajectory to the sink, and the energy-sensitivity to the tracking tolerances ϵ. Figures 8(a) and 8(b) show the case when the acceptable trajectory representation error is low ($\varepsilon = 1m$), whereas Figures 8(c), 8(e) and 8(d), 8(f) show the cases for $\varepsilon = 10m$ and $\varepsilon = 100m$ based on synthetic models and real-traces respectively. As it can be seen, the DDP and DOpt configurations are consistently achieving the lowest overall energy consumption, up to 10 times less than the RAW approach as shown using real-traces, while the energy consumption of the DDR and Hybrid approaches (denoted as DDP+DDR and DOpt+DDR) is reducing as the tolerance-threshold increases. The stop-and-go characteristic of real-traces is implicitly exploited by all *but* the RAW approach, leading to even higher energy savings for the DDR, DDP and DOpt approaches. As a note, the difference in performance between DDP and DOpt based components are minor, hence they sometimes overlap in the graphs, i.e. when the tolerance error gets smaller.

The network lifetime [Dietrich and Dressler 2009] is a metric of utmost importance when evaluating the effectiveness of any approach in WSN settings. In addition to the sheer energy consumption, an important aspect for the overall lifetime is the *distribution* of the energy consumption throughout the network. We quantify the distribution of the energy consumption via the standard deviation of the residual energy levels of sensor nodes in the entire network, as we illustrate it in Figure 9. As shown, the DDP/DOpt approaches achieve nearly half of the energy imbalance manifested by the RAW approach, based on synthetic models, and a third when stop-and-go conditions are considered in the real data traces. An important observation is that stop-and-go conditions create tracking hot-spots, perusing a small set of nodes that are continuously monitoring a stationary object for prolonged periods of time, as captured in Figure 9.

One may observe that the hybrid approaches are consistently consuming more energy than the DDR, DDP and DOpt in isolation; one may wonder why we are



(a) $\epsilon = 1m$, synthetic model (b) $\epsilon = 1m$, real traces

(c) $\epsilon = 10m$, synthetic model (d) $\epsilon = 10m$, real traces

(e) $\epsilon = 100m$, synthetic model (f) $\epsilon = 100m$, real traces

Figure. 8. Impact of tolerance-threshold on energy consumption based

considering it at all. The reason is two-fold:
(1) When DDR is used with a threshold ε, as demonstrated in [Trajcevski et al. 2006], the error that the sink has for the representation of the history of the object's motion is twice as large as receiving every single sample and then applying data reduction in-situ. Namely, the DDR obtained trajectory may be up to $2 \cdot \varepsilon$ from the actual object's trajectory obtained through all the individual samples. Adding DDP or DOpt in the Hybrid approach ensures the error bound of ε.
(2) When DDP and DOpt are used in isolation, the energy-savings are higher; however, as we will show next, the "freshness" of the sink's knowledge about the tracked object's whereabouts is lower (has much higher latency), a normal consequence of buffering larger amounts of data between subsequent updates. Adding DDR into

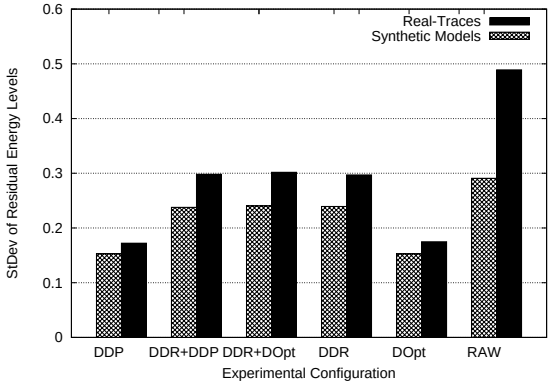
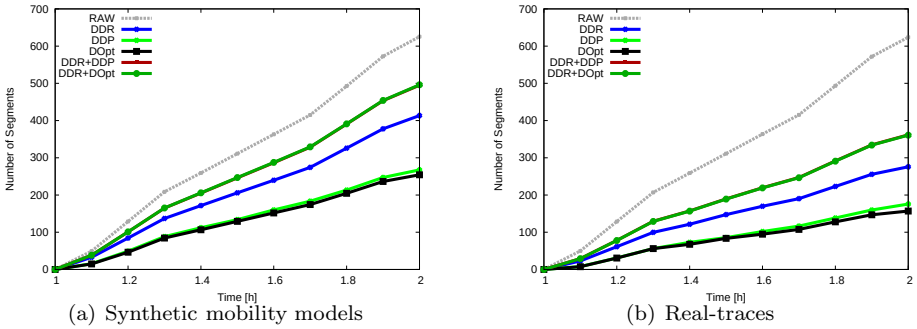


Figure. 9. Load balancing disruption analysis based on the standard deviation of the residual energy levels



(a) Synthetic mobility models (b) Real-traces

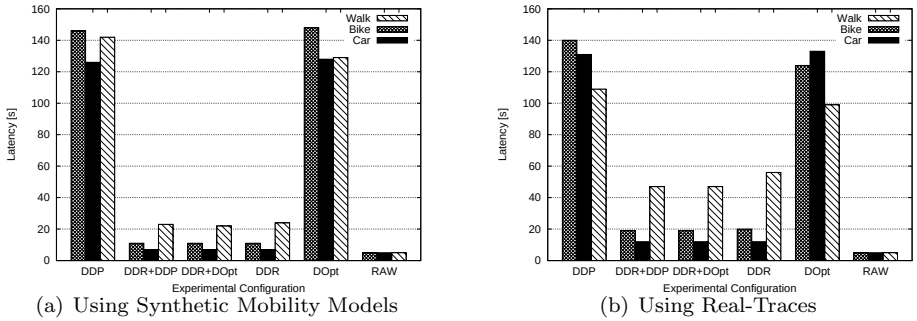
Figure. 10. Average number of trajectory segments transmitted

the Hybrid approaches helps reducing the latency of the sink's knowledge, especially when the object exhibits high mobility, hence achieving a certain balance between energy consumption and information acquisition latency.

Figure 10(b) shows a complementary measure which, in a sense, provides an intuitive justification of the results presented in Figure 8. Namely, it illustrates the number of transmitted trajectory segments towards the sink, averaged over all the scenarios outlined in Table I. As can be observed, the RAW approach transmits almost twice as many segments as DDR and almost three times as many as DDP and DOpt.

We note that although the collection of curves in Figure 10(b) (a)—(b) is similar in appearance to the collections in Figure 8 (a)—(f), they cannot be quite mapped to any of them via simple scaling. The reason is that the curves in Figure 8 (a)-(f) take also into account the energy spent in each of the relay nodes during a transmission towards the sink as well as the overhead of transferring the tracking information (i.e. partially filled buffers) from one node to another as tracking progresses.

While the DDP and DOpt approaches yield the least overhead in terms of the size



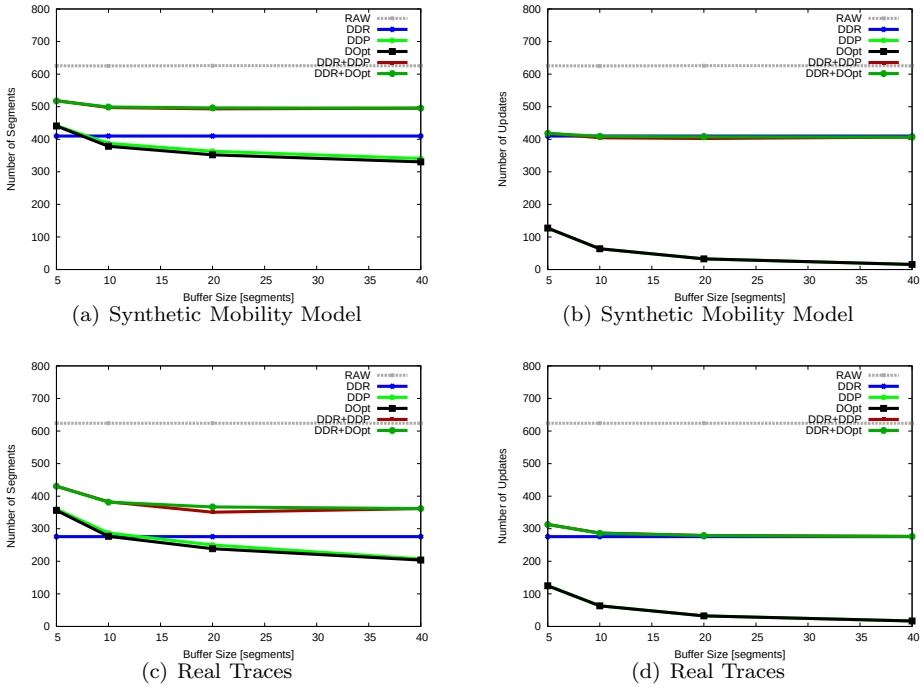
(a) Using Synthetic Mobility Models (b) Using Real-Traces

Figure. 11. Average Latency of Sink Updates

of transmitted data, as we mentioned in Section 3, each of them affects the "freshness" of the data at the sink, in terms of the latency of the sink's knowledge about the tracked object's trajectory. As illustrated in Figure 11, the RAW approach exhibits the smallest latency—basically, the time-interval between location updates sent to the sink. We observe that although the DDP and DOpt approaches yield the highest degree of energy savings, they also cause substantially long periods of sink's "un-awareness" about the actual motion of the tracked object. Although we have averaged the results over all the simulation runs, it becomes apparent that the Hybrid approaches are much better than the DDP/DOpt and, surprisingly, sometimes even better than the DDR in terms of the data-freshness in the sink. The latter case is due to the fact that, when C_b is small and ε is large, the DDP/DOpt components of the Hybrid approaches may cause an update of their own.

Having discussed the performance improvement of various algorithmic configurations, we start focusing on the impact of the rest of the tuning parameters that may alter the performances of the DDP/DOpt based approaches.

First, we start analyzing the impact of the *Buffer_Size* (C_b) over the number of segments ultimately transmitted to the sink node. While one may expect that the larger the buffer, the better, as illustrated in Figure 12(a) and 12(c), a relatively small buffer size is sufficient for attaining near-maximum performance—a particularly important observation considering the strict memory limits in sensor nodes. As shown, when the buffer capacity approaches 40 segments, the gain in the reduction of the number of segments transmitted using the DDP/DOpt approaches is minimal – (approximately 0.5KB, assuming 4B per each of the X, Y and Time coordinate value). Similar "saturation" is reached slightly earlier for the Hybrid approaches (approximately 25 segments in the buffer). As expected, the *Buffer_Size* does not affect the number of transmitted segments when DDR approach is used. Figures 12(b) and Figures 12(d) alternatively count the number of *updates* sent to the sink node, each update consisting of one or more trajectory segments. In such a setting, we observe that right around its saturation-point ($C_b \simeq 25$), the Hybrid approach converges, in the number of updates, to DDR. This means that for values of the buffer size $C_b \geq 25$, it is the DDR part of the Hybrid approach that is most likely to trigger an update to be sent to the sink, possibly causing the



(a) Synthetic Mobility Model (b) Synthetic Mobility Model

(c) Real Traces (d) Real Traces

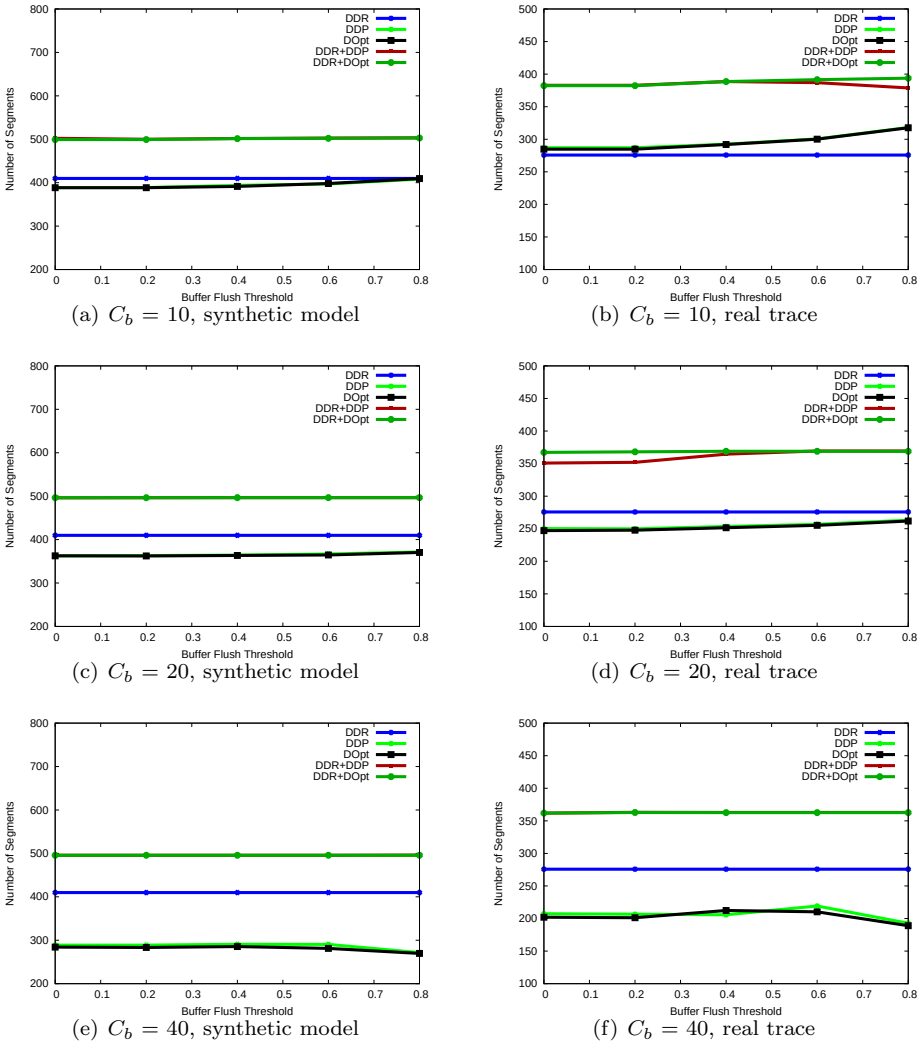
Figure. 12. Impact of the buffer size C_b over the number of updates and segments transmitted.

data-reduction algorithm to execute over only a partially-filled buffer. Also, as can be observed in Figures 12(a) and 12(c), indeed there exists a cross-over point at a certain selection of the buffer size ($C_b \simeq 8$ for synthetic models, $C_b \simeq 10$ for real traces) below which the performance of the DDP/DOpt components subside to the DDR configuration.

We finalize this section by presenting experimental observations regarding the impact of the *Buffer_Size* and *Buffer_Mgmt* policies used when specifying a given **TTQ** (cf. Section 3.3).

Figure 13 depicts a breakdown of the influence of the threshold $\xi_\theta \in \{0, .2, .4, .6, .8\}$ for different buffer sizes $C_b = \{10, 20, 40\}$ when partial-scope reduction is being adopted. As it can be seen, it confirms the intuition discussed in Section 3.3 regarding the potential impact of the choice of ξ_θ. Figure 13 (a) and (b) may appear as anomalous, but they are actually illustrating the consequences of having small buffer sizes. In such cases, most of the residual buffer sizes of subsequent iterations. i.e., increasing ξ_θ in such cases will only reduce the effectiveness of the trajectory reduction algorithm. As we mentioned in Section 3.3, apparently, there may be an optimal value for determining when the local buffer should be transmitted to the sink—however, such an investigation is beyond the scope of this work.

Figure 14 illustrates the improvement of the fully-scoped reduction vs. the partial one. The Full-Scope reduction policy has an additional benefit: it appears,



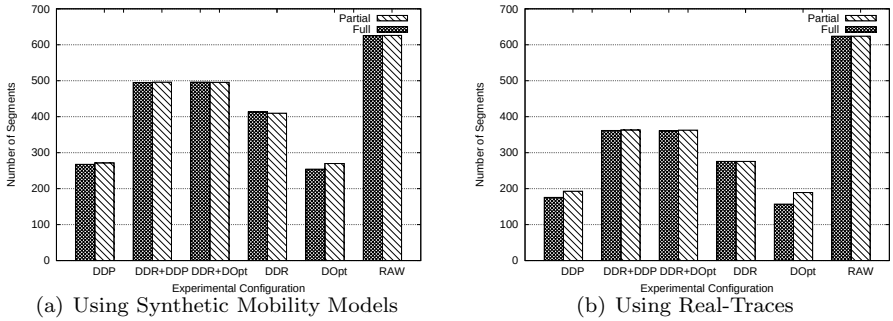
(a) $C_b = 10$, synthetic model (b) $C_b = 10$, real trace

(c) $C_b = 20$, synthetic model (d) $C_b = 20$, real trace

(e) $C_b = 40$, synthetic model (f) $C_b = 40$, real trace

Figure. 13. Impact of buffer flush threshold ξ_θ over the number of trajectory segments transmitted with partial-scope reduction

according to the results presented in Figure 15, that it is not too sensitive on the particular value of ξ_θ for the purpose of improving the performance in terms of number of trajectory segments and updates, as the Partial-Scope policy, rather exhibiting consistent performances directly proportional to ξ_θ. We re-iterate, however, that the selection of ξ_θ should also account for the latency requirement of the sink, enabling the user to achieve better performance trade-off for a particular latency-constraint. Lastly, as expected, the performance differences exhibited between the DDP- and DOpt-based configurations are not very significant.



(a) Using Synthetic Mobility Models (b) Using Real-Traces

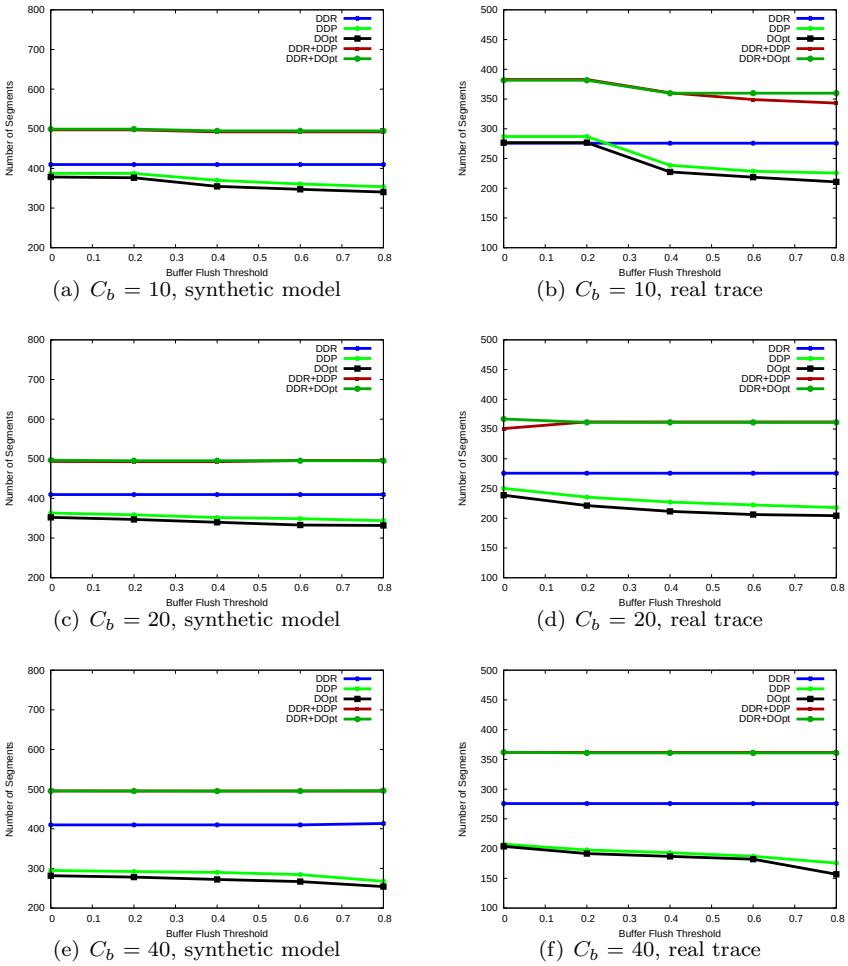
Figure. 14. Impact of the scope of the line simplification over the number of segments transmitted

5. RELATED WORK

There are two large and complementary bodies of works that are relevant to our results, originating in two different communities.

On one hand, the researchers in MOD have addressed the problem of trajectory data reduction for the purpose of reducing the storage space requirement [Cao et al. 2006] and considered the impact of the data reduction on the error in the answer of different query types. This work assumed a complete knowledge of the history of the objects' motions and applied modified versions of the Douglas-Peuker heuristic [Douglas and Peuker 1973] and the optimal algorithm developed by the CG community [Chan and Chin 1996], appropriately modifying them so that the temporal attribute was taken into account. The issues related to online reduction were introduced in [Wolfson et al. 1999] where the original DR method used in this paper was introduced. The impact of the online data reduction on the quality of the archived trajectory was considered in [Trajcevski et al. 2006] and, subsequently, a comprehensive study was conducted in [Lange et al. 2009] combining DR with other techniques. While all these works had some form of a reduced bandwidth utilization as a goal, they considered settings in which devices on-board moving objects *directly* communicate with the MOD servers. We focused more on the peculiarities of the problem when the tracking and transmission of the results to the sink has to be done in WSN settings.

On the other hand, the researchers in WSNs have cast the problems of localization and tracking as one of the canonical ones and the results abound [Cao et al. 2005; Chen et al. 2003; Jeong et al. 2007; Mao and (ed.s) 2009; Pattem et al. 2003; Tanin et al. 2008; Wang et al. 2005]. We note that, in this context, our work did not address any of the energy-efficiency and accuracy aspects of the tracking process per se—on the contrary, we completely relied on the existing works for the implementation of our algorithms in the SIDnet simulator. However, we addressed a complementary aspect to the tracking problem, namely, providing energy savings by applying in-network data reduction of the trajectories obtained via tracking. We demonstrated that sacrificing the accuracy and the "freshness" of the trajectory data at the sink can yield up to 10 times less energy expenses during the routing



(a) $C_b = 10$, synthetic model (b) $C_b = 10$, real trace

(c) $C_b = 20$, synthetic model (d) $C_b = 20$, real trace

(e) $C_b = 40$, synthetic model (f) $C_b = 40$, real trace

Figure. 15. Impact of buffer flush threshold on the number of segments with full-scope reduction

process.

A recent work that is very similar in spirit to ours is presented in [Xu and Lee 2007] where a delay-tolerant trajectory compression (DTTC) is applied to reduce the size of the transmitted data pertaining to tracking-based trajectories of moving objects. DTTC exploits the trade-offs among the delay-tolerance, accuracy, and the energy saving based on forming clusters in a given network, and relying on the cluster-heads to generate a compact description of a given (partial) trajectory. Although the objectives are the same, the specific approach taken in [Xu and Lee 2007] is complementary to ours—namely, we do not consider clustering of the underlying nodes in WSNs, and we apply different data-reduction techniques.

Another recent work which addresses similar problem as part of our work is given in [Abam et al. 2007], where line simplification algorithms are considered in stream-

ing (limited memory) settings. Both Hausdorff and Fréchet distances are used when applying the error-tolerance and algorithms are presented allowing augmentation of the memory resources under competitive ratios. In a similar context, in this article, we addressed the issue of ξ_θ as a parameter that can be used for managing the *Buffer_Size* of the tracking nodes under two different *Buffer_Mgmt* policies – however, the overall scope of our work is focused on the communication/energy savings in WSN settings.

6. CONCLUSIONS AND FUTURE WORK

We addressed the problem balancing the energy-savings with the delay/precision of the tracking-based trajectories' data management in WSN settings, and investigated the potential benefits of applying distributed spatio-temporal data reduction techniques. We proposed and analyzed three approaches: DDR, DDP and DOpt; and we also considered two additional hybrid-like combinations: (DDR,DDP) and (DDR,DOpt). Our experiments demonstrated the benefits of the proposed methods in terms of networks energy expenditures, as well as trade-offs between the data freshness at the sink and the total amount of the communicated data. We also considered and experimentally evaluated different policies for managing the local buffer in the tracking nodes, and demonstrated that selection of different values could further improve the network-wide energy savings.

There are several immediate extensions to this work. The first task is to incorporate some more realistic parameters related to the tracking aspect in our algorithms, e.g., incorporating the impact of tracking errors and leader selections; better time-synchronization [Sundararaman et al. 2005] and energy vs. accuracy trade-offs of the tracking process per se [Pattem et al. 2003; Pattem et al. 2008]; etc. Concurrently with this, we would like to investigate the prospect of dynamically adjusting the value of the *Buffer_size* C_b and of the buffer flush threshold ξ_θ in the DDP/DOpt and the Hybrid approaches for the purpose of further improving the balance between the near-past accuracy and the freshness of the data at the sink, and incorporate the sleeping schedule of the nodes as a parameter for balancing the accuracy.

Our long term goal is to adapt the approaches presented here, in order to tackle the efficient processing of some spatio-temporal queries (e.g., range, nearest-neighbor) in WSN settings and exploit the energy vs. accuracy trade-offs, especially when there are multiple such queries posed from multiple sinks.

REFERENCES

www.openstreetmap.org.

Abam, M. A., de Berg, M., Hachenberger, P., and Zarei, A. 2007. Streaming algorithms for line simplification. In *Symposium on Computational Geometry*.

Akkaya, K. and Younis, M. 2005. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks 3,* 3.

Alaybeyogly, A., Erciyes, K., Kantarci, A., and Dagdeviren, O. 2010. Tracking fast moving targets in wireless sensor networks. *IETE Technical Review 27,* 1.

Bai, F., Sadagopan, N., and Helmy, A. 2003. Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *INFOCOM*.

Bhattacharya, S., Xing, G., Lu, C., Roman, G.-C., Chipara, O., and Harris, B. 2005. Dynamic wake-up and topology maintenance protocols with spatiotemporal guarantees. In *IPSN*.

Bose, P., Chen, D. Z., Deascu, O., Goodrich, M. T., and Snoeyink, J. 2002. Efficiently approximating polygonal paths and three and higher dimensions. *Algorithmica 33,* 4.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. 2006. Randomized gossip algorithms. *IEEE Transactions on Information Theory 52,* 6, 2508–2530.

Camp, T., Boleng, J., and Davies, V. 2002. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing 2,* 5.

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

Cao, Q., Abdelzaher, T., He, T., and Stankovic, J. 2005. Towards optimal sleep scheduling in sensor networks for rare-event detection. In *IPSN*.

Cao, Q., Yan, T., Stankovic, J., and Abdelzaher, T. 2005. Analysis of target detection performance for wireless sensor networks. In *DCOSS*. 276–292.

Chan, W. and Chin, F. 1996. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal on Computational Geometry and Applications 6,* 1.

Chen, W., Hou, J., and Sha, L. 2003. Dynamic clustering for acoustic target tracking in wireless sensor networks. In *IEEE International Conference on Network Protocols (ICNP'03)*.

Dietrich, I. and Dressler, F. 2009. On the lifetime of wireless sensor networks. *TOSN 5,* 1.

Dodge, S., Weibel, R., and Forootan, E. 2009. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environments and Urban Systems*. doi:10.1016/j.compenvurbsys.2009.07.008.

Douglas, D. and Peuker, T. 1973. Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. *The Canadian Cartographer 10,* 2.

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

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

Hartung, C., Han, R., Seielstad, C., and Holbrook, S. 2006. Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *MobiSys*.

He, G. and Hou, J. C. 2005. Tracking targets with quality in wireless sensor networks. In *13th IEEE International Conference on Network Protocols (ICNP)*.

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

Hershberger, J. and Snoeyink, J. 1992. Speeding up the douglas-peuker line-simplification algorithm. In *Proceedings of the 5th International Symposium on Spatial Data Handling*.

Imai, H. and Iri, M. 1988. Polygonal approximations of a curve-formulations and algorithms. In *Computational Morphology*. Elsevier Science Publishers, New York, N.Y., 71–86.

Jeong, J., Guo, S., He, T., and Du, D. 2008. Apl: Autonomous passive localization for wireless sensors deployed in road networks. In *INFOCOM*.

Jeong, J., Hwang, T., He, T., and Du, D. H.-C. 2007. Mcta: Target tracking algorithm based on minimal contour in wireless sensor networks. In *INFOCOM*.

Kim, S., Pakzad, S., Culler, D. E., Demmel, J., Fenves, G., Glaser, S., and Turon, M. 2007. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN*. 254–263.

Krishnamachari, B., Estrin, D., and Wicker, S. 2002. Impact of data aggregation in wireless sensor networks. In *Proc. International Workshop on Distributed Event-Based Systems (DEBS)*.

Lange, R., Farrell, T., Dürr, F., and Rothermel, K. 2009. Remote real-time trajectory simplification. In *PerCom*.

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

Lee, S., Muhammad, R. M., and Kim, C. 2007. A leader election algorithm within candidates on ad hoc mobile networks. In *ICESS*.

Liang, B. and Haas, Z. J. 2003. Predictive distance-based mobility management for multidimensional pcs networks. *IEEE/ACM Trans. Netw. 11,* 5, 718–732.

Madden, S., Franklin, M., Hellerstein, J., and Hong, W. 2002. Tag: a tiny aggregation service for ad hoc sensor network. In *Proc. Fifth Symp. on Operating Systems Design and Implementation, USENIX OSDI*.

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

Manjhi, A., Nath, S., and Gibbons, P. B. 2005. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD Conference*.

Mao, G. and (ed.s), B. F. 2009. *Localization Algorithms and Strategies for Wireless Sensor Networks*. IGI Global – Invormation Science Publishing.

Mueller, J. 1968. An introduction to the hydraulic and topographic sinuosity indexes1. 371.

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

Pattem, S., Krishnamachari, B., and Govindan, R. 2008. The impact of spatial correlation on routing with compression in wireless sensor networks. *TOSN 4,* 4.

Pattem, S., Poduri, S., and Krishnamachari, B. 2003. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *IPSN*.

Poduri, S., Pattem, S., Krishnamachari, B., and Sukhatme, G. S. 2009. Using local geometry for tunable topology control in sensor networks. *IEEE Trans. Mob. Comput. 8,* 2.

Santi, P. 2005. *Topology Control in Ad Hoc and Sensor Networks*. John Wiley & Sons.

Shrivastava, N., Buragohian, C., Agrawal, A., and Suri, S. 2004. Medians and beyond: New aggregation techniques for sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

Singh, S., Woo, M., and Raghavendra, C. 1998. Power-aware routing in mobile ad hoc networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*.

Song, W., Wang, Y., Li, X., and Frieder, O. 2004. Localized algorithms for energy efficient topology in wireless ad hoc networks. In *MobiHoc* (Anchorage, AK).

Sundararaman, B., Buy, U., and Kshemkalyani, A. D. 2005. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks 3,* 3, 281–323.

Szewczyk, R., Mainwaring, A. M., Polastre, J., Anderson, J., and Culler, D. E. 2004. An analysis of a large scale habitat monitoring application. In *SenSys*.

Tanin, E., Chen, S., Tatemura, J., and Hsiung, W.-P. 2008. Monitoring moving objects using low frequency snapshots in sensor networks. In *MDM*.

Trajcevski, G., Cao, H., Wolfson, O., Scheuermann, P., and Vaccaro, D. 2006. On-line data reduction and the quality of history in moving objects databases. In *MobiDE*.

Wang, H., Yao, K., and Estrin, D. 2005. Information-theoretic approaches for sensor selection and placement for target localization and tracking in sensor networks. *Journal of Communications and Networks 7,* 4.

Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., Johnson, J., Ruiz, M., and Lees, J. 2006. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing 10,* 2.

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

Wu, S. and Candan, K. S. 2007. Power-aware single- and multipath geographic routing in sensor networks. *Ad Hoc Networks 5,* 7.

Xu, Y. and Lee, W.-C. 2007. Compressing moving object trajectory in wireless sensor networks. *IJDSN 3,* 2.

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

Zhang, Q., Sobelman, G. E., and He, T. 2009. Gradient-based target localization in robotic sensor networks. *Pervasive and Mobile Computing 5,* 1.

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

Zhong, Z., Zhu, T., Wang, D., and He, T. 2009. Tracking with unreliable node sequences. In *INFOCOM*. 1215–1223.

Oliviu Ghica received his B.Sc. degree in Computer and Electrical Engineering from "Politehnica" University of Bucharest, Romania, in 2002, followed by a M.Sc. degree at Northwestern University, USA, in 2006, in Electrical Engineering and Computer Sciences. He is currently a PhD candidate in the Department of Electrical Engineering and Computer Sciences at Northwestern University, USA. His research interest cover spatio-temporal data management and routing in large-scale wireless sensor networks with a particular interest in energy efficiency and lifetime benefits of algorithmic implementations. He is currently an active player in open-source communities focusing on simulation methodologies for sensor network applications.

Goce Trajcevski received his B.Sc. degree from the University of Sts. Kiril i Metodij, and his MS and PhD degrees from the University of Illinois at Chicago. His main research interests are in the areas of spatio-temoral data management, routing and data management in wireless sensor networks, and reactive behavior in dynamic systems. He has published over 45 papers in refereed conferences and journals and received a Best Paper Award at the CoopIS conference (2000), as well as US Geological Survey Scholar Award (2000). His research has been funded by BEA, Northrop Grumman Corp. and the NSF. He is currently an Assistant Chairman with the Department of Electrical Engineering and Computer Science at the Northwestern University.

Ouri Wolfson's main research interests are in database systems, distributed systems, and mobile/pervasive computing. He received his B.A. degree in mathematics, and his Ph.D. degree in computer science from Courant Institute of Mathematical Sciences, New York University. He is currently the Richard and Loan Hill Professor of Computer Science at the University of Illinois at Chicago, where he directs the Databases and Mobile Computing Laboratory, and the newly established Mobile Information Systems Research Center. He is also an Affiliate Professor in the Department of Computer Science at the University of Illinois at Urbana Champaign. He served as a consultant to Argonne National Laboratory, to the US Army Research Laboratories, to DARPA, and to the Center of Excellence in Space Data and Information Sciences at NASA. He is the founder of Mobitrac, a high-tech startup specializing in advanced fleet management software; it had about forty employees in Chicago and listed major companies such as Fedex among it clients, before being acquired by Fluensee. Most recently he founded Pirouette Software Inc., and currently serves as its President. The company specializes in Mobile Peer-to-Peer software for local search. Before joining the University of Illinois he has been on the computer science faculty at the Technion and Columbia University, and he has been a Member of Technical Staff at Bell Laboratories.

Ugo Buyis an Associate Professor in the Department of Computer Science of the University of Illinois at Chicago. He obtained his MS and PhD degrees in Computer Science from the University of Massachusetts at Amherst in 1983 and 1990, respectively. From 1983 to 1986 he was a Senior Software Engineer with the Digital Equipment Corporation in Hudson, Massachusetts. He has been on the UIC faculty since 1990. His main research interests are in software engineering. He has been involved in several projects sponsored by NSF and NIST whose common goal is the definition of techniques and tools for enhancing the reliability of concurrent and real-time software. He is currently investigating techniques and tools for supporting the development of software running on multicore architectures.

Peter Scheuermann is a Professor of Electrical Engineering and Computer Science at Northwestern University. He has held visiting professor positions with the Free University of Amsterdam, the University of Hamburg, the Technical University of Berlin and the Swiss Federal Institute of Technology, Zurich. During 1997-1998 he served as Program Director for Operating Systems at the NSF. Dr. Scheuermann has served on the editorial board of the Communications of ACM, The VLDB Journal, IEEE Transactions on Knowledge and Data Engineering and is currently an associate editor of Data and Knowledge Engineering. His research interests are in distributed database systems, mobile computing, sensor networks and data mining. He has published more than 120 journal and conference papers. His research has been funded by NSF, NASA, HP, Northrop Grumman, and BEA, among others. Peter Scheuermann is a Fellow of IEEE and AAAS (American Association for the Advancement of Science).

Fan Zhou received the BS degree in computer science from Sichuan University, China, in 2003 and the MS degree in computer science and engineering from University of Electronic Science and Technology of China, in 2006. He is now a PhD candidate at the University of Electronic Science and Technology of China, currently working at the DBSN lab at Northwestern University as a pre-doctoral visiting scholar. His research interests include mobile computing and wireless sensor networks.

Dennis Vaccaro is the Director of Advanced Projects for Northrop Grumman Corporation, Technical Services Sector in Rolling Meadows, IL. He has broad and extensive engineering and program management experience involving state-of-the-art military electronic systems. He has a strong record of successful product development, fiscal performance, business development and growth, technical innovation, and customer relations. He is currently a senior member of the IEEE, a member of the Association of Old Crows (AOC), and Awards Chairman of the AOC Windy City Roost Board of Directors. He holds several US Patents related to sensor devices and networks. In 2000, he was awarded the Technology Pioneer Award by the AOC. In 2003, he was elected to the AOC Electronic Warfare (EW) Hall of Fame.