Computer Communications 101 (2017) 26–43

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Efficient detection of motion-trend predicates in wireless sensor networks

Besim Avci[a,*], Goce Trajcevski[a], Roberto Tamassia[b], Peter Scheuermann[a], Fan Zhou[c]

[a] Northwestern University, USA
[b] Brown University, USA
[c] University of Electronic Science and Technology, China

ARTICLE INFO

Article history:
Received 28 July 2015
Revised 13 June 2016
Accepted 26 August 2016
Available online 1 September 2016

Keywords:
Wireless sensor networks
WSN
Distributed algorithms
Spatial data
Motion trends
Data aggregation

ABSTRACT

This work addresses the problem of efficient distributed detection of predicates capturing the motion trends of mobile objects evaluated with respect to a (boundary of a) polygonal region, in the settings in which the *(location, time)* data is obtained via tracking in Wireless Sensor Networks (WSN). Specifically, we discuss in-network distributed algorithms for detecting two motion-trend predicates: *Continuously Moving Towards* and *Persistently Moving Towards*: first for a single object, and then the corresponding variants for multiple objects. We also present methodologies which consider the energy vs. latency trade-offs when multiple tracked objects are being considered for validating the monitored predicates. Our experiments demonstrate that our proposed technique yield substantial energy savings when compared to the naïve centralized and cluster-based approaches in which the raw *(location, time)* data is transmitted to a dedicated sink where the predicates are being evaluated.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Wireless Sensor Networks (WSNs) consist of hundreds or even thousands of nodes, each equipped with devices for sensing the values of a particular physical phenomenon (e.g., temperature, humidity, vibration, etc.) and capable of performing basic computations. More importantly, nodes are also enabled with transmitters and receivers, which enables them to self-organize into a wireless network and communicate observations from different parts of the network to each other. These features have rendered WSNs an important tool in a wide range of applications, including traffic management, environmental safety, hazard detection, wildlife tracking, infrastructure maintenance, and health care (see, e.g., [25,32,50,59]).

A canonical research problem in WSN settings is the one of *tracking* mobile objects. Various facets of the problem have been investigated: from the tradeoff between energy consumption and the accuracy of the tracking process, to routing protocols for conveying location-in-time information to a given sink [12,16,29,46,57]. Typically, the location of a given object is determined by some form of collaborative trilateration among the tracking sensors equipped with different distance-estimation devices (e.g., vibrations, signal-strength, etc.). Detecting the sequence of such locations generates information about the *trajectory* of the moving object, which is a sequence of points $(L_1, t_1), (L_2, t_2), \ldots, (L_k, t_k)$ where:

- L_i denotes the (detected) location of the tracked object, in some reference coordinate system, at time t_i, and
- $(\forall i, j)\ (i < j) \Rightarrow (t_i < t_j)$.

Since each packet transmitted flow from sensor nodes to the sink depletes the energy of the nodes participating in the communication and network lifetime, purposeful coupling of sensing and transmission becomes an alternative techniques. In this work, we focus on the problem of efficient detection of certain *trends* related to the motion of the tracked object(s) relative to a region of interest inside the geographic area covered by the WSN nodes. Such predicates arise naturally in various WSN applications: in habitat monitoring scenarios, one may be interested in detecting that certain types of animals are approaching the region of a pond or a river; in security-related scenarios, one may be interested in detecting when an object is approaching the perimeter of a particular building; in traffic management, we may want to detect when a certain number of vehicles is approaching an area of interest (e.g., concert, demonstrations, congested road-segments, etc...).

* Corresponding author.
E-mail addresses: besim@u.northwestern.edu, besim.namik.avci@gmail.com (B. Avci), goce@eecs.northwestern.edu (G. Trajcevski), rt@cs.brown.edu (R. Tamassia), peters@ece.northwestern.edu (P. Scheuermann), fan.zhou.uestc@gmail.com (F. Zhou).

http://dx.doi.org/10.1016/j.comcom.2016.08.012
0140-3664/© 2016 Elsevier B.V. All rights reserved.

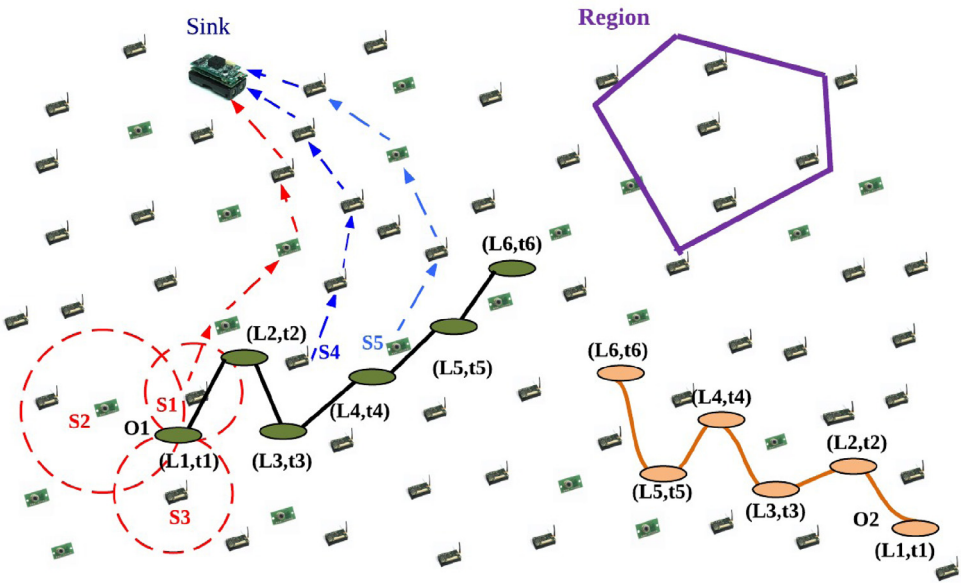


Fig. 1.1. Examples of motion-trend predicates in WSN: Starting with location L_3, the trajectory of object O_1 consisting of the sequence L_3, L_4, and L_5, exhibits a trend of continuously moving towards the *Region*. The trajectory of object O_2 does not have the property of "continuity", however, it is persistent in approaching the *Region*.

Specifically, we focus on the efficient detection of predicates describing whether a given object is *moving towards* a given region with a polygonal boundary. A motivational scenario illustrating the two different kinds of *moving towards* trends, is shown in Fig. 1.1.

If one is to use a naïve approach when detecting certain properties about an object's trajectory, the individual *(location, time)* data is transmitted from one of the sensors performing the collaborative trilateration – denoted as the *tracking principal* [22] – to the sink. However, this approach can incur a lot of unnecessary communication overhead. For typical sensor nodes, the communication, be it a transmission or active listening/reception, drains substantially more energy—up to three orders of magnitude—than the processes of sensing and local-computations [2]. Given that the batteries on many types of sensors are non-renewable, especially when deployed in inaccessible terrains, avoiding unnecessary communication is paramount. Let us consider the example of Fig. 1.1, where we are interested in detecting whether the object O_1 has been *Continuously Moving Towards* the region R for a period of at least 3 consecutive samples. The centralized approach would have each of the tracking principals transmit towards the sink messages with the detected location and time-stamp for the object. In addition to the "regular" communication overheads of this approach of transmitting each individual location to the sink, there is an added contextual waste. In particular, the sequence of points (L_3, t_3)... (L_5, t_5) will be transmitted to the sink, whereas by performing in-network aggregation it suffices for sensor S_5 to notify the sink that the tracked object satisfies the predicate. In a way, we couple the problem of object-tracking with the one of tracking the predicate of interest. On a closer look we also find another type of overhead due to the fact that naïve approach is history oblivious. Namely, the messages pertinent to the portion of the trajectory (L_1, t_1) and (L_2, t_2) are also transmitted when we could detect locally that they do not satisfy the predicate.

The objective of this work is to provide light-weight distributed protocols and algorithms that enable in-network detection of two such predicates: *Continuously Moving Towards (CMT)* and *Persistently Moving Towards (PMT)*. We re-iterate the subtle difference between these two predicates, illustrated by the trajectories in Fig. 1.1: although it *O2* does not quite move as *continuously towards* the region as *O2*, there is some *persistency* in its motion. Namely, despite the deviations in the sense of moving away from the region, over 50% of the time throughout its motion, O_2 does "turn-towards" the region.

Many applications (e.g., traffic management [41]) require detection of motion trends exhibited simultaneously by multiple objects. Towards that, one may rely on evaluating a Boolean combination of the individual in-network detected predicates based on the respective notifications transmitted to the dedicated sink. However, we postulate that further benefits are possible if the notifications about detecting the trends for individual objects are managed in a distributed and aggregated manner. Towards this, we formalize additional variants of motion-trends predicates – CMT_{mult} and PMT_{mult} – and we present efficient approaches for their distributed processing. First, we provide a baseline approach that is a hierarchical routing method with static clustering to compare our proposed scheme that is a dynamic data-centric routing algorithm [1]; both methods incorporate delay-sensitive aggregation criteria.

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

- We present distributed protocols and algorithms for efficient in-network detection of predicates capturing spatio-temporal motion trends relative to a spatial region of interest. We also investigate the impact of two different consumption policies for processing "primitive" (i.e., individual) events corresponding to a single *(location,time)* samples.
- We present efficient in-network solution for detection of the multi-objects variant of the predicates being satisfied within a given time-interval of interest, and propose their in-network detection via coupling notifications' routing and aggregation.
- We conduct experiments that provide a quantitative measure of the benefits of our proposed approach. We also experimentally analyze the energy vs. latency trade-off when detecting CMT_{mult} and PMT_{mult} variants for multiple objects via static clustering and data-centric routing trees.

A preliminary version of this paper [52] introduced distributed algorithms for detecting the *Continuously Moving Towards (CMT)* predicate, while a centralized solution was proposed in [53]. This paper extends of the results in [52] by: (1) introducing the

Persistently Moving Towards (PMT) variant of the problem; (2) introducing the multi-object variants of the respective predicates; (3) providing experimental evaluations.

The rest of this paper is structured as follows. Section 2 gives an overview of the background material, which is subsequently used in Section 3, where we present our main results on CMT and PMT predicates for the case of tracking a single object. Variations of the predicates to detect motion-trends for multiple objects are discussed in Section 4. Experimental observations are presented in Section 5 and a comparison with the relevant literature is given in Section 6. Section 7 summarizes the contributions of the paper and outlines directions for future work.

2. Preliminaries

We assume that a WSN consists of N nodes, $SN = \{sn_1, sn_2, \ldots, sn_N\}$, where each node is capable of detecting the presence of an object within its range of sensing, e.g., based on strength of a vibration or acoustic signals, or other distance-estimation technique [26].

Each node is assumed to be aware of its location, $sn_k = (x_k, y_k)$, either via a GPS or by using some other techniques e.g., collaborative multilateration [42,61]; and is also assumed to know the locations of all of its one-hop neighbors (i.e., the nodes directly within its communication range). We assume that the nodes are static and that the network is dense enough to ensure coverage for the purpose of detecting object's presence at any location. Moreover, we also assume that the density and the coverage ensure sufficient amount of nodes for both collaborative trilateration as well as a selection of a neighbor(s) to whom the task of tracking can be handed-off [29,36,37,46,57]. Lastly, we assume that between two consecutive location detections, the tracked objects move along straight line and with a constant speed. Hence, the location at any time instant in-between samplings can be obtained via linear interpolation.

Throughout this paper (and in our implementation) we do not consider issues related to sleeping-schedules of the individual nodes (cf. [17,49]) or the epoch-based synchronization and selection of tracking principals [22]. The principal election algorithm is essentially a distributed scheme that predicts the objects' future movement, and depending on the sparsity of the network, keeps the subset of nodes awake in order to ensure coverage of the tracked objects' locations. The principal coordinates the trilateration process and we assume that it is the sensor node closest to the sink (in terms of the Euclidian distance) among the nodes that can participate in the trilateration. Note that a given sensor node may be a tracking principal for more than one location-sampling instance.

Hierarchical routing is one of the approaches frequently used for data gathering/dissemination in WSN [1]. One particular type of hierarchical routing scheme is based on forming clusters and electing a designated cluster head that aggregates the data on behalf of the cluster ([24,27]). Even though some approaches may change cluster heads over time to extend the network lifetime, clusters remain static [56].

Given a set of points $P = \{p_1, p_2 \ldots, p_M\}$, their *Voronoi diagram* [5] is a planar subdivision (faces, edges and vertices) induced by the points in P with the following properties:

- Each face (Voronoi cell) contains one point $p_i \in P$ in its interior and, for all the other points $p_j \neq p_i$ of P and every point q in the interior of the face, we have $dist(q, p_j) > dist(q, p_i)$, where $dist(.,.)$ denotes the Euclidian distance between the two points.
- Each edge (Voronoi edge) corresponds to a bisector between two points, p_k and p_l, of P.

Voronoi diagrams are one of the most extensively studied structures in Computational Geometry [7], and algorithms for their construction in WSNs have also been proposed [6,43]. Among the extensions from the original definition (pertaining to a discrete set of points) are the variants for non-discrete sets of points (e.g., line-segments and polygons) [7] and throughout this work we will utilize the concept of Voronoi diagrams for the exterior of convex polygons.

As illustrated in Fig. 3.1, the edges of the Voronoi diagram of a given region R bounded by a convex polygon are defined by the rays originating at the vertices of the polygon and emanating perpendicularly to the incident edges. There are two basic types of Voronoi cells:

- Edge cell: the set of points in the plane for which the closest points on the boundary of R are along a given edge (bounded by parallel half-lines); and
- Vertex cell: the set of points in the plane for which the closest point on the boundary of R is one of its vertices (i.e., points within a wedge originating at a vertex);

For a given edge $\overline{A_{i-1}A_i}$ from R's boundary, let $VCell(\overline{A_{i-1}A_i})$ denote its Voronoi cell with respect to R. Also, let $Edge(A_i, VCell(\overline{A_{i-1}A_i}))$ denote the edge (i.e., the perpendicular half-line to $\overline{A_{i-1}A_i}$) of the $VCell(\overline{A_{i-1}A_i})$ originating at A_i, and let $Edge(A_{i-1}, VCell(\overline{A_{i-1}A_i}))$ denote the one originating at A_{i-1}. The Voronoi cell belonging to a given vertex A_i is denoted by $VCell(A_i)$ and its boundary edges will coincide with the ones corresponding to the boundary edges of the Voronoi cells belonging to the two adjacent edges to A_i (i.e., the cells of $\overline{A_{i-1}A_i}$ and $\overline{A_iA_{i+1}}$.

3. Trends detection algorithms

We now present our techniques for efficient in-network detection of the occurrence of the two motion-trend predicates: *Continuously Moving Towards* and *Persistently Moving Towards*. Before proceeding with the algorithmic details, we address two relevant issues:

1. propagation of the request from the sink node to the rest of the nodes of the network and the creation of the Voronoi diagram of the region of interest; and
2. consumption policies regarding the past locations detected along the tracking process.

3.1. Disseminating the request and events consumption

Recall that a dedicated sink node sn_k is serving as a gateway to the other application-contexts, for which it needs to raise a notification about detecting the occurrence of the desired predicate. However, at the initial phase of the application sn_k has to inform the rest of the nodes in the WSN about all the details of a particular request. Throughout this work, we consider them to consist of:

- Sink's own location and node ID.
- Description of the region of interest R, e.g., specified as the sequence of its vertices in counter-clockwise order.
- The begin-time and the end-time during which the detection of the predicates is required (e.g., t_b and t_e).
- The duration of the time interval Δ as a threshold during which CMT predicate needs to be detected. In the case of the PMT predicate, an additional parameter Θ may be given ($0 \leq \Theta \leq 1$) specifying the fraction of Δ during which the object needs to move towards R;

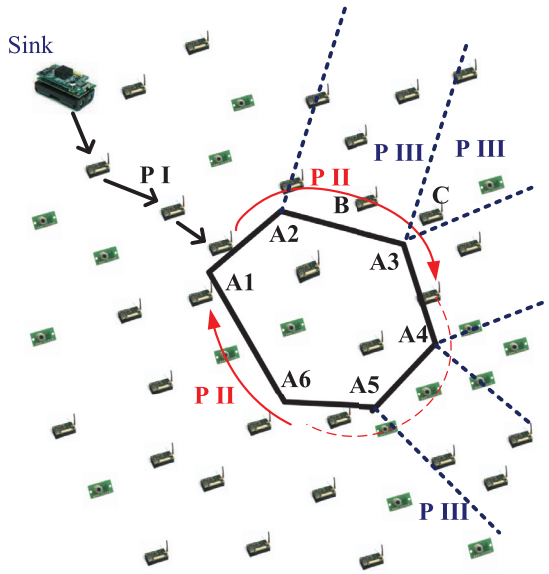


Fig. 3.1. Disseminating a request.

- The specific variant[1] of the moving towards predicate (i.e., Continuously or Persistently).

Thus, the sink will need to send a message containing the tuple ($Sink$, R, t_b, t_e, Δ, P_R) throughout the network. One obvious way to do it is via flooding [2], where every node, upon receiving the message, will perform the following tasks:

1. Forward it to its neighbors that it has not heard from yet (at the time of receiving the message);
2. Proceed with detecting which Voronoi cell of R it belongs to. Towards this, the node needs to find the point on R (along the edges or in a given vertex) which is geographically closest to its location.

The above naïve approach of disseminating requests may incur a significant overhead in terms of energy consumption and brings limitations to memory capacity of nodes, since there may be many predicates being monitored over the course of network lifetime. To avoid certain communication overheads induced by flooding, we observe that for the purpose of detecting the corresponding Voronoi cell to which a given sensor node, sn_j, belongs to, it need not be aware of all the vertices of R. Hence, we propose the following three-phase dissemination protocol, illustrated in Fig. 3.1.

Phase I (P I): In this phase, instead of starting the flooding process, the sink simply sends the packet containing the quintuple ($Sink$, R, t_b, t_e, Δ, P_R) to the sensor node on the boundary of R that is closest to it, using Trajectory-Based Forwarding (TBF) [45].

Phase II (P II): In the second phase, the node that received the request from the sink will forward the request to its neighbors along the boundary of R, each of which will recursively propagate it in the chosen reference-direction.

Phase III (P III): The third phase of the dissemination protocol can actually be pipelined with the second phase. In this phase, the moment a particular node along the outer-boundary of R receives the request, it determines the edge (or vertex) of R closest to it—implying the Voronoi cell that it belongs to. Subsequently, that node will selectively notify its neighbors in the exterior of R about the edge defining the boundaries of its $Vcell$.

The phases are illustrated in Fig. 3.1. For instance, during *P I* the sink sends a packet to the sensor node near edge $\overline{A_1A_2}$ which, in turn, will initiate *P II* for the nodes along the boundary of P_R. Once those nodes have been notified about the details of a particular request, each of them will (attempt to) propagate the request throughout the corresponding Voronoi cell, following *P III*. For instance, the sensor node B will send the message ($Sink$, $\overline{A_2A_3}$, t_b, t_e, Δ, P_R) to its neighbors. Note that the sensor nodes that are closest to a given vertex of R (e.g., node $C \in VCell(A_3)$ in Fig. 3.1) will transmit the two edges incident to the vertex to its neighbors in R's exterior. Thus, node C will send the message: ($Sink$, ($\overline{A_2A_3}$, $\overline{A_3A_4}$), t_b, t_e, Δ, P_R) to its neighbors. The reason for this approach is to help the subsequent nodes in the WSN – in particular, $VCell(A_3)$ for the node C – which may receive more than one such message, disambiguate which $VCell$ they belong to and, of course, which $VCell$ does a particular location of the tracked object belong to.

Our proposed three-phase dissemination protocol should yield substantial savings in terms of communication cost when compared to the naïve flooding of the request, given the expensive nature of the flooding.

There is one more issue that needs to be determined for the purpose of detecting the occurrence of the predicates: the *consumption policy* of the individual location-samples. To illustrate this aspect, consider a scenario where the CMT predicate has been detected within five consecutive samples. Clearly, this detection should initiate a notification sent to the sink. Now the question becomes: should the 6*th* location sample be considered as an indication for another notification that the object is moving towards (with respect to the 5*th* sample)? Clearly, this is something that needs to be decided, as a matter of policy, by the application designers. A detailed discussion of consumption policies for the primitive constituent events upon a detection of a desired composite event/predicate is beyond the scope of this work (see [13]). However, we note that the algorithms presented in the rest of this section will work correctly for both *chronicle* based consumption (i.e., only the oldest location-sample participating in the detection is discarded, the rest are still considered) and *cumulative* based consumption (i.e., the moment the composite predicate is detected, all the participating location-samples are ignored and the detection starts anew).

3.2. Continuously moving towards (CMT)

When detecting the CMT predicate, the sink will start the three-phase protocol with the message ($Sink$, R, t_b, t_e, Δ, CMT). Upon completing the pre-processing stage, the nodes in the WSN can begin combining the tracking process with detecting whether the CMT predicate has been satisfied with a particular localization-instance. Towards this goal, upon trilateration, the node elected to be the principal of the tracking process, say, sn_{TP}, will execute Algorithm 1 (CMT).

The execution of the CMT algorithm amounts to the tracking principal sn_{TP} receiving the accumulated time (T_A) of the continuous motion towards the target R from the previous tracking principal[2] up to, and including, the previously-observed location. Subsequently, it calculates the value of variable T_{CT}, which updates T_A in accordance with the object's motion along segment $\overline{L_pL_c}$. Should the combined values exceed the desired threshold Δ, node sn_{TP}

[1] One may argue that the predicate in question could be inferred from the cardinality of the argument's signature – however, in practice, one would also expect an option to explicitly select the predicate on a particular user interface.

[2] We re-iterate that a particular sensor node can serve as a tracking principal for more than one localization instance.

Algorithm 1 CMT — executed by the tracking principal.

Input: Request parameters (*Sink*, t_b, t_e , Δ, R, CMT);accumulator structure containing the previously detectedlocation+time (L_p, t_p); accumulated time T_A of continuously moving towards R up to t_p.

1: Detect the location L_c of the tracked object at the current time t_c
   // via trilateration with neighboring nodes
2: T_{CT} = *TotalTimeTowards*((L_c, t_c),(L_p, t_p), R, T_A)
3: **if** $T_{CT} \geq \Delta$ **then**
4: notify *Sink*
5: Update T_A in accordance with the consumption policy
6: **else**
7: $L_p \leftarrow L_c$;
8: $t_p = t_c$;
9: $T_A \leftarrow T_{CT}$;
10: **end if**
11: Send $((L_p, t_p), T_A)$ to the nexttracking principal

will initiate a notification to the *Sink* along a shortest-path route in a TBF manner [45]. The moment the notification is sent, the accumulator variable T_A is either set to 0 (cumulative consumption) or decremented by the duration of the time-interval corresponding to the very first localization that initiated the detection of the CMT predicate at the current location-sample. We note that any time the value of T_A is updated from 0 to some $\varepsilon > 0$, we need to retain a queue (FIFO) that will maintain all the values following ε throughout the rest of the tracking process.

To calculate value T_{CT}, node s_{TP} executes procedure *TotalTimeTowards*, which is specified by Algorithm 2. We illustrate this procedure with the scenario of Fig. 3.2, which shows a region R bounded by a pentagon with vertices $\{A_1, A_2, A_3, A_4, A_5\}$, along with its Voronoi cells. Assume that the application is interested in detecting whether a given object has been continuously moving towards R for at least 35 time-units and five location-samples with the respective time-values.

First, note that Algorithm 2 distinguishes between two main cases:

1. The tracked object is inside the Voronoi cell of an edge (handled by lines 1—16)
2. The tracked object is inside the Voronoi cell of a vertex (handled by lines 17—39)

The rationale is that if the object's location falls inside a VCell of a given edge *and* the previously sampled location is inside the same VCell – then the object can either move completely towards or completely away from the region R, throughout the entire interval of its motion inside that VCell. This is illustrated with locations (L_1, 0) and (L_2, 20) in Fig. 3.2. Since $dist(L_2, \overline{A_5A_1}) < dist(L_1, \overline{A_5A_1})$ and both L_1 and L_2 are in the same $VCell(\overline{A_5A_1})$, the T_{CT} value is updated to 20.

If, on the other hand, the tracked object's current and previous locations are in $VCell(A_i)$ belonging to vertex A_i of the polygonal boundary of R, then we have an additional case to consider (cf. line 20 of Algorithm 2): namely, if the perpendicular from A_i to the line defined by L_c and L_p falls inside the line-segment $\overline{L_cL_p}$, we know that at the terminus of the perpendicular, the motion plan of the tracked object has changed from MovingTowards (i.e., the distance to R begins to increase). This is illustrated with the portion $\overline{L_iL_3}$ of the segment $\overline{L_2L_3}$ in Fig. 3.2. Both L'_2 and L_3 are in $VCell(A_5)$ and, initially, the object is moving closer towards R, i.e., its distance to A_5 is decreasing. However, at point L'_2, which is the terminus of the perpendicular line from A_5 to $\overline{L_i, L_3}$, the distance has reached its minimum at begins to grow. Hence, at (L_3, 40) the time of moving towards R is set to $T_{CT} = 0$.

Finally, we note that in both main cases – the current location-sample being within a Voronoi cell of an edge or a vertex – Algorithm 2 makes recursive calls when the previous and the current location-samples belong to different Voronoi cells. This scenario applies to both sampling (L_3, 40) and (L_5, 80) in Fig. 3.2. Specifically, when calculating the value of T_{CT} at (L_3, 40), firstly the location of L_i – intersection of $\overline{L_2L_3}$ and $Edge(A_5, VCell(\overline{A_5A_1}))$ is found and the expected time at that location (26.6 time-units) is calculated via linear interpolation. Subsequently, we have two recursive calls (lines 13—14 and 35—36 in Algorithm 2), each calculating the respective time spend moving towards R along the segments $\overline{L_2L_i}$ and $\overline{L_iL_3}$.

Since each recursive call decreases the total length of the segment used, Algorithm 2 is guaranteed to terminate. As for its com-

Algorithm 2 TotalTimeTowards (TTT) — executed by the tracking principal.

Input: Accumulator structurecontaining the previously detected location+time (L_p, t_p),along with the accumulated time T_A of continuously movingtowards R up to t_p, and the currently detected location andtime(L_c, t_c)

1: **if** $L_c \in VCell(\overline{A_{i-1}A_i})$ $(i \in \{1, 2, \ldots, n\})$ **then**
2: **if** $L_p \in VCell(\overline{A_{i-1}A_i})$ **then**
3: **if** dist(L_c, $\overline{A_{i-1}A_i}$) $\geq$dist(L_p, $\overline{A_{i-1}A_i}$) **then**
4: // the object is moving away
5: $T_T \leftarrow 0$;
6: **else**
7: $T_T \leftarrow T_A + (t_c - t_p)$;
8: **end if**
9: **else**
10: //L_c and L_p are in different VCells
11: L_I = *IntersectPoint*($\overline{L_cL_p}$, *Edge*(A_i, *VCell* ($\overline{A_{i-1}A_i}$));
12: t_I = *InterpolateTime*(L_I, $\overline{L_cL_p}$);
13: T'_A = *TotalTimeTowards*((L_I, t_I), (L_p, t_p), T_A,R);
14: T_T = *TotalTimeTowards*((L_c, t_c), (L_I, t_I), T'_A,R);
15: **end if**
16: return T_T
17: **else**
18: // L_c is inside a *VCell* of a vertex, say A_i
19: **if** $L_p \in VCell(A_i)$ **then**
20: **if** The point P_{min} of the minimal distance betweenA_i and $\overline{L_cL_p}$ is inside $\overline{L_cL_p}$ **then**
21: $T_T \leftarrow 0$;
22: // the object switched from MovingTowards
23: // to MovingAway from R at P_{min}
24: **else if** dist(L_c, A_i) $\geq$dist(L_p, A_i) **then**
25: // the object is still moving away
26: $T_T \leftarrow 0$;
27: **else**
28: // the object is strictly moving towards R
29: $T_T \leftarrow T_A + (t_c - t_p)$;
30: **end if**
31: **else**
32: //L_c and L_p are in different VCells
33: L_I = *IntersectPoint*($\overline{L_cL_p}$,*Edge*(A_i, *VCell*($\overline{A_{i-1}A_i}$));
34: t_I = *InterpolateTime*(L_I, $\overline{L_cL_p}$);
35: T'_A = *TotalTimeTowards*((L_I, t_I), (L_p, t_p), T_A,R);
36: T_T = *TotalTimeTowards*((L_c, t_c), (L_I, t_I), T'_A,R);
37: **end if**
38: return T_T
39: **end if**

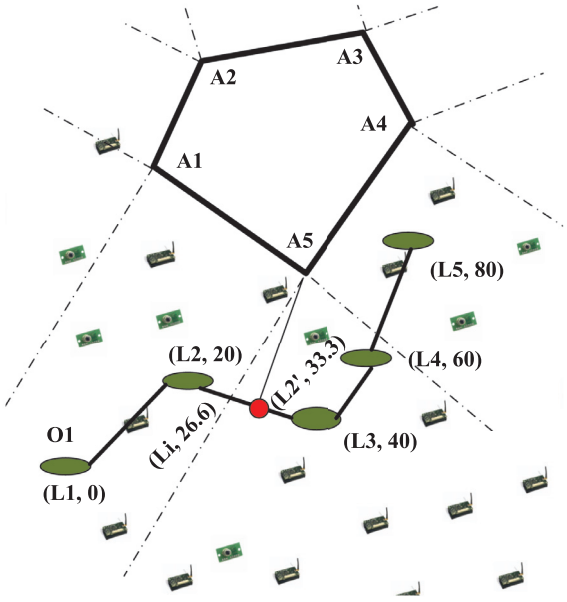


Fig. 3.2. Detecting Continuously Moving Towards.

plexity, note that, in the worst case (e.g., a fast-moving object coupled with a small convex polygon with n vertices), a given line segment intersects the edges of $O(n)$ Voronoi cells. This is the bound on the number of the recursive calls, each of them taking a constant amount of time to complete within a single cell. Hence, the running time of Algorithm 2 is $O(n)$, which is also an upper bound on the number of messages that the current principal may need to exchange in order to determine all the actual cells that participate in the CMT predicate satisfaction. Assuming that the localization step is performed regularly in intervals of δ_s, the running time of Algorithm 1 is $O(\lceil (t_e - t_b)/\delta_s \rceil n)$.

3.3. Persistently moving towards (PMT)

The crucial feature separating the *PMT* predicate from the *CMT* one is that the change of the tracked object's motion in a manner that will cause it to move away from the region R need not instantly invalidate the *PMT* predicate. This, in turn, demands corresponding modifications in the corresponding processing algorithm(s). The pseudo-code of the main algorithm executed by a particular tracking principal processing the PMT predicate is shown in Algorithm 3.

Algorithm 3 maintains a sequence of the points corresponding to locations detected at consecutive time-instants, in addition to the time-accumulator variable. The reason for maintaining the sequence of past points is due to the possibility that within a given time-interval Δ, the object has not exceeded the threshold of persistently moving towards R for $\Theta \cdot \Delta$ time. Hence, the algorithm will have to "slide" the Δ-window and incorporate the subsequent location samples, however, to maintain its temporal size, some of the older data will need to be discarded (cf. the *while*–loop in lines 11—14).

We note that the variable T_{PT} in line 2 of Algorithm 3 has a very similar meaning to the variable T_{CT} (cf. line 2.) from Algorithm 1 – to update the time-accumulator by incorporating the impact of the currently sampled location with respect to the previous one. The function used to calculate that effect – *PartialTimeTowards*, is also very similar to its "counterpart" – *TotalTimeTowards* from Algorithm 1, with one subtle difference. Namely, if the current location falls inside the wedge corresponding to the Voronoi cell of a given vertex ($VCell(A_i)$), then the shift from moving towards trend into moving away, does not annul the time-accumulator (cf. lines 20—23 of Algorithm 2). Instead, Algorithm 4 will use linear interpolation to determine how long of the motion between the previous and current location was spent moving towards R and add it to the time-accumulator variable. This is reflected in the lines 18–22 in the pseudo-code of Algorithm 4.

Algorithm 3 PMT – executed by the tracking principal.

Input: Request parameters ($Sink$, t_b, t_e , Δ, Θ, R, PMT)
Accumulator List structure containing a *sequence* ofpreviously-detected location+time pairs $(L_{p1}, t_{p1}), \ldots, (L_{pk}, t_{pk})$ along with the accumulated time T_A of persistently moving towards R up to t_{pk}

1: Detect the location L_c of the tracked object at the current time t_c
   // via trilateration with neighboring nodes
2: T_{PT} = *PartialTimeTowards*$((L_c, t_c),(L_{pk}, t_{pk}), R, T_A)$;
3: **if** $T_{PT} \geq \Theta \cdot \Delta$ **then**
4: notify the Sink
5: Update the Accumulator List structure in accordance with consumption policy
6: **else**
7: $L_{p(k+1)} \leftarrow L_c$;
8: $t_{p(k+1)} = t_c$;
9: $T_A \leftarrow T_{PT} + T_A$
10: $j = 1$
11: **while** $(t_{p(k+1)} - t_{pj}) > \Delta$ **do**
12: Remove (L_{pj}, t_{pj}) from the Accumulator List
13: Decrement T_A by the time that the object spentmoving towards R during the time-interval $[t_{pj}, t_{p(j+1)}]$
14: **end while**
15: **end if**
16: Send Accumulator List structure and T_A to thenext tracking principal;

The complexity of the running time of Algorithm 4 is bounded by $O(n)$, where n is the number of vertices of the polygon bounding the region R. Although the complexity of the Algorithm 4 is the same as the complexity of the Algorithm 2, there is an extra overhead in the running time of Algorithm 3 when compared to the one of Algorithm 1. Namely, at the end of its execution, the tracking principal is supposed to transmit the entire sequence of points to the next tracking principal. Hence, assuming once again that the sampling occurs in intervals of δ_s, the upper bound on the running time of the Algorithm 3 is $O(\lceil (t_e - t_b)/\delta_s \rceil \lceil \Delta/\delta_s \rceil n)$.

3.4. Global observations

We conclude this section with a few summary observations regarding the properties of the predicates whose algorithmic solutions we presented.

The first observation is illustrated in Fig. 3.3, in which the left portion (Fig. 3(a)) shows that there are three categories of trends of the moving objects with respect to the given region: some that move continuously towards, some that move continuously away; and a group of objects which does neither (called "lurkers"). Similar categorization is shown in Fig. 3(b) with respect to the persistency-variant. The main observation is that we can:

- Reuse the query dissemination protocol with all its phases (cf. Section 3.1) of constructing the boundary and the respective regions of the Voronoi cells
- Algorithms 1 and 3 almost-verbatim to detect when a particular object is *Continuously Moving Away*, instead of towards. The

Algorithm 4 PartialTimeTowards (PTT) – executed by the tracking principal.

Input: Accumulator List structure containing a *sequence* ofpreviously detected location+time pairs $(L_{p1}, t_{p1}), \ldots, (L_{pk}, t_{pk})$ along with the accumulated time T_A ofpersistently moving towards R up to t_{pk}; location and timeof the current sample (L_c, t_c)

1: **if** $L_c \in VCell(\overline{A_{i-1}A_i})$ $(i \in \{1, 2, \ldots, n\})$ **then**
2: **if** $L_p \in VCell(\overline{A_{i-1}A_i})$ **then**
3: **if** dist(L_c, $\overline{A_{i-1}A_i}$) ≥dist(L_p, $\overline{A_{i-1}A_i}$) **then**
4: // the object is moving away
5: $T_T \leftarrow 0$;
6: **else**
7: $T_T \leftarrow T_A + (t_c - t_p)$;
8: **end if**
9: **else**
10: //L_c and L_p are in different VCells
11: $L_I = IntersectPoint(\overline{L_cL_p}, Edge\ (A_i, VCell(\overline{A_{i-1}A_i})))$;
12: t_I = $InterpolateTime(L_I, \overline{L_cL_p})$;
13: T'_A = $TotalTimeTowards((L_I, t_I), (L_p, t_p), T_A, R)$;
14: T_T = $TotalTimeTowards((L_c, t_c), (L_I, t_I), T'_A, R)$;
15: **end if**
16: return T_T
17: **else**
18: // L_c is inside a *VCell* of a vertex, say A_i
19: **if** $L_p \in VCell(A_i)$ **then**
20: **if** The point P_{min} of the minimal distance betweenA_i and $\overline{L_cL_{pk}}$ is inside $\overline{L_cL_{pk}}$ **then**
21: t_{p-min} = $InterpolateTime(P_{min}, \overline{L_cL_{pk}})$;
22: $T_T \leftarrow (t_{p-min} - t_{pk})$;
23: /∗ incorporate the time-interval corresponding to the portion of thesegment $\overline{L_cL_{pk}}$ during which theobject was still moving towards R (i.e., before itbegan to move away) ∗/
24: **end if**
25: **else**
26: //L_c and L_p are in different VCells
27: L_I = $IntersectPoint(\overline{L_cL_p}, Edge(A_i, VCell\ (\overline{A_{i-1}A_i}))$;
28: t_I = $InterpolateTime(L_I, \overline{L_cL_p})$;
29: T'_A = $PartialTimeTowards((L_I, t_I), (L_p, t_p), T_A, R)$;
30: T_T = $PartialTimeTowards((L_c, t_c), (L_I, t_I), T'_A, R)$;
31: **end if**
32: return T_T
33: **end if**

main difference would be when comparing the relative positioning of two consecutive locations with respect to the region of interest, R.

The second observation, in a sense, pertains to the "subsumption" between the predicates that we introduced. Let CMT(t_b, t_e, Δ, R) and PMT(t_b, t_e, Δ, Θ, R) denote the corresponding predicates for *Continuously Moving Towards* and *Persistently Moving Towards*, where all the values in the respective argument-signatures are the same (i.e., same intervals of interest $[t_b, t_e]$; same duration of the continuity/persistency Δ; and same region R). Similarly, let CMA(t_b, t_e, Δ, R) and PMA(t_b, t_e, Δ, Θ, R) denote the respective predicates for the *Moving Away* case. We have the following relationships:

$$(\forall t_b, t_e, \Delta, R) CMT(t_b, t_e, \Delta, R) \Rightarrow PMT(t_b, t_e, \Delta, \Theta, R)$$

$$(\forall t_b, t_e, \Delta, R) CMA(t_b, t_e, \Delta, R) \Rightarrow PMA(t_b, t_e, \Delta, \Theta, R)$$

The proof(s) is a straightforward consequence of the definitions of the respective predicates. Namely, the additional parameter of *PMT* (resp. *PMA*) predicate – Θ – is restricted to values between 0 and 1, indicating a fraction of Δ. Thus, one can trivially add a "virtual Θ" with a fixed value of 1 in the *CMT* (resp. *CMA*) predicate.

4. Trends detection for multiple objects

We now present the extensions to the approaches for motion trends detection, in order to handle settings involving multiple objects. More specifically, we focus on the cases in which a given sink may be interested in detecting whether a certain number of mobile objects, κ, have been *Continuously Moving Towards* a region R (retaining the other parameters – i.e., for a period Δ within the time-interval $[t_b, t_e]$). To separate these predicates from the ones used when detecting the validity of motion trends pertaining to a single object, we use CMT_{mult} and PMT_{mult} to denote the corresponding variants of the predicate focusing on multiple objects.

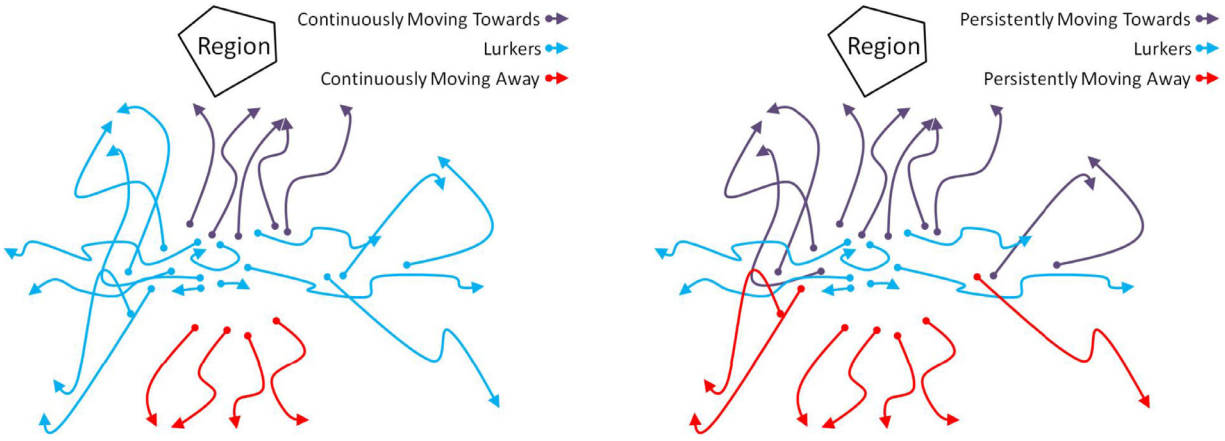
Before proceeding with the details of the detection methodology, we identify certain issues which arise naturally in scenarios involving multiple objects tracking. Namely, we note that accurately maintaining the identities of individual objects when multiple objects are being tracked is computationally intensive, as identified in the literature (cf. [14]) – except for certain results focusing on counting the number of objects [20]. In the sequel, we assume that the nodes are equipped with multiple sensors and able to perform location estimation via sensor fusion, similar in spirit to the [4], and we capitalize on the same principal election algorithm for tracking a particular object mentioned in the previous section.

Our main idea behind flow-like predicates pertaining to multiple objects'motion trends is to improve the utilization of the network resources, while balancing the latency of the detection – via in-network aggregation. More specifically, we observe that for reasonably large values of the total number of objects of interest for a given predicate, it is very unlikely that a particular sensor – be it a simple presence/proximity detecting one or a tracking principal – is the one to detect all κ of them and be able to directly notify the sink. This observation, in turn, implies that with a brute-force approach, one would have to wait for the sink to gather all the individual detections of the instances of single *CMT* or *PMT* predicates and count their total in order to detect the occurrence of CMT_{mult} and PMT_{mult}. To do so, every detection of an individual sensor would need to initiate a route toward the sink (e.g., in a TBF-like manner [45], using shortest path) as in Section 3. However, (re)transmitting those individual detections via multiple hops may put a heavier burden than necessary on the energy expenditures – even more so due to reusing some nodes as intermediate hops for multiple routes. Thus, a multi-layer efficient architecture that enables data aggregation is eminent.

Given the asynchronous nature of predicates, it is unlikely that all the individual notifications can arrive simultaneously at the sink – or, for that matter, that the sink can handle simultaneous transmissions regarding multiple notifications about *CMT* and/or *PMT* detections by different tracking principals. Hence, the sink needs to establish some *time-window* γ, during which the aggregation can be considered valid.

The crux of our proposed aggregation method is that it may be the case that more than one notifications will be using same (portions of a particular) route. For this case, the objective is to utilize some form of data reduction through aggregation, whenever possible, thereby eliminating multiple messages to be aggregated exclusively at the sink.

Before we explore in detail the different ways of merging individual CMT and/or PMT notifications, we note that merger node



(a) Continuously Towards vs. Away (b) Persistently Towards vs. Away

Fig. 3.3. Inverse-motion predicates.

(aggregator) accumulates the numerical values corresponding to the number of objects satisfying moving towards predicates (κ_i) coming from multiple sources. Subsequently, it may adjust the value of $\Sigma_i \kappa_i$ in the case that *chronicle* consumption context is used for the primitive *(location, time)* events (cf. Section 3.1) and the delay-tolerance is large enough to enable double-counting of the same object.

In the rest of this section, we separately discuss two approaches for aggregating the individual detections of the *CMT* and *PMT* predicates: static cluster-based approach as a baseline and data-centric routing with a Bayesian model where detections related to individual objects are merged along the process of their propagation towards the sink. These two approaches illustrate the impact of the two complementary facets of a trade-off between overall energy savings vs. latency (i.e., "freshness" of the data at the sink) with regards to number of objects. Experiments prove that our proposed approach beats the baseline in terms of latency and communication expenditure as the number of objects in the field is greater than 10.

4.1. Baseline – static clusters

Motivated by [24,27], we form clusters based on the geographic area of interest. Namely, we partition the region by means of a $n \times n$ grid such that the nodes inside each cell correspond to one cluster.

Every cluster has an associated cluster head, and the cluster heads form a routing tree among themselves, which we call cluster heads tree (*CHT*). The root of *CHT* is the cluster head that is closest to the sink. This tree is formed (statically) as part of the "wake-up" protocol of the WSN, as soon as the nodes are deployed and become operational. The protocol for routing messages within the WSN from any source towards the sink is relatively straightforward:

1. The source nodes in a cluster send data to their cluster head.
2. Each cluster head sends a message to its parent in *CHT*.
3. The previous step is repeated until the root of *CHT* is reached.
4. The root of *CHT* sends a message to sink.

Although partitioning the whole area of interest into clusters is independent of the sink location, the formation of the *CHT* depends on the sink location. Upon completion of the geographic partitioning, every node knows its corresponding cluster head, i.e., its parent in the routing hierarchy. Queries are disseminated from the sink to all nodes through the network containing all the parameters for the single-object scenario (i.e., t_b, t_e, R and Δ)– for both moving towards and away predicates. However, the cluster heads retain additional parameters such as k and p. In addition to those, cluster heads have aggregation parameter which governs the "freshness" limit γ.

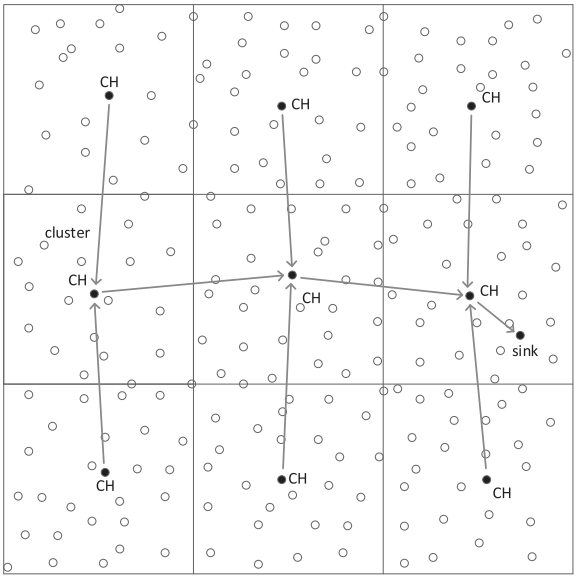


Fig. 4.1. Clusters and CHT. Label "CH" denotes a cluster head.

As soon as a given cluster head receives the query parameters (say for CMT_{mult}) (R, k, p, Δ, t_b, t_e), and γ, it determines its role in terms of the *CHT* and then starts acting accordingly in terms of aggregation and transmission of the aggregated data. We note that a particular cluster head may still serve as a tracking principal for detecting occurrence of *CMT* and/or *PMT* predicates pertaining to different objects.

An example of the *CHT*-based routing scheme is illustrated in Fig. 4.1. The location of the sink at the east portion of the monitored geographic area determines the root of CHT, and the routing tree setup is illustrated by the arrowed edges. We note that the cluster heads may also change over time (cf. [27]), however, the detailed analysis of such impacts is beyond the scope of this paper.

We reiterate that each individual tracking principal will apply the algorithm for detecting *CMT* or *PMT* predicate, meanwhile, they will also detect the reverse of queried *CMT*s or *PMT*s–which

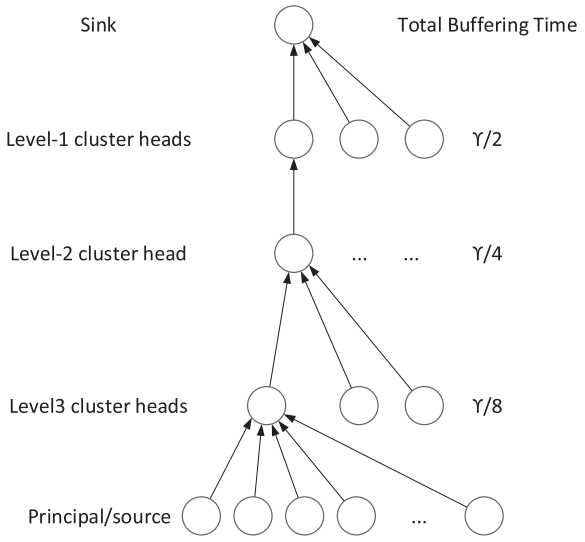


Fig. 4.2. Routing Tree.

is moving away. However, instead of routing the data towards the sink, it will send the data to its cluster head for the grid-cell/cluster that where it is located.

A cluster head node, in turn, is in charge of data aggregation in a manner that could cater to the objective of detecting respective CMT_{mult} or PMT_{mult}, hence reducing the data to be transmitted towards the sink. *CHT* simply gathers all the instances of the predicates'occurrences (as well as the instances of reverse occurrence of objects) from the children, and calculates whether the desired p % threshold has been reached while number of objects are greater than κ.

In addition to the sheer routing information regarding the *CHT*, every cluster head receives an afore-mentioned additional parameter which sets the freshness of the data at a given level of the *CHT*, relative to the freshness tolerance parameter γ of the sink. Without the γ parameter, messages can be buffered in the network for an extended period time and achieve higher energy savings obviously. However, this may render the notifications useless in the context of event-driven paradigm. The messages pertaining to detections of *CMT* and/or *PMT* predicates are asynchronous in nature, since they depend on the motion of individual objects that are being tracked. Hence, upon receiving an initial notification from one of its children (tracking principals) a given cluster, similarly to the sink, waits for a while for other children or tracking principals to send data that could possibly be merged in a single message towards the sink. Given the value of parameter γ for the total delay acceptable in the sink, the wait time on each level of the *CHT* is progressively smaller.

To achieve upper-bounded delay, we set the buffering time at each consecutive level below the root to be decreased as a geometric progression with a common ratio 1/2. Since $\lim \sum_{i=1}^{\infty} \frac{\gamma}{2^i} = \gamma$, this ensures that the delay at the sink will be within the bound γ. An illustration is provided in Fig. 4.2.

The manner in which a given cluster head aggregates the data coming from its children – be it internal nodes of *CHT* (i.e., lower-level cluster heads) or tracking principals – is to simply compose one larger message consisting of all the notifications regarding individual *CMT* or *PMT* detections. Clearly, if a particular CH does not receive any messages, it does not trigger any action, and its parent in the routing structure will be aware of that simply by the expiration of the time-limit without having received a message. One may observe that a situation like this may occur when a given CH malfunctions or dies – however, for that we assume separate messages that check the availability/liveness, which is outside the scope of this work but has been addressed in WSN context [51].

This approach is formalized in Algorithm 5, where PCH represents the parent cluster head in the routing hierarchy and γ_i represents the total wait time at the i-th level ($\gamma_i = \gamma/2^i$).

Algorithm 5 Data aggregation.

Require: γ_i, Parent CH
Ensure: Successful aggregation
1: **if** Receive a message **then**
2: **while** Time-waited $< \gamma_i$ **do**
3: **if** Receive another message **then**
4: Merge notifications per queried CMT_{mult} or PMT_{mult}
5: **end if**
6: **end while**
7: Send merged data to Parent CH
8: **end if**

Lastly, standard deviation in the energy levels across the network will be high for a protocol with static clusters, since cluster heads tend to have more communication/computation duties. To alleviate this effect, rotating cluster heads or dynamically changing clusters have been proposed [27] at the cost of reconfiguration overhead. For our experiments, we give the baseline advantage by keeping them as static clusters.

4.2. Dynamic (data-centric) routing infrastructure

Even though the baseline offers benefits for reducing communication cost (consequently, energy consumption) due to aggregating data in cluster heads, they incur another kind of overhead – the *delay*. Since some applications may need to receive as prompt notifications as possible, the delay introduced by static routing hinders the whole system performance. To address this, we capitalize on another routing scheme that will still perform forms of aggregation but will decrease the time delay of notifications to the sink.

We note that any type of aggregation in WSN will incur some delay since > 1 nodes will have to convey information to the aggregator node. What we are trying to achieve with our variation of data-centric routing, i.e. aggregating-on-the-fly scheme, is to provide a light-weight distributed protocol that will enable every node to decide whether it should or should not declare itself as an aggregator. The main motivation for this approach is based on the following observations:

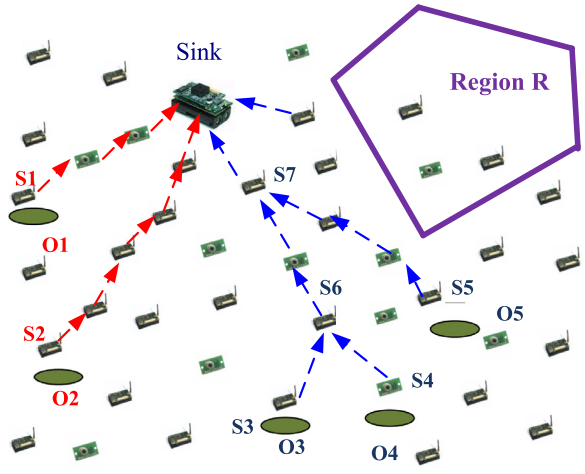


Fig. 4.3. Dynamic aggregation of Predicates' Detection.

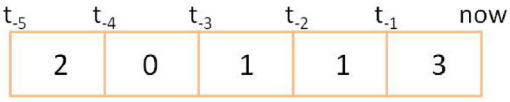


Fig. 4.4. Bag of messages.

Table 4.1
Prior estimates.

$Y = 0$	$Y = 1$
0.2432869	0.7567131

1. Any tracking principal who has detected an occurrence of the *CMT* and/or *PMT* (or their reverse) would initiate a multi-hop transmission towards the sink;
2. Having multiple objects the motions of which are tracked, it may be the case that more than one routes from different principals may merge in some nodes along the respective shortest paths towards the sink;

The observations are illustrated in Fig. 4.3 which shows a detection of the predicates (*CMT* and/or *PMT*) for five objects (O_1, ..., O_5) by five distinct sensors (S_1, ... S_5). Each sensor will initiate its own shortest path based routing towards the sink, notifying it of the detection of the motion trend. The actual routes mapping the respective shortest path to actual multi-hop routing sequences for S_1 and S_2 do not intersect. However, in the case of detections done by S_3 and S_4, they can merge the notifications at S_6 to calculate partial *flow* predicate. Furthermore, that partial information can be augmented at S_7 which is where the multi-hop route of S_5 is intersecting with the ones of S_3 and S_4.

There are many practical application-scenarios where the above properties exploited for data-centric routing, for instance, a herd of animals moving closely to each other. From a complementary perspective, sometimes the sparsity of the deployment may dictate that multiple routes towards the sink need to be merged.

To take advantage of these observations, we need to establish the "behavioral policy" of each node in the network, in the sense of delaying and buffering decisions. In other words, since each node may become an aggregator, it needs to adhere to some criterion regarding how long a particular message related to a notification can be buffered, waiting to be merged with a message pertaining to a different instance of the notification. On the other hand, if every node tries to buffer every message it is relaying, then subset of packages will reach the sink with substantial delay. This makes the buffering decision critical. Although, there are many data-centric routing schemes, none of them aligns with our problem setup, because our event sources are asynchronous and semi-consistent in terms of location. Even source locations has dependency in the time domain since one of the objectives of tracking principals is to minimize the hand-off.

Towards that, there can be several techniques to determine the buffering strategy:

- Buffering every packet.
- Defining a heuristics to selectively buffering packets.
- Building a statistical model to predict whether to buffer or not upon reception of a message.

We decided to employ a Naïve Bayes classifier on the nodes to model buffering decision problem, because it can adapt to different scenarios, computationally inexpensive for the nodes and effective.

Naïve Bayes Classifier

Naïve Bayes classifiers are supervised learning algorithms based on Bayesian reasoning [44]. The classifier tries to approximate the function $f : X \rightarrow Y$ or $P(Y|X)$, where Y is the desired label and X is (a set of) input random vector(s). Applying Bayes rule on this function gives us:

$$P(Y = y_i|X = x_k) = \frac{P(x_k|y_i)P(y_i)}{\sum_j P(x_k|y_j)P(y_j)}$$

where y_i denotes one of the possible outcomes and x_k denotes an arbitrary input vector. In plain terms, estimating the probability of output being the class y_i is equal to multiplication of the prior probability of that class, likelihood of observing x_k given y_i, divided by the evidence probability, which is the marginal probability of observing x_k. One way to estimate these probabilities – consecutively the posterior probability of an output class, given evidence – is to collect training data and use the distribution of the training data.

To estimate likelihood probabilities, the amount of data needed for training and the amount of storage to hold likelihoods is exponential on the number of features. For example, if our feature vector consists of n variables, then in order to estimate $P(X|Y)$, we need to store 2^n parameters. Given that wireless sensor network nodes have very limited capacity, this approach is

Table 4.2
Likelihoods estimates.

	Number of received messages					
	0	1	2	3	4	5
				$[t_5 - t_4]$		
$Y = 0$	0.4166164053	0.5331724970	0.0461399276	0.0037696019	0.0001507841	0.0001507841
$Y = 1$	0.3084157456	0.5762555749	0.0970040721	0.0158522397	0.0019875897	0.0004847780
				$[t_4 - t_3]$		
$Y = 0$	0.4140530760	0.5389022919	0.0435765983	0.0031664656	0.0003015682	0.0000000000
$Y = 1$	0.2787473337	0.6022881520	0.1004944735	0.0159491953	0.0020360675	0.0004847780
				$[t_3 - t_2]$		
$Y = 0$	0.4010856454	0.5550361882	0.0405609168	0.0031664656	0.0000000000	0.0001507841
$Y = 1$	0.2490304441	0.6277389955	0.1041787861	0.0164339732	0.0021330231	0.0004847780
				$[t_2 - t_1]$		
$Y = 0$	0.353136309	0.609620024	0.035132690	0.002110977	0.000000000	0.000000000
$Y = 1$	0.209084739	0.671272057	0.100445996	0.016579407	0.002133023	0.000484778
				$[t_1 - t_0]$		
$Y = 0$	-	0.9330518697	0.0637816647	0.0030156815	0.0000000000	0.0001507841
$Y = 1$	-	0.8682373473	0.1118867559	0.0172580958	0.0021330231	0.0004847780

not feasible. Therefore, we make the naïve assumption of conditional independence. Given the learning objective $P(Y|X)$, where $X = X_1, X_2, \ldots, X_n$, we assume all X_i's are independent of each other given Y as follows:

$$P(X|Y = y) = P(X_1, X_2, \ldots X_n|Y = y) = P(X_1|y)P(X_2|y)\ldots P(X_n|y)$$

What the second equation gives is that each sensor now needs to hold 2^*n estimations for the likelihoods.

Now that we set up estimation mechanism for the components of the model, the next challenge becomes choosing the right feature vector X. We explored several avenues for feature selection and based on the information gain they provided, we decided to apply, what we call, *bag of messages* approach. Each node discretizes the time and keeps track of the messages it received in each epoch for the past 25 s. Then, based on the history, it calculates $P(Y = \textit{should buffer}\ |X)$ and if the probability is ≥ 0.5 it buffers the message it received.

Our feature vector splits the past 25 s into 5-s chunks and holds a vector for number of message arrivals in each interval. An example can be seen in Fig. 4.4, where t_i indicates the timestamp from $now + 5 * i$. 5 s is the sampling interval – epoch duration – in our experimental setups. We defined each feature as categorical variables taking values between 0 and 5. If number of messages in any epoch exceeds 5, then it is binned to 5. In other words, if a node receives 6, 8, or 10 messages in an epoch – however unlikely –, it treats them as 5 messages. In addition, we explored the direction of the incoming messages as features. We discretized the space to slices with 60° angles and counted the number of messages coming from each direction as in *bag of messages* approach, but these features did not provide enough information gain since messages follow a greedy shortest path towards the sink and their reception angles overlap most of the time.

To train our model, we ran 3 simulations with 10-, 20- and 50-objects settings for data gathering, then estimated model parameters based on more than 40,000 data points we collected. Our model achieved 73.1% accuracy. Learned parameters are on Tables 4.1 and 4.2.

The likelihoods for 0 messages received in the last epoch does not exist since not having any messages will not initiate the model because there will be no message to make buffering decision on. Also, we do not include evidence probabilities for brevity and since

$$\{P(Y = 1|X) > 0.5\} = \{P(X|Y = 1)P(Y = 1) > P(X|Y = 0)P(0)\}.$$

{.} represents true-false function, where true means *should buffer* decision and false means *relay* decision.

Finally, we define the buffering time for intermediate nodes. In order to ensure responsiveness, we set the buffering time as 1 epoch duration plus some ϵ to account for transmission delay. Thus, the delay is, formally, equal to *samplingInterval* + *pseudoHops* | *samplingInterval* seconds.

The *samplingInterval* variable denotes the value of the current sampling interval (epoch) of the nodes participating in tracking. The *pseudoHops* denotes the hop-distance from a given node to the sink, and it is approximated by dividing distance between the location of that node and location of the sink by the radius of the communication range. Therefore, the aggregator will wait at most the sampling interval plus some $\varepsilon = \textit{pseudoHops}\ |\ \textit{samplingInterval}$ between any two messages. As can be seen, ε decreases as the node location gets towards the sink, since it is more important to aggregate the data in earlier steps of the transmission in order to achieve higher communication savings.

5. Experimental evaluation

The experimental observations were generated using SIDnet-SWANS, our open-source simulator for WSN [21]. We considered a WSN consisting of 750 (varying for multiple object settings) 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. To alleviate the lack of spatio-temporal dependency among consecutive motion-segments present in the random way-point model, we used trajectories based on the *Gauss-Markov Mobility Model* (GMMM) [10,39], which does exhibit spatial and temporal dependency. In the GMMM model, at each time-slot, the speed and direction are computed based on the ones from the previous time-slot. Throughout our experiments, we used three different categories of speeds of motion, corresponding to walking, riding a bicycle, and driving a car. Finally, the sensing field size is 1500 m by 1500 m.

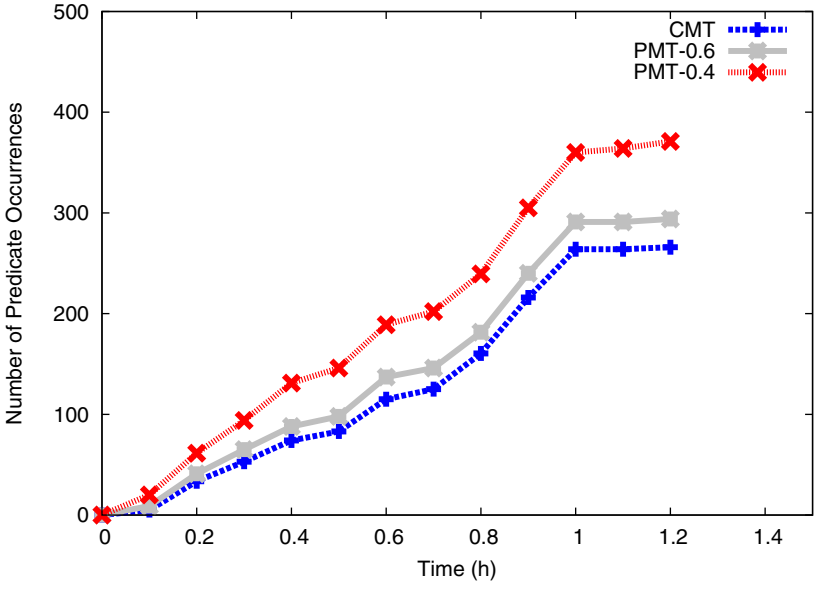


Fig. 5.1. Continuous vs. persistent motions.

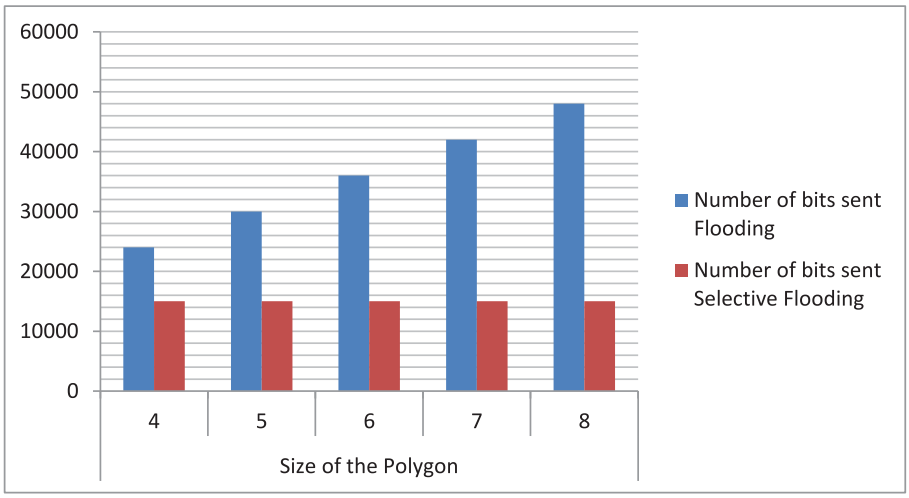


Fig. 5.2. Bits transmitted for request dissemination.

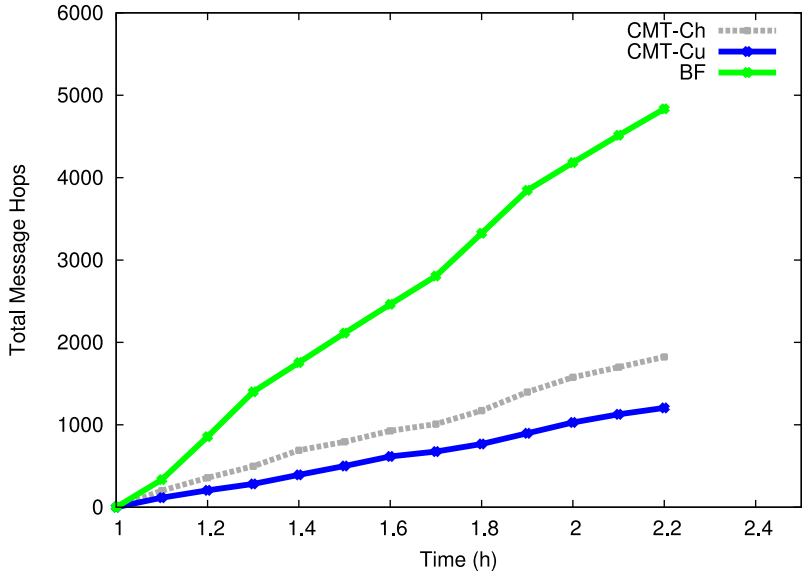


Fig. 5.3. Detecting *Continuously Moving Towards*.

We report on two distinct categories of experimental observations: one pertaining to tracking and detecting the predicates' occurrences for a single object, and the other pertaining to tracking and detecting the predicates' occurrences for multiple objects. We note that, in reality, the quality of links in a WSN may vary and in order to eliminate any "bias" for one methodology or another based merely on the network status, we report as the relative measure the total number of message-hops when comparing different approaches. Besides, we eliminate communication expenses related to overhead incurred by tracking principal algorithm to ensure fairness. Also, message queue for cluster heads are set to 25 to emulate message congestion. Therefore, when message queue is full, a retry is issued by the sender.

5.1. Single-object tracking

The first group of experiments aims at providing quantitative observations as to why we introduced the two variants, *CMT* and *PMT*. Namely, in security-related scenarios, an attacker may periodically change the direction of motion, while still maintaining a level of "persistency" towards a particular region. By varying the parameter Θ, the system can detect different numbers of occurrences of the predicate capturing the motion trend. Fig. 5.1 shows how a particular object has more detections of the event of interest for smaller values of Θ (*PMT* corresponds to $\Theta = 0$) and increasing at a faster rate over time. We set the total time towards region as 15 s for this set of experiments and sampling interval is 5 s.

Our next group of experiments aims at illustrating the benefits of our three-phase protocol for disseminating the requests via selective flooding. As shown in Fig. 5.2, the total number of bits transmitted between the pairs of nodes in the network grows linearly with the size (number of vertices) of the polygon bounding the query region, *R*. On the other hand, using the proposed selective flooding—which guarantees that the nodes in the WSN will be able to correctly process the request for detecting the *CMT* or *PMT* predicate—the total number of bits transmitted is almost a constant. Both observations are consistent with the intuitive expectations.

Our next set of simulations aimed at providing some quantitative observations regarding the savings obtained when using our Algorithm 1 for processing the *CMT* predicate, when compared to the centralized approach of transmitting location samplings by tracking principals to the sink, and performing trend detection on the dataset. The results are illustrated in Fig. 5.3,

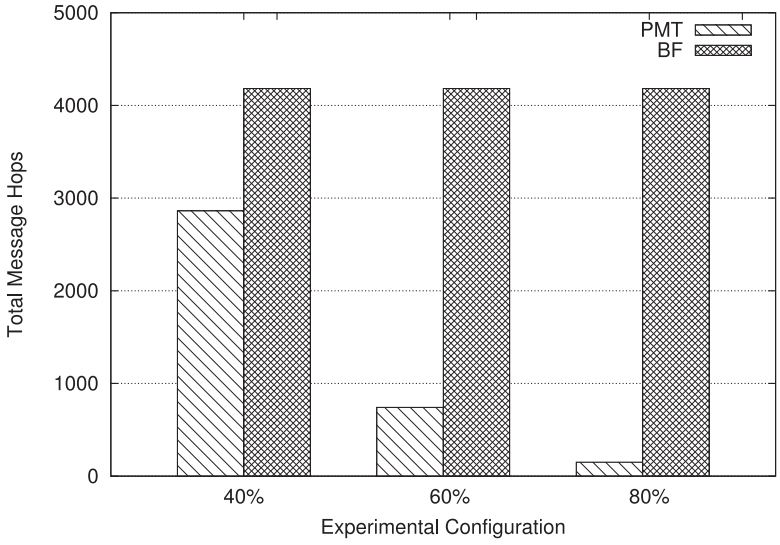


Fig. 5.4. Impact of Θ *Persistently Moving Towards*.

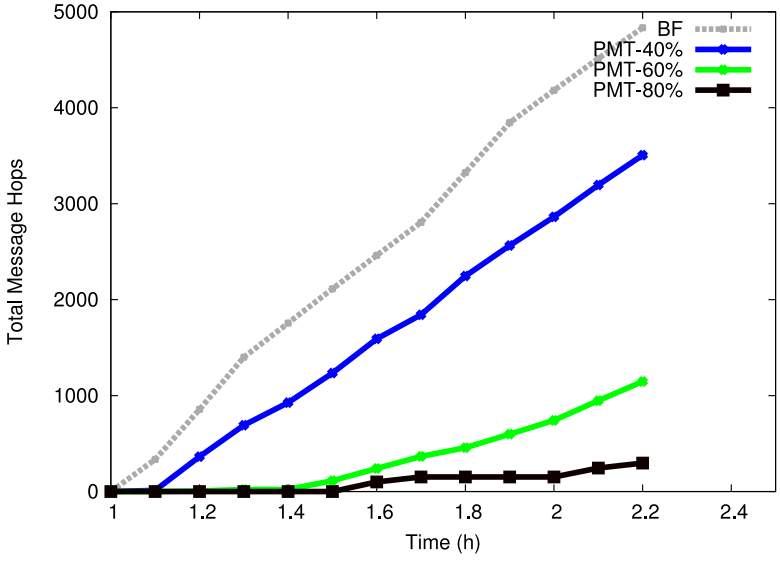


Fig. 5.5. Detecting *Persistently Moving Towards*.

which shows averaged observations for sampling intervals of 5 s, 10 s and 30 s; and the criteria of $\Delta = 3 \cdot SamplingInterval$ and $\Delta = 5 \cdot SamplingInterval$ needed to satisfy the *CMT* predicate.

As can be seen, the centralized approach (denoted "BF" in Fig. 5.3) generates much higher volume of messages than the *CMT* approach. Note that there are two curves, one for each of the consumption policies (Ch – Chronical, and Cu – Cumulative) corresponding to the in-network *CMT* processing. As expected, the Ch–consumption will generate more messages towards the sink, since it "recycles" all the former (location, time) pairs, except for the oldest initiator of a detection; whereas the Cu-consumption completely eliminates the history upon detection of the predicate.

When it comes to the *PMT* predicate, recall that there are two temporal values of interest: (1) the duration of the observational time-window Δ; and (2) the fraction Θ of that time-window, for which we would like the tracked object to be moving towards the region *R*. Fig. 5.4 illustrates the difference in the message traffic generated by the centralized (BF) approach and our approach for in-network processing of the *PMT* predicate, based on Algorithm 3.

The results show three scenarios, each of which has a fixed value of the $\Delta = 10min$, and Θ is varied between 40%, 60% and 80%. Firstly, observe that in each case, our approach generates substantially fewer messages than the BF one. Secondly, the number of messages decreases when Θ increases – which is to be expected. Namely, smaller values of Θ are likely to generate more instances that satisfy the *PMT* predicate.

Our last set of experiments for this section illustrates the benefits of our Algorithm 3 for processing the *PMT* predicate when compared to the centralized approach, over time. Specifically, we show the settings in which the duration of the sliding time-window is $\Delta = 10$ min. Fig. 5.5 depicts the corresponding curves for the values of $\Theta = 0.4\Delta$, $\Theta = 0.6\Delta$ and $\Theta = 0.8\Delta$, along with the corresponding one for the centralized approach (once again labelled "BF"). As can be seen, the centralized approach incurs a significant overhead in terms of the messages traffic in the network, much larger than our algorithms. Once again, as expected, the smaller values of Θ enable more frequent satisfiability of the *PMT* predicate which, in turn, will generate more messages towards the sink.

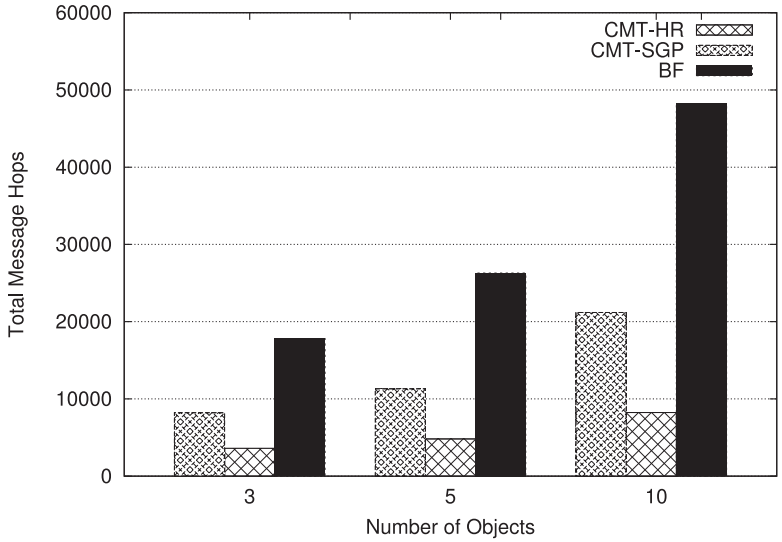


Fig. 5.6. Communication with Cluster Based Routing.

5.2. Multiple objects

We now show the evaluation of the benefits and trade-offs of our proposed aggregation versus the baseline for routing when detecting the motion trends for multiple objects. The same experimental setup has been used for this section and all objects move independent of each other.

5.2.1. Baseline

Three different scenarios are applied to quantify the impact of aggregation in cluster heads of the static clustering scheme. In the first experimental setup, naive approach is used where tracking principals update the sink with whereabouts of the object in every sampling interval using shortest geographic path routing (SGP, or TBF) scheme. In the second setup, we applied CMT-Ch (Continuously Moving Towards - Chronical) predicate where each tracking principal was independently using the shortest geographic path algorithm to report a detection of the *CMT* predicate to the sink, using the chronical consumption policy. Lastly, we evaluated hierarchical routing scheme with CMT-Ch predicate, with freshness parameter, γ, of 100 s. For all settings, results show the averaged values for sampling intervals of 5 s, 10 s and 30 s, and for the settings, in which CMT-Ch is exercised, $\Delta = 3 \cdot SamplingInterval$ parameter is used to satisfy the predicate.

The total number of message hops sent for each settings is illustrated in Fig. 5.6, where CMT-SPG denotes principals send messages via SGP when CMT is detected, CMT-HR denotes the cluster-based approach, and BF denotes the naive approach – where all predicate detection messages are sent to the sink individually. We present the observations for 3, 5 and 10 objects. As shown, when it comes to multiple objects, the hierarchical clustering scheme does yield significant communication savings – 2–3 times less overhead than independent shortest path routing and 3–5 times than the centralized approach.

Fig. 5.7 illustrates our next set of observations, which is, how the partition of the geographic region in grid-cells when forming clusters affects the performance of the static hierarchical routing. The purpose of this experiment is to validate the effectiveness of the baseline. As shown, the larger the number of clusters (or, the smaller the area), the larger the communication expenses. This is due to the fact that the increase of the granularity of partitioning yields a routing tree with more inner-layers. In addition, we observe (as expected) that the costs are increasing with the increase of number of objects.

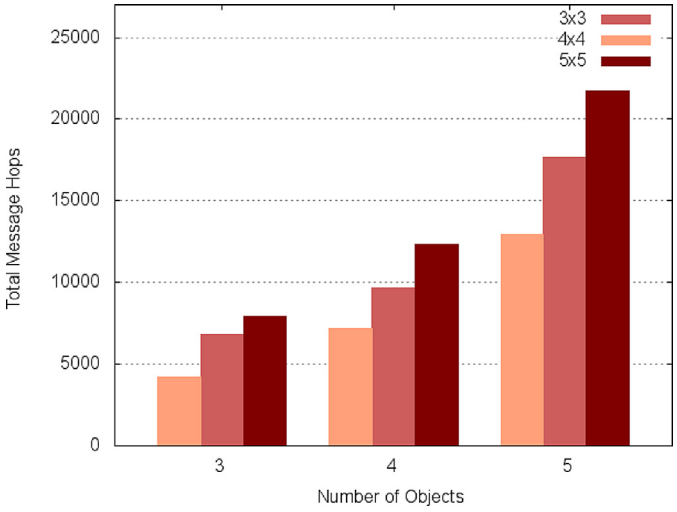


Fig. 5.7. Impact of Cluster/Grid Size.

5.2.2. Dynamic data-centric routing

Recall that the proposed dynamic aggregation scheme relies on merging of the notifications “on the fly” in the relay nodes from reporting tracking principals towards the sink based on a statistical model. Due to this, to ensure better validity of the observations, we evaluated its performance against the baseline approach in the settings with 800 nodes and varying number of objects.

We make the general observation that the dynamic combination of routing and aggregation yields substantially smaller communication and energy savings than cluster-based hierarchical routing if the number of objects are relatively small, since freshness parameter can be adjusted to reduce communication overhead by sacrificing occurrence notification delay. Fig. 5.8 shows the total message hops for both baseline (hierarchical) and data-centric techniques. In addition, a log-linear regression has been fit to both data. As can be seen after 8–10 objects, data-centric scheme starts to be more appealing since this technique thrives on merging paths, which is very common as the number of objects increases. As expected, static cluster-based approach shows a linear increase

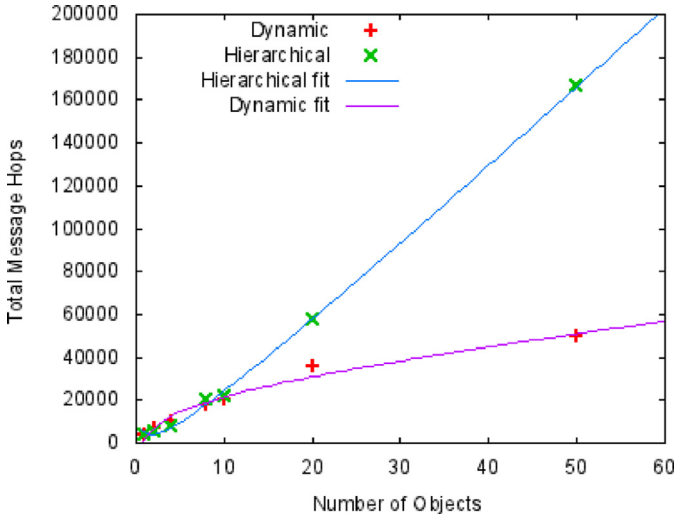


Fig. 5.8. Transmission overhead per number of objects.

over time. Note that, these experiments were done with freshness parameter, γ, set to 100 s for static case, and 30 s to refresh USL for data-centric case. Also the results show the total message hop costs for 15 min with 5 s sampling interval and *CMT-Cu* as underlying motion trends protocol with $\Delta = 5 \cdot SamplingInterval$.

Although the dynamic routing consumes more energy (or, yields less savings when compared to the "independent" shortest path based routing by individual principals) for fewer number of objects than the baseline, it yields substantially smaller delays. On the contrary, as the number of objects increases, greater than 15, freshness parameter should also increase for cluster-based approach in order to reduce communication expenditure when compared to data-centric approach. Otherwise, for large number of objects data-centric scheme saves more transmission cost and augments less delay.

A more direct comparison of the two routing and aggregation methodologies in terms of delays is presented in Fig. 5.9, which compares the values of the average delays and the maximum delays (each averaged across the different setups in terms of network size and number of tracked objects), where HR denotes cluster-based hierarchical routing scheme and DR denotes dynamic data-centric routing scheme.

Table 5.1
Load Balance.

Approach\Number of Objects	1	4	20	50
Static	0.76	1.14	2.49	4.09
Data-centric	0.63	1.38	2.3	2.58

Finally, we analyze nodes'energy distribution in the network for the two approaches. Our performance metric is network-wide standard deviation of the remaining energy level percentage of the nodes. After 2 h of experimentation, Table 5.1 outlines the standard deviation difference between the approaches. Note that, the more the simulation runs the wider the gap will grow. Because message paths tend to be very similar with few number of objects while having many objects would distribute the network load more evenly.

6. Related work

Most research on tracking in sensor networks has concentrated on the movement of object in the free, with some more recent works on tracking objects in transportation network. Among the works in free space we can distinguish a number of approaches: tree-based, topological (geometric) and predication-based [8,11]. We can also classify the various methods based upon the types of queries supported: find queries that request the location of a particular object, range queries requesting the number of detected objects inside a given area and pattern queries, concerned with detecting certain pattern of multiple target movement.

The Drain-and-Balance (DAB) [34] was the first in-network object tracking approach to make use of tree structure in order to avoid query flooding and sending update messages directly to the sink. However, the logical tree of [34] does not consider the physical structure of the sensor network. The tree structure proposed in [40] is addressing these concerns by keeping at each node information about of the all the objects in the subtree rooted at that node. Object location updates are triggering messages that travel along this tree structure. Query flooding is avoided by using the tree structure in a top-down search. It is assumed that mobility information is available about the frequency of objects crossing between sensor nodes and this information is used to compute the

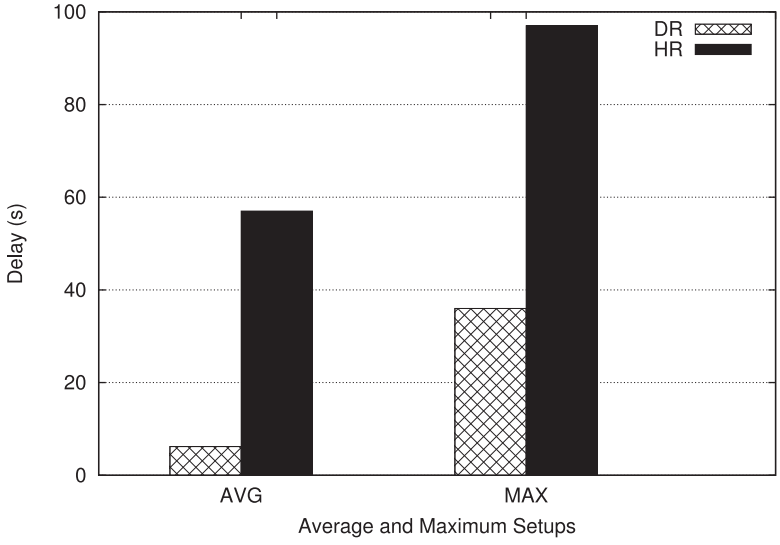


Fig. 5.9. Relative comparison of delays.

weights of the edges in the graph G representing the Delaunay triangulation of the sensor network. The DAT algorithm presented in [40] computes a tracking tree which minimizes the hop count in the graph G. While most of the work on maintaining tree tracking structures assume that the sink is located in the center or at a corner of the wireless sensor network, this is not necessarily the optimal location for it. The algorithm proposed in [15] introduces a tree structure where the sink is selected using a center of gravity policy.

In the category of geometrical approaches we find Trail [33] which maintains for every detected object a tracking structure consisting of a list of segments from the detected object to the center of the network. The structure is maintained in a distance sensitive fashion: moves of the mobile target cause only partial updates in the tracking trail. The larger the move is the greater the number of segments that need to be modified. Trail supports queries that request the current location of an object by sending explore messages that travel along circles of increasing radii until they intersect the corresponding tracking structure. Geographic routing is then used to forward the query from the intersection point to the current location of the target by following the child pointers in the trail.

A unique geometrical approach is presented in [47] which allows for tracking of targets which are individually identifiable as well as those without identifiers. This method maintains a function on the edges of the sensor network that is a co-vector field with respect to the target detection data. This function, denoted as a differential one-form, has the property that its integral along a curve C corresponds to the integral of the region bounded by C. When targets move from one face to an adjacent one the method requires only updating the differential one-form for the edge common to these faces. The differential forms support different type of range queries, such as find requests or range queries. Finding a target with a known identity starting from a given position P is performed by using the differential form maintained for the target and issuing successive range queries for larger boxes centered at P. Computing the number of targets in a given region is done by summing up the values of the differential one-forms on all the faces comprising the region.

The approaches discussed so far track multiple objects by supporting queries that count the number of objects in a range [47] or by keeping track of the object identifiers in the various nodes of the tracking tree [40]. Many applications require tracking of patterns of movement associated with a group of objects [3], for example finding leadership patterns or detecting flocks of objects. A flock (n, k, p) is defined as a set of n objects which for a time period of k consecutive time units are all contained within a disk of radius p. The first decentralized algorithm for flock detection was reported in [35]. A different pattern of object movement is described in [3] where the concept of leadership is defined. A moving target is considered to be a leader within a time interval if it doesn't follow any other targets during this interval and is followed by a sufficiently large number of objects. A distributed algorithm for detecting leadership is given in [3], where an object o_j is said to follow object o_i if o_j belongs to a region to the front of o_i. A variation of the flocking algorithm is introduced in [48], where the semi-flocking pattern is introduced in order to strike a balance between robust area coverage and target coverage.

A geometric approach to the problem of tracking a group pattern is taken in [62] where arbitrary patterns can be detected by tracking the contour of the sensor readings above a given threshold. Although this problem can be reduced to that of finding the holes in a network [19], in a dynamic environment where the targets move this is a rather inefficient approach. Instead, [62] presents a light weight algorithm which requires recomputation of a contour network constructed at time t using only local neighborhood information of the places where the contour brakes at some future time $t + d$.

The category of prediction-based methods present integrated approaches for handling efficiently sleep scheduling and target tracking [11,31]. Most of the proposed methods fall into the category of kinematics-based methods [31] that consider the motion of vehicles that move in a smooth curvilinear trajectory without abrupt direction changes. One of the first schemes presented is Prediction-based Energy Scheme (PSE) [60] which attempts to predict the future movements of an object so that only the sensor nodes expected to discover the object at the next reporting period need to stay active. The assumption is that the target positions need to be reported every T seconds to a sink by the alarm node that detects the target. PSE presents some simple models to predict the location of the destination alarm node as well as a number of heuristics to determine which nodes need to be awaken in its neighborhood. MCTA [28] uses also vehicular kinematics in order to restrict the set of awaken nodes during a given time interval to those that the target can visit based on its current position, speed, direction as well as the vehicle turning time.

The paper by [31] presents an elaborate protocol, called PPSS, which schedules the sleep patterns of the awakened nodes individually according to their distance and direction away from the current motion state of the target. Compared to MCTA, PPSS reduces further the number of nodes to be awakened during a time period by combining a kinematics-based prediction step with a probability-based prediction step which computes the scalar displacement and the deviation (polar angle). While the above schemes are limited to single target detection, the algorithm introduced in [30] presents an energy saving scheme that is aimed at multiple target prediction by taking into account the overlapping areas of various alarm nodes broadcasts.

Geometrical approaches have been used in conjunction with prediction-based methods. The DOT protocol presented in [54] uses face routing in order to achieve power savings: only the nodes adjacent to the beacon node in the current face need to be awaken during the target detection phase. The tracking process includes a second phase in which a mobile agent (the source) is directed towards the current beacon, i.e., the sensor node that is now closest to the target object. If the mobile source arrives at the location of the current beacon and the target is still there the process stops, i.e., the source catches the object. On the other hand, if the target has moved, the source is directed towards the next beacon node in the trail and this process continues until the sources catches the object.

As discussed earlier, most number of prediction-based methods are intended for vehicular movement, but they do not consider explicitly the underlying transportation network. A number of recent works have considered the problem of tracking moving objects in a transportation network [9,18]. Sensors capable of detecting moving objects are embedded in a number of fixed locations in the transportation network, called checkpoints. The transportation network and the checkpoints induce a connectivity graph which shows which checkpoints are directly accessible from which other checkpoints. The tracking information is stored in local tables maintained by the checkpoint nodes. In order to track the direction of movement it is assumed that a method is available to record at a given checkpoint the origin and destination of each moving object being sensed. The local tables store at each checkpoint the movement events that occur on the edges of the connectivity graph incident to that checkpoint. However, these tracking algorithms assume that all the checkpoints that are neighbors in the connectivity graph are also connected in the communication graph. In the absence of this property, it is necessary to rely on data mules to physically move information towards a given sink [9].

7. Concluding remarks and future work

We have introduced two novel spatio-temporal predicates capturing motion trends of a moving object with respect to a polygonal region. While in both predicates the spatial motion-trend property of interest is for the object to move *towards* a given region, their main difference is in the temporal dimension. Predicate *Continuously Moving Towards* does not tolerate temporary changes of the motion (e.g., from "towards" to "away") within the time-interval of interest. Instead, predicate *Persistently Moving Towards* is more flexible in terms of direction-changes. Nevertheless, it presumes some lower-bound on the changes of direction to preserve the direction trend in the temporal dimension, which can be accumulated in a discontinuous manner during the object's motion. In addition to'moving towards'predicates, one can define/query the inverse variants of the same predicates – interpreted as'moving away'. Their implementation is a straightforward extension to our proposed algorithms.

We have focused on the efficient detection of the satisfaction of the two predicates at a given sink of the WSN, where the detection of the location of the moving object in a given time instant is done by collaborative trilateration of the tracking sensors. We have presented efficient distributed algorithms for in-network detection of the predicates as part of the tracking process, where the sink is only signaled a notification upon their occurrence. We have also discussed policies for processing primitive (*location*, *time*) events upon detection of the composite event defining the *CMT* and *PMT* predicates.

Our experiments have demonstrated that the proposed algorithms bring substantial savings in terms of reducing the number of messages that need to be communicated throughout the network, when compared to the naïve approach which transmits every detected location (along with its time-stamp) to the sink. Our experiments have also shown that the communication overhead for establishing the necessary partitioning of the network in terms of the Voronoi cells of the polygonal regions of interest is not significant.

We have also extended our approach to the concurrent detection of motion trends for multiple objects within a given time-interval of interest. We have proposed two approaches for data aggregation—static (cluster-based) and dynamic—and we have experimentally evaluated the trade-offs between energy efficiency and latency of detection at the sink.

As part of future work, we plan to further investigate how the semantics of the problem domain and the expected quality of service may affect the trade-offs between energy efficiency of the communication, data savings, and latency.

Currently, we are extending our approaches in two directions: (1) we would like to propose an efficient scheme for handling multiple queries – both in terms of the number of regions of interests, as well as the predicates. For example, when considering CMT_{mult}, if there is also a predicate CMA_{mult} processed in-network, a particular aggregator node may combine the results of the two counters in order to further speed-up the respective detections. (2) we also plan to handle the detection of motion trends for groups of objects moving closely together, as in the case of a *flock* of trajectories (see, e.g., [55]). There are few other challenges that we would like to address in the future as extensions of this work. First, we plan to modify the existing algorithms so that the epoch-based synchronized operation of the nodes can be taken into account, along with the corresponding policies for selecting tracking principals. Next, we would like to investigate the impact of having heterogeneous network settings where, in addition to static nodes, there are also mobile nodes [38].

Another problem to work on in the future is to make the area of interest a moving region rather than a static one, and even further extend the settings to incorporate deformable mobile shapes. This extension will make our work suitable for a broader set of real-life applications such as tracking tornados or any other object which does not have a constant shape over time.

Finally, we are planning to investigate different aspects of collaboration among static and mobile nodes when detecting motion trends for multiple objects. To this end, we are planning to build upon some recent findings in variants of the pursuer-evader problem in WSN [23,58]

References

[1] K. Akkaya, M. Younis, A survey on routing protocols for wireless sensor networks, Ad Hoc Netw. 3 (3) (2005).
[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (4) (2002).
[3] M. Andersson, J. Gudmundsson, P. Laube, T. Wolle, Reporting leaders and followers among trajectories of moving point objects, Geoinformatica 12 (4) (2008) 497–528, doi:10.1007/s10707-007-0037-9.
[4] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, M. Miyashita, A line in the sand: a wireless sensor network for target detection, classification, and tracking, Comput. Netw. (Elsevier 46 (2004) 605–634.
[5] F. Aurenhammer, Voronoi diagrams - a survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (3) (1991).
[6] B.A. Bash, P. Desnoyers, Exact distributed Voronoi cell computation in sensor networks, in: IPSN, 2007, pp. 236–243.
[7] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer-Verlag New York, Inc., 2001.
[8] S. Bhatti, J. Xu, Survey of target tracking protocols using wireless sensor network, in: Proceedings of the 2009 5th International Conference on Wireless and Mobile Communications, in: ICWMC '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 110–115, doi:10.1109/ICWMC.2009.25.
[9] A. Both, M. Duckham, P. Laube, T. Wark, J. Yeoman, Decentralized monitoring of moving objects in a transportation network augmented with checkpoints, Comput. J. 56 (12) (2013) 1432–1449, doi:10.1093/comjnl/bxs117.
[10] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Commun. Mobile Comput. 2 (5) (2002).
[11] Z. Can, M. Demirbas, A survey on in-network querying and tracking services for wireless sensor networks, Ad Hoc Netw. 11 (1) (2013) 596–610, doi:10.1016/j.adhoc.2012.08.007.
[12] Q. Cao, T. Yan, J. Stankovic, T. Abdelzaher, Analysis of target detection performance for wireless sensor networks, in: DCOSS, 2005, pp. 276–292.
[13] S. Chakravarthy, V. Krishnaprasad, E. Anwar, S. Kim, Composite events for active databases: semantics, contexts and detection, in: 20th VLDB Conference, 1994.
[14] L. Chen, N. Tokuda, An efficient computational approach for multitarget tracking from bearings-only measurements by decentralized cooperative processing, Chapter 7, 2013, pp. 147–170.
[15] M. Chen, Y. Wang, An efficient location tracking structure for wireless sensor networks, Comput. Commun. 32 (13–14) (2009) 1495–1504, doi:10.1016/j.comcom.2009.05.005.
[16] W. Chen, J. Hou, L. Sha, Dynamic clustering for acoustic target tracking in wireless sensor networks, in: IEEE International Conference on Network Protocols (ICNP'03), 2003.
[17] I. Demirkol, C. Ersoy, F. Alagöz, MAC protocols for wireless sensor networks: asurvey, IEEE Commun. Mag. 44 (4) (2006) 115–121.
[18] M. Duckham, Decentralized Spatial Computing, Springer-Verlag, 2013.
[19] Q. Fang, J. Gao, L.J. Guibas, Locating and bypassing holes in sensor networks, MONET 11 (2) (2006) 187–200.
[20] Q. Fang, F. Zhao, L. Guibas, Lightweight sensing and communication protocols for target enumeration and aggregation, in: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, in: MobiHoc '03, ACM, New York, NY, USA, 2003, pp. 165–176, doi:10.1145/778415.778436.
[21] O. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischoff, N. Valtchanov, Sidnet-swans: a simulator and integrated development platform for sensor networks applications, in: SenSys, 2008, pp. 385–386.
[22] O. Ghica, G. Trajcevski, F. Zhou, R. Tamassia, P. Scheuermann, Selecting tracking principals with epoch-awareness, in: Proceedings of the 18th ACM SIGSPATIAL GIS Conference, 2010, pp. 222–231.
[23] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, E. Kohler, The tenet architecture for tiered sensor networks, in: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, in: SenSys '06, ACM, New York, NY, USA, 2006, pp. 153–166, doi:10.1145/1182807.1182823.
[24] M. Handy, M. Haase, D. Timmermann, Low energy adaptive clustering hierarchy with deterministic cluster-head selection, in: in IEEE MWCN, 2002, pp. 368–372.
[25] C. Hartung, R. Han, C. Seielstad, S. Holbrook, Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments, MobiSys, 2006.

[26] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J.A. Stankovic, T.F. Abdelzaher, Achieving real-time target tracking using wireless sensor networks, IEEE Real Time Technology and Applications Symposium, 2006.
[27] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8, in: HICSS '00, IEEE Computer Society, Washington, DC, USA, 2000. 8020–
[28] J. Jeong, T. Hwang, T. He, D. Du, Mcta: target tracking algorithm based on minimal contour in wireless sensor networks, 2007.
[29] J. Jeong, T. Hwang, T. He, D.H.-C. Du, Mcta: target tracking algorithm based on minimal contour in wireless sensor networks, INFOCOM, 2007.
[30] B. Jiang, B. Ravindran, H. Cho, Energy efficient sleep scheduling in sensor networks for multiple target tracking, in: Distributed Computing in Sensor Systems, 4th IEEE International Conference, DCOSS 2008, Santorini Island, Greece, June 11–14, 2008, Proceedings, 2008, pp. 498–509.
[31] B. Jiang, B. Ravindran, H. Cho, Probability-based prediction and sleep scheduling for energy-efficient target tracking in sensor networks, IEEE Trans. Mobile Comput. 12 (4) (2013) 735–747, doi:10.1109/TMC.2012.44.
[32] S. Kim, S. Pakzad, D.E. Culler, J. Demmel, G. Fenves, S. Glaser, M. Turon, Health monitoring of civil infrastructures using wireless sensor networks, in: IPSN, 2007, pp. 254–263.
[33] V. Kulathumani, A. Arora, M. Sridharan, M. Demirbas, Trail: a distance-sensitive sensor network service for distributed object tracking, TOSN 5 (2) (2009), doi:10.1145/1498915.1498921.
[34] H.T. Kung, D. Vlah, Efficient location tracking using sensor networks, in: 2003 IEEE Wireless Communications and Networking, WCNC 2003, New Orleans, LA, USA, 16–20 March, 2003, 2003, pp. 1954–1961, doi:10.1109/WCNC.2003.1200686.
[35] P. Laube, M. Duckham, T. Wolle, Decentralized movement pattern detection amongst mobile geosensor nodes, in: Proceedings of the 5th International Conference on Geographic Information Science, in: GIScience '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 199–216.
[36] L. Lazos, R. Poovendran, J.A. Ritcey, Analytic evaluation of target detection in heterogeneous wireless sensor networks, TOSN 5 (2) (2009).
[37] S. Lee, R.M. Muhammad, C. Kim, A leader election algorithm within candidates on ad hoc mobile networks, ICESS, 2007.
[38] X. Li, J. Yang, A. Nayak, I. Stojmenovic, Localized geographic routing to a mobile sink with guaranteed delivery in sensor networks, IEEE J. Sel. Areas Commun. 30 (9) (2012) 1719–1729.
[39] B. Liang, Z.J. Haas, Predictive distance-based mobility management for multidimensional pcs networks, IEEE/ACM Trans. Netw. 11 (5) (2003) 718–732.
[40] C.-Y. Lin, W.-C. Peng, Y.-C. Tseng, Efficient in-network moving object tracking in wireless sensor networks, IEEE Trans. Mobile Comput. 5 (8) (2006) 1044–1056, doi:10.1109/TMC.2006.115.
[41] W. Liu, Y. Zheng, S. Chawla, J. Yuan, X. Xing, Discovering spatio-temporal causal interactions in traffic data streams, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '11, ACM, New York, NY, USA, 2011, pp. 1010–1018, doi:10.1145/2020408.2020571.
[42] G. Mao, B. Fidan, Localization Algorithms and Strategies for Wireless Sensor Networks, IGI Global – Invormation Science Publishing, 2009.
[43] M.M.A. Mohamed, A.A. Khokhar, G. Trajcevski, Voronoi trees for hierarchical in-network data and space abstractions in wireless sensor networks, in: 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '13, Barcelona, Spain, November 3–8, 2013, 2013, pp. 207–210.
[44] K.P. Murphy, Naive Bayes Classifiers, 2006. Accessed: 06/10/2016.
[45] D. Niculesu, B. Nath, Trajectory based forwarding and its applications, MOBICOM, 2003.
[46] S. Pattem, S. Poduri, B. Krishnamachari, Energy-quality tradeoffs for target tracking in wireless sensor networks, IPSN, 2003.
[47] R. Sarkar, J. Gao, Differential forms for target tracking and aggregate queries in distributed networks, in: Proceedings of the 16th Annual International Conference on Mobile Computing and Networking, in: MobiCom '10, ACM, New York, NY, USA, 2010, pp. 377–388, doi:10.1145/1859995.1860038.
[48] S.H. Semnani, O.A. Basir, Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance systems, IEEE Trans. Cybern. 45 (1) (2015) 129–137, doi:10.1109/TCYB.2014.2328659.
[49] B. Sundararaman, U. Buy, A.D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey, Ad Hoc Netw. 3 (3) (2005) 281–323.
[50] R. Szewczyk, A.M. Mainwaring, J. Polastre, J. Anderson, D.E. Culler, An analysis of a large scale habitat monitoring application, in: SenSys, 2004, pp. 214–226.
[51] B. Tong, G. Wang, W. Zhang, C. Wang, Node reclamation and replacement for long-lived sensor networks, in: SECON, 2009, pp. 1–9.
[52] G. Trajcevski, B. Avci, F. Zhou, R. Tamassia, P. Scheuermann, L. Miller, A. Barber, Motion trends detection in wireless sensor networks, MDM, 2012.
[53] G. Trajcevski, P. Scheuermann, H. Brönnimann, A. Voisard, Dynamic topological predicates and notifications in moving objects databases, Mobile Data Management (MDM), 2005.
[54] H.-W. Tsai, C.-P. Chu, T.-S. Chen, Mobile object tracking in wireless sensor networks, Comput. Commun. 30 (8) (2007) 1811–1825, doi:10.1016/j.comcom.2007.02.018.
[55] M.R. Vieira, P. Bakalov, V.J. Tsotras, On-line discovery of flock patterns in spatio-temporal data, in: GIS, 2009, pp. 286–295.
[56] D. Wang, An energy-efficient clusterhead assignment scheme for hierarchical wireless sensor networks, IJWIN 15 (2) (2008) 61–71.
[57] H. Wang, K. Yao, D. Estrin, Information-theoretic approaches for sensor selection and placement for target localization and tracking in sensor networks, JCN 7 (4) (2005) 438–449.
[58] Y.-C. Wang, F.-J. Wu, Y.-C. Tseng, Mobility management algorithms and applications for mobile sensor networks, Wireless Commun. Mobile Comput. 12 (1) (2012).
[59] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, J. Lees, Deploying a wireless sensor network on an active volcano, IEEE Internet Comput. 10 (2) (2006).
[60] Y. Xu, J. Winter, W. Lee, Prediction-based strategies for energy saving in object tracking sensor networks, in: 5th IEEE International Conference on Mobile Data Management (MDM 2004), 19–22 January 2004, Berkeley, CA, USA, 2004, pp. 346–357, doi:10.1109/MDM.2004.1263084.
[61] L. Yang, C. Feng, J.W. Rozenblit, H. Qiao, Adaptive tracking in distributed wireless sensor networks, ECBS, 2006.
[62] X. Zhu, R. Sarkar, J. Gao, J.S.B. Mitchell, Light-weight contour tracking in wireless sensor networks, in: INFOCOM, 2008, pp. 1175–1183.