J. Parallel Distrib. Comput. 71 (2011) 460–470

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Distributed network control for mobile multi-modal wireless sensor networks

Doina Bein [a], Yicheng Wen [b], Shashi Phoha [a,*], Bharat B. Madan [a], Asok Ray [b]

[a] Applied Research Laboratory, The Pennsylvania State University, University Park, PA 16802-5018, USA
[b] Department of Mechanical Engineering, The Pennsylvania State University, University Park, PA 16802-1412, USA

ARTICLE INFO

Article history:
Received 18 March 2010
Received in revised form
25 August 2010
Accepted 27 August 2010
Available online 27 September 2010

Keywords:
Distributed sensor networks
Multi-modal data fusion
Mobile wireless network
Target localization
Target tracking

ABSTRACT

A sensor network operates on an infrastructure of sensing, computation, and communication, through which it perceives the evolution of events it observes. We propose a fusion-driven distributed dynamic network controller, called MDSTC, for a multi-modal sensor network that incorporates distributed computation for in-situ assessment, prognosis, and optimal reorganization of constrained resources to achieve high quality multi-modal data fusion. For arbitrarily deployed sensors, a certain level of data quality cannot be guaranteed in sparse regions. MDSTC reallocates resources to sparse regions; reallocation of network resources in this manner is motivated by the fact that an increased density of sensor nodes in a region of interest leads to better quality data and enriches the network resilience. Simulation results in NS-2 show the effectiveness of the proposed MDSTC.[1]

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

A distributed sensor network consists of many low-cost mobile or fixed sensor nodes that are networked to observe spatial-temporal distributed events. Observations, decisions or estimates made by individual sensors are fused together to create a composite view of the situation. The sensor nodes must dynamically collaborate to fuse information in the vicinity of an arbitrary critical event to optimize its identification, classification, and tracking in operational environments. The sensor nodes and the network they form are resource-constrained in terms of network bandwidth, the lifetime of the node's battery, sensing range and communication radius. Consequently, it is important to dynamically control the available resources to deliver acceptable sensor fusion performance and possibly to maximize this performance, measured in terms of data accuracy for fusion. The techniques described combine distributed control and self-adaptation of the network topology to the requirements of the application running at the highest level at each sensor node, and can be applied to solve many problems in military, homeland security, medicine, civil engineering, bathymetry, terrain mapping, etc. Previous research efforts [3] focused on human-controlled or human-guided sensor networks. In this paper we consider networks that are autonomous; there is no human intervention involved in designing or organizing the network.

Heterogeneous sensor networks have been extensively studied for the purpose of classifying and tracking mobile targets, by obtaining an accurate estimation of the sensed event(s) locations based on the sensors' locations [27], measured signal strength, and the quality of the raw data [5,16,25,14]. When a multi-modal sensor network is used to track moving targets, the critical resource optimization parameters are power consumption, coverage, communication, and data fusion. Major sources of wasted energy are communication collisions, overhearing, idle listening, and control overhead [32]. Idle listening can be minimized by an appropriate sleep-and-awake schedule for nodes [33,4,10,26]. Collisions can be minimized by scheduling the local transmission [1] and overhearing can be minimized by reducing the transmission power [12]. To optimize the region coverage, various fault tolerant schemes for the network topology are researched in order to guarantee that a certain area is covered [34,7,18,28]. To optimize communication, the network capacity is to be maximized and the network delay is to be minimized [17]. To optimize data fusion, interference is to be reduced by scheduling nodes' transmission [11], higher fidelity of raw data is to be obtained by moving sensors closer to observed event(s) [18], high fidelity compressed data is to be obtained from the raw data [6], and the delivery rate of fused data is to be increased [31].

When a mobile target approaches a sensor node and the stimulus at the node is above a given threshold, an event is said to have been detected. However, if the threshold is not reached then the

* Corresponding author.
E-mail address: pab33@psu.edu (S. Phoha).

[1] This material is based upon work supported by the US Army Research Laboratory and the US Army Research Office under the eSensIF MURI Award No. W911NF-07-1-0376. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsor.

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.08.016

event is labeled as undetected and ignored. When an event is detected by a sensor node, the node broadcasts a message containing the event description and the event data. Since the events are assumed to be localized in time and space, a burst of communication will occur in a small geographic region surrounding the target. So, instead of analyzing the topology of the entire network, we concentrate on the topology of a relatively small geographical region around the mobile target where nodes that are geographically near observe an event of interest and form multi-modal (heterogeneous) clusters that together support a specified level of multi-modal data fusion. We present MDSTC, a distributed network controller for a multi-modal sensor network, responsible for clustering nodes in the space–time vicinity of a moving object, improving data fusion by adapting the network topology (i.e., changing the location of selected mobile nodes and the transmission range of selected nodes), localizing and tracking the object. MDSTC is an adaptation of the dynamic space–time clustering (DSTC) protocol [8,19–21] to the sensors' heterogeneity and mobility; it has the enhanced capability to coordinate mobile nodes within a cluster to move to nearby locations in order to lower interference, avoid field obstructions, and cover sparse regions. Thus the topology of the cluster self-adapts to achieve better data fusion.

During the data fusion (necessary for tracking moving events), two modes of intra-cluster communication are supported: between the same modality sensors and between sensors of complementary modalities. In order to minimize the local network communication that leads to reduced power consumption and network bandwidth, we also describe a novel multicast group management protocol that handles multicast groups between same-modality and complementary sensors.

The paper is organized as follows. In Section 2 we present the network architecture, the system model, and the communication optimization through multicast. The multi-modal data fusion at the nodes observing an event of interest is presented in Section 3. In Section 4 we present the multi-modal cluster formation and the dynamic adaptation of the multi-modal cluster to improve data fusion. In Section 5 we show how the proposed multi-modal network controller improves the quality of the multi-modal data fusion and we present the results of the experiments conducted in our laboratory. Section 6 presents conclusions and future work.

2. Network architecture and system model

An event is a dynamical process that is observed by sensors of various modalities. The time-series data observed by a sensor of a certain modality can be compressed to a probabilistic finite state automaton (PFSA) with an a priori fixed alphabet [24], which depends strictly on the modality of the sensor. The PFSA is designed to extract the maximum semantic information from the raw sensor data, while at the same time it compresses the sensory data to minimize the network bandwidth requirements. At each node the data is handled in a module called Information Space and the network communication issues and adaptation decisions are addressed by a separate module called Network Design Space [22,23]; the system model at a node is presented in Fig. 1.

DSTC [8] protocol works for single-modality wireless networks and it has five basic operations: creating a cluster, disbanding a cluster, adding a node to a cluster, removing a node from a cluster, and moving the data from the cluster head to another cluster member. We adapted DSTC to multi-modal mobile sensors and we call the new algorithm the *multi-modal dynamic space–time clustering* (MDSTC). MDSTC is part of NDS and is responsible for clustering nodes in the space–time vicinity of a moving object, improving data fusion, adapting the network topology by moving mobile sensors to dynamically computed locations in order to cover sparse regions and lower communication interference, localizing and tracking the object that has triggered the composite pattern.

As DSTC, MDSTC allocates the sensor network resources only in the space–time vicinity of an event or object, resources that belong to multi-modal, resource-constrained, mobile sensor nodes. It also dynamically alters the topology of each cluster formed around an event by allocating and de-allocating resources (i.e., moving sensors in or out of a cluster) in order to maintain a specified level of fusion performance. The space–time vicinity of an event [8,20,9] is defined as the set of all space–time points that are close to the space–time point (x_0, t_0) within a time and space boundary:

$$\eta(x_0, t_0, \Delta x, \Delta t) = \{(x, t) | \|x - x_0\| \leq \Delta x \wedge \|t - t_0\| \leq \Delta t\}. \quad (1)$$

The quantities Δx and Δt define the size of the neighborhood. Multi-modal data that was measured within a distance Δx around x_0 and within the time interval $t_0 \pm \Delta t$ will be used for data fusion and the sensors within this neighborhood will be considered for a multi-modal space–time cluster.

A distributed network of sensors achieves high performance by exploiting sensor data fusion. Sensors of various modalities group together to form dynamic multi-modal clusters; their sensed data (of various modalities) is fused locally at each node. The clustering of multi-modal sensors is done in two stages.

Stages 1 and 2 of data fusion entail the following type of communication. In Stage 1, *same-modality* sensor nodes located within the sensing radius of the sensed event need to inform one another about the minimum *semantic distance* between the PFSA observed by a sensor node and some reference PFSA from the library. The minimum semantic distance argument translates into the best match argument between an observed PFSA pattern and the set of reference patterns. Thus, during Stage 1, the sensor nodes need only to communicate with like-modality sensors.

Stage 2 focuses on constructing the composite PFSA that has the best possible composite metric. The composite PFSA is made up of PFSA from all sensors of complementary sensing modalities. Consequently, during Stage 2 only communication between sensors of different sensing modalities is needed. In the sequel, we describe a novel multicast grouping algorithm that can substantially reduce the communication among sensors, as well as computation at nodes.

Let $\mathcal{S}$ be the set of sensors in a given region and let $k = |\mathcal{S}|$. For any given sensing modality s, the set $\mathcal{S}$ can be partitioned into $\mathcal{S}_s$, the subset of sensors that have the same sensing modality s, and $\mathcal{S}_{\neg s} = \mathcal{S} \setminus \mathcal{S}_s$, the subset of nodes with the sensing modality complementary to s; let $k_s = |\mathcal{S}_s|$.

During Stage 1, all nodes in $\mathcal{S}$ need to broadcast their semantic distance among themselves; the message will be processed by an application entity present in the application layer of the protocol stack of each node. However, this broadcast message will have relevance only for the $|k_s|$ ($\ll |k|$) sensor nodes. The application entities of the remaining $|k - k_s|$ sensors of complementary sensing modality will however, receive, process and eventually discard such messages, resulting in unnecessary processing cycles and consumption of limited battery power.

Stage 2 deals with the formation of composite PFSA at each node of some modality s, which are made up of individual PFSA provided by sensors of $\neg s$ sensing modalities in addition to the PFSA constructed by the node. Therefore in Stage 2, a node from $\mathcal{S}_{\neg s}$ of sensing modality $s' \neq s$ needs to send its PFSA only to a node of sensing modality s; this is the converse of Stage 1. After a node $i \in \mathcal{S}_{\neg s}$ broadcasts its PFSA to all the nodes in $\mathcal{S}$, such a message will be relevant to only the nodes in $\mathcal{S}_s$; the remaining $k - k_s - 1$ nodes will waste their processing cycles in receiving and then discarding such a message. The number of wasted processing cycles can be quite substantial, since a packet meant for an application has to traverse all the protocol layers, before it is accepted or thrown

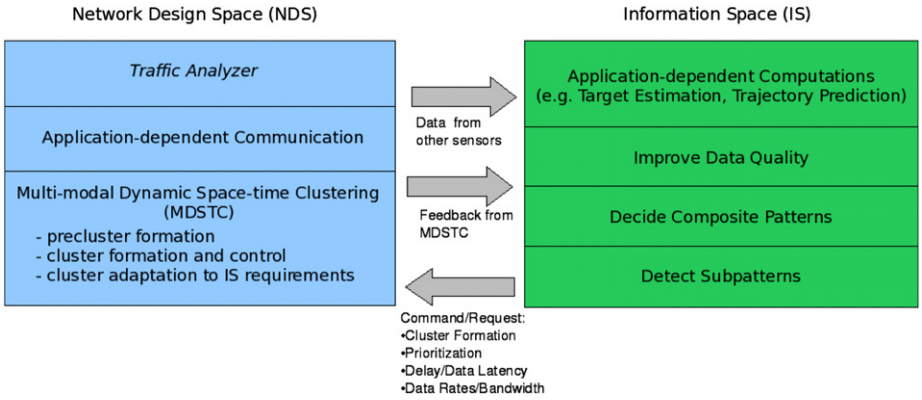


Fig. 1. System model for a sensor node (independent of the sensor modality).

away by the application layer. Without multicasting, the amount of CPU cycles used by a single sensor node for receiving and subsequently rejecting a packet is equal to $C_{\text{physical layer}}+C_{\text{MAC layer}}+C_{\text{link layer}}+C_{\text{network layer}}$. With multicasting, in comparison the number of CPU cycles wasted by a single sensor node for receiving and subsequently rejecting a packet is equal to $C_{\text{physical layer}}+C_{\text{MAC layer}}$, since an IP multicast address gets embedded within the MAC address assigned to the network interface. Consequently, packets whose destination address does not correspond either to the unicast address or to one of the multicast addresses (joined by this node) are filtered out by the MAC layer itself, thus saving CPU cycles and energy for not having to process the packet in the link and network layers.

Let us consider a fully connected sensor network with *A*coustic (A), *V*isual (V), and *C*hemical (C) sensors, n of each modality. During Stage 1, an A-modality sensor needs to communicate with the rest of the A-modality sensors only to exchange the semantic distance metric information. This implies that the remaining $2n$ sensors waste their processing cycles when A-modality broadcast packets traverse their protocol stacks. During Stage 2, the selected cluster head (let us say an A-modality sensor) needs to communicate only with the non-A modality sensors, since the purpose of this communication is to create a composite PFSA. This implies that, when V- and C-modality sensors broadcast their PFSA, only the A-modality cluster head has to take cognizance of these broadcast packets. A naive processing of the broadcast packets would require remaining the $3n-2$ nodes to process and then discard such packets. To address this waste, we describe a novel multicast group formation technique that replaces the broadcast with the multicast, thus eliminating the waste described earlier. It turns out that the minimum number of multicast groups required to support Stages 1 and 2 is independent of the number of sensor nodes, and it depends only on the number of modalities (Lemma 1).

Lemma 1. *Given a sensor network with n nodes and m sensing modalities, the minimum number of distinct multicast groups required to support Stages* 1 *and* 2 *is* $2m$*. Out of these* $2m$ *groups, each sensor node needs to join only* $m+1$ *groups.*

Proof. During Stage 1, at least one node of some modality X needs to send its PFSA to sensors of the same modality and it does so by joining a group G_X. Thus the minimum number of multicast groups to be created during Stage 1 is m since each modality is represented by at least one group.

During Stage 2, a node of some modality w needs to send its PFSA to the sensor nodes of non-X modalities and it does so by joining another group $G_{\neg X}$ that contains sensors of modalities other than X. Also, all nodes of non-X modalities will also need to join the $G_{\neg X}$ group to be able to receive a PFSA from a X-modality sensor. Thus we need to create m additional groups in order to support Stage 2. In all, we require a minimum of $2m$ distinct multicast groups. □

Table 1
Multicast groups for multi-modal data fusion.

(a) Groups for Stage 1				(b) Groups for Stage 2			
Modality	A	C	V	Modality	A	C	V
A	G_A			A		$G_{\neg C}$	$G_{\neg V}$
C		G_C		C	$G_{\neg A}$		$G_{\neg V}$
V			G_V	V	$G_{\neg A}$	$G_{\neg C}$	

As an example, for the 3-modality network described earlier six multicast groups, G_A, G_C, G_V, $G_{\neg A}$, $G_{\neg C}$, and $G_{\neg V}$, need to be formed. During Stage 1, a sensor node of modality i needs to communicate with sensors of the same modality. This communication takes place on a multicast group G_i (see Table 1(a)). During Stage 2, a sensor node of modality i needs to communicate with sensor nodes of modality j ($j \neq i$). This communication occurs on the multicast group given by the entry (j, i) (of row j and column i) of the Table 1(b).

Each sensor of modality A will have to communicate using four multicast groups, G_A, $G_{\neg A}$, $G_{\neg C}$, and $G_{\neg V}$, in order to form a composite pattern of interest. In general, when there are m modalities, each sensor needs to use $m+1$ multicast groups out of a total of $2m$ multicast groups that need to be created, which is independent of the number of the sensors.

3. Information space—multi-modal data fusion

When a mobile target approaches a sensor node and the stimulus at the node is above a given threshold, an event is said to have been detected. However, if the threshold is not reached then the event is labeled as undetected and ignored. When an event is detected by a sensor node, the node broadcasts a message containing the event data. The time-series data observed by a sensor of a certain modality can be compressed to a probabilistic finite state automaton (PFSA) over an a priori fixed alphabet [24]. The PFSA is designed to extract the maximum semantic information from the raw sensor data, while at the same time it compresses the sensory data to minimize the network bandwidth requirements.

Multi-modal data fusion is done at individual nodes in the Information Space (IS) module. During the training phase we identify and store a number of PFSA of interest, which we call *subpatterns*, in the IS of a sensor node. An example of such a subpattern is the pressure signature of a car passing by a pressure sensor (Section 3.1). In the operational phase, when an event is observed by the sensor, the IS at the sensor node will construct a PFSA based on the time-series data from the sensing unit. The constructed PFSA will be compared against the library of subpatterns and a *semantic distance* [6] will be derived from each of the existent subpatterns. If the constructed PFSA has a semantic distance less than a given ϵ with regards to a particular subpattern,

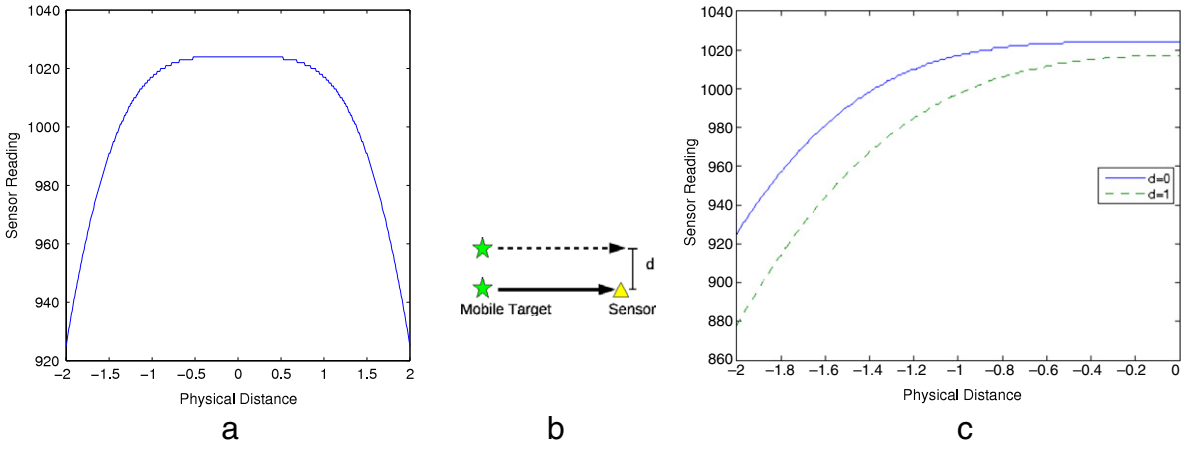


Fig. 2. Semantic distance for pressure sensors.

we conclude that the sensor has detected that subpattern, and we have a *subpattern match*.

After the IS matches a computed PFSA to a specific subpattern, it sends to the network controller the message *subpatternObserved* that contains the subpattern ID and the metric which is the semantic distance between the observed subpattern and the subpattern stored in the databank. The exact time when the message is sent by the IS depends on the sensor modality, since the IS has to collect enough data in order to construct a PFSA. On receiving the message *subpatternObserved* from the IS, MDSTC appends to it some control information such as the node position, node ID, the time, and broadcasts it as the message *Subpattern*. In this way the neighboring nodes are notified about the observed subpattern. The messages *Subpattern* can be used for target localization (see [29]).

Based on the observed subpattern and received subpatterns from other nodes, the IS will try to decide on a composite pattern. Namely, at each node, the data received from sensors of different modalities is fused locally in order to generate patterns of interest which we call *composite patterns*. If successful, the IS notifies MDSTC about the decided composite pattern.

A *composite pattern* is the Cartesian product of subpatterns in each modality. If we have $M > 1$ different modalities, a composite pattern $P = \{P^1, P^2, \ldots, P^M\}$ is a M-tuple of subpatterns, one subpattern for each modality. At each node, the Information Space stores a set of composite patterns of interest, $\mathbb{P}$, in a so-called *databank*. The databank contains, for each stored pattern of interest and for each individual subpattern, the minimum number of sensors of that modality that are required to observe the subpattern $NP_j = (p_{j,1}, p_{j,2}, \ldots, p_{j,M}), p_{j,l} \geq 1, \forall 1 \leq l \leq M$. A pattern of interest has an associated maximum lifetime that represents the period during which the pattern is observable.

3.1. Constructing a subpattern

To construct subpatterns in modality m, a sensing model for modality m is assumed. We show next how subpatterns are constructed for pressure sensors; the pressure sensor model in this case is omni-directional, but this is not a requirement for other modalities. The data reading of a pressure sensor changes as a moving object passes by (Fig. 2(a)); when a mobile object approaches the pressure sensor, different paths of the target determine different time series data for the sensor. Let d be the minimum distance between the object and the sensor at any moment of time (Fig. 2(b)); the solid line corresponds to the subpattern (trained PFSA when $d = 0$), which corresponds to the time series data (solid curve) in Fig. 2(c). The PFSA constructed from the time-series data starting when the target was at distance $d = 1$ m to the sensor is shown as a dotted curve in Fig. 2(b).

When the object is at the same location as the sensor we are guaranteed that we have a perfect subpattern match. This coincides with the concept of *closest point of approach (CPA) events* [2,15,30]. In other terms, when the object passes closest to the sensor node, the sensor's signal reaches a peak and the event is signaled. The captured event is called a *CPA event*. The signal peak of the CPA event is directly related to the distance between the object and the sensor because the signal-to-noise ratio of sensing decreases with the distance. The closer the target is to the sensor, the higher the signal peak of the CPA event is. The difference between two time series data is reflected in the so-called semantic distance ϵ.

For magnetic sensors, the subpatterns are constructed exactly the same way as for the pressure sensors. However, the subpatterns of video sensors are constructed differently and they do not incur CPA events. Therefore the PFSA constructed by the video sensors will only be used for pattern classification but not for estimating the event location.

4. Network design space

MDSTC is part of the Network Design Space (NDS) module; it has has several responsibilities: adjust the transmission power based on the battery power, communication with other nodes, maintain and provide statistics of this communication to the IS module, cluster multi-modal sensor nodes in the space–time vicinity of an event, identify sensor nodes to be included or excluded from the clusters, request sensor nodes to move in order to improve the quality of the data fusion or to cover sparse regions in which events are expected, coordinate the detection of events and correlate events based on spatial-temporal occurrence. The precluster and cluster formation of DSTC are adapted in MDSTC to the heterogeneity of the sensors; this adaptation is straightforward. To improve the quality of data needed for deciding on a composite pattern and application-dependent computations (such as estimating the target location, velocity, and predicting its trajectory), the cluster also adapts its topology to the requirements of the IS (see Section 4.1) by adding selected sensors (Sections 4.2 and 4.4) in specific locations (Section 4.3).

Each multi-modal cluster has an associated maximum lifetime. When the maximum lifetime expires, the cluster is disbanded by the cluster head. Thus all cluster members, including the cluster head, become free. The cluster head can also disband the cluster when its battery power falls below a certain threshold.

When the mobile target moves through the sensor network, a node other than a cluster head may observe the pattern better. Since the cluster head is selected as the node that currently

observes the event the best, we may need make that node as the cluster head. In this case, we may need to disband the current cluster and form another cluster. The local network stability may suffer if we create and disband clusters as soon as nodes observe better than the current cluster heads. We then take a *lazy* approach, to maintain stability: we allow the cluster to live for at least some period of time, that we call *min_lifetime*. During the *min_lifetime*, the cluster head does not change due to a better composite metric for the same composite pattern. Once the *min_lifetime* expires and before *max_lifetime* expires, it is possible for a cluster member or a non-cluster member to become a cluster head and force the current cluster head to disband the cluster.

4.1. Dynamically changing the local topology

We recall that the IS of the node knows a priori the number of nodes of each modality needed to distinguish between subpatterns or composite patterns. Sensors need to be moved in two distinct cases. One case is when the IS of the node determines that it has too little information to properly identify either a uni-modal or a composite pattern of interest, or it needs better quality data for composite data fusion. Another case is when the IS of a node is notified that the trajectory of a mobile target will intersect its sensing coverage region and it has determined that there are not enough sensors of a certain modality within the node's communication region to be able to do data fusion.

In either case, the IS sends the message *moreData* to the MDSTC with the sensor modality needed, the number of extra sensors of that modality needed, and the *desired* region where the sensors are needed (the center and the radius). The MDSTC broadcasts the message *addNode* appended with the node ID, the node's physical location or coordinates, the sensor modality needed, and the desired region. The number of sensors needed of that modality is stored by the MDSTC.

On receiving the message *addNode*:

- If the sensor is of the modality requested by the requester node and the requester is further away from the center of the desired region, then drop the message.
- If the node's modality is not the same as the requested one and the node's location is not far from the desired location, then broadcast the message further.
- If the node's modality is the same as the requested one then the message is sent to the IS for further processing (see Section 4.2) and may agree to participate. We note that the nodes that have agreed to the request have to move within to the requester's transmission range.

Among the sensors that have responded positively to the request, the MDSTC of the requester will decide where they will be placed (see Section 4.3) and which of them will be asked to move (see Section 4.4).

4.2. When a node decides to move

The decision of moving to a specific region is done in the IS. We note that, if the node is not within the communication range of the requester node, the node has to move into the requested region in order to participate in data fusion and receive subsequent messages. If the node is capable of moving and is not part of any cluster, then the IS estimates the cost of moving per unit distance

$$cpud = 1/(P \cdot \rho)$$

based on the node's power P (the more power, the lower the cost) and the *density* ρ, namely how many sensors of the same modality exist within the node's communication range (the lower the density, the higher the cost). This value together with the *speed* which is the average speed the sensor can move and the requester ID is sent as the message *Available* to the MDSTC. When the MDSTC receives the message *Available* from the IS, it appends the node ID and the node position and the message is then sent to the neighbor from which the message *addNode* has been received. On receiving the message *Available* not addressed to itself, a node forwards it to the node from which the message *addNode* has been received. On receiving the message *Available* addressed to itself, the MDSTC of the requester sends it to the IS. The information from the messages *Available* received within the timeout period will be used by the IS of the requester to select a number of sensors and request them to move. The selection is based on (i) the cost per unit distance, (ii) the interval of time necessary to move, computed using the speed of the mobile sensor and its distance, and (iii) the number of sensors that already exist in the region where the sensor will be required to move, which we call *obstruction*.

Let C be the region of interest in which the sensors will be asked to move by a requester node; for simplicity we assume that C is a circle of radius R. Sensors of the desired modality will be placed in region C, which is included in the requester communication region, in order for the requester to be able to form a cluster. Instead of requiring sensors to move arbitrarily to any location in C we propose a uniform placement of them. To this end, we propose two methods in which C is partitioned into K desired regions (see Section 4.3). For each of the available sensors, let us say M in total, we compute the distance between the node and each desired region. In Section 4.4 we present a Greedy algorithm that selects nodes for each of the K regions, using the $M \times K$ such distances.

4.3. Partitioning a region of interest C

We propose two methods of partitioning C into K desired regions:

(i) *Concentric circles*: Region C is partitioned into K equidistant concentric circles (see Fig. 3(a)).
(ii) *Circular sectors*: Region C is divided into K circular sectors (see Fig. 3(b)), of angle $\lceil 360/K \rceil$.

We compute the distance a node (whose modality is needed by the requester node r) has to move in order to reach any of the K partitions of C. For any node u, let X_u be the location of node u.

4.3.1. Concentric circles

Region C is partitioned into K equidistant concentric circles (see Fig. 3(a)). The K concentric circles, $\{C_i | i = 1, \ldots, K\}$, create $K - 1$ rings or annulii,[2] $\{CS_i | i = 1, \ldots, K-1\}$, with the cross-section size R/K; the radius of the circle C_i is $i \cdot \frac{R}{K}$. By abuse of notation, let CS_0 be the circle C_1.

We will compute the distance a node whose modality is needed by the requester r has to move in order to reach any of the K regions. Let w be such a node whose location X_w is outside C. Let $d(X_w; CS_i)$, $0 \le i \le K - 1$, be the distance between X_w and the region CS_i, which is the smallest distance the node w needs to move in order to touch the boundary of CS_i. Then

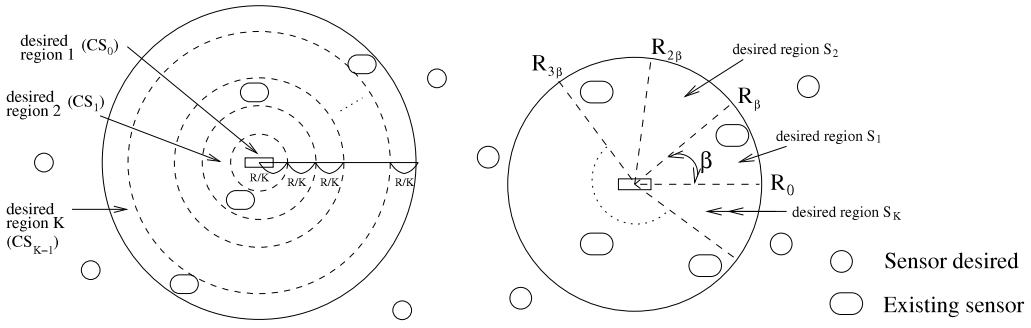
$$d(X_w; CS_i) = X_w - (i + 1)R/K$$

for any i, $0 \le i \le K - 1$.

4.3.2. Circular sectors

We divide C into K equal circular sectors $\{S_i, i = 1, \ldots, K\}$, $C = \cup_{i=1}^{K} S_i$ (see Fig. 3(b)). The central angle of each sector is $\beta = \frac{360}{K}$. For simplicity, we assume that 360 is divisible by K. Each S_i is

[2] An *annulus* is the region between two concentric circles of different radii.



(a) Concentric circles (b) Circular sectors

Fig. 3. Partitioning the communication region of the requester into K desired regions.

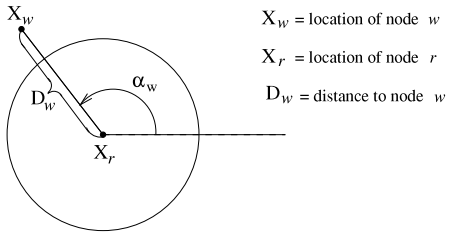


Fig. 4. Node w is outside C and needs to move inside C.

defined by the center X_r and the two radii $R_{(i-1)\cdot\beta}$ and $R_{i\cdot\beta}$: $S_i = (X_r; R_{(i-1)\cdot\beta}; R_{i\cdot\beta})$.[3] Note that two consecutive circular sectors S_j and S_{j+1} share the radius $R_{j\cdot\beta}$.

Consider some node w whose modality is desired by the requester r and whose location X_w is outside C. Let D_w be the distance from X_w and X_r and α_w be the angle under which node w is seen by node r (see Fig. 4).

Let $d(X_w; S_i)$, $1 \leq i \leq K$, be the smallest distance between X_w and any location in the sector S_i, which represents the smallest distance node w needs to move in order to touch the boundary of S_i. To compute $d(X_w; S_i)$, i, $1 \leq i \leq K$, we have five cases.

Case (1) $\alpha_w < (i-1)\beta$ and $(i-1)\beta - \alpha_w < \frac{\pi}{2}$. Let $\gamma = (i-1)\beta - \alpha_w$ (Fig. 5(a)–(b)). We have two subcases:

Case (1.1) $D_w \cos\gamma > R$ (Fig. 5(a)). The closest point in S_i to X_w is the intersection point of C and the radius $R_{(i-1)\cdot\beta}$. The distance node w needs to move in order to reach S_i is

$$d(X_w; S_i) = \sqrt{(D_w \sin\gamma - R)^2 + (D_w \cos\gamma)^2}.$$

Proof. The distance $d(X_w; S_i)$ is the length a of the edge in Fig. 5(a), and $a = \sqrt{b^2 + c^2}$ where $b = D_w \sin\gamma - R$ and $c = D_w \cos\gamma$. □

Case (1.2) $D_w \cos\gamma \leq R$ (Fig. 5(b)). The closest point in S_i to X_w is on the radius $R_{(i-1)\cdot\beta}$. The distance node w needs to move in order to reach S_i is

$$d(X_w; S_i) = D_w \sin\gamma.$$

Case (2) $\alpha_w < (i-1)\beta$ and $(i-1)\beta - \alpha_w \geq \frac{\pi}{2}$ (Fig. 5(c)). The closest point in S_i to X_w is X_r. The distance node w needs to move in order to reach S_i is

$$d(X_w; S_i) = D_w + \epsilon$$

where $\epsilon > 0$ is a value close to 0 and is needed to ensure that the sensor w will not be placed on the same location as r.

[3] We denote by R_α the ray of C at the angle α from the x axis.

Case (3) $(i-1)\beta \leq \alpha_w \leq i\beta$ (Fig. 5(d)). The closest point in S_i to X_w is the intersection point of C and the radius R_{α_w}. The distance node w needs to move in order to reach S_i is

$$d(X_w; S_i) = D_w - R.$$

Case (4) $\alpha_w > i\beta$ and $\alpha_w - i\beta < \frac{\pi}{2}$. This case is similar to Case 1.

$$d(X_w; S_i) = \sqrt{(D_w \sin\gamma - R)^2 + (D_w \cos\gamma)^2}.$$

Case (5) $\alpha_w > i\beta \wedge \alpha_w - i\beta \geq \frac{\pi}{2}$. This case is similar to Case 2.

4.4. Placement algorithm

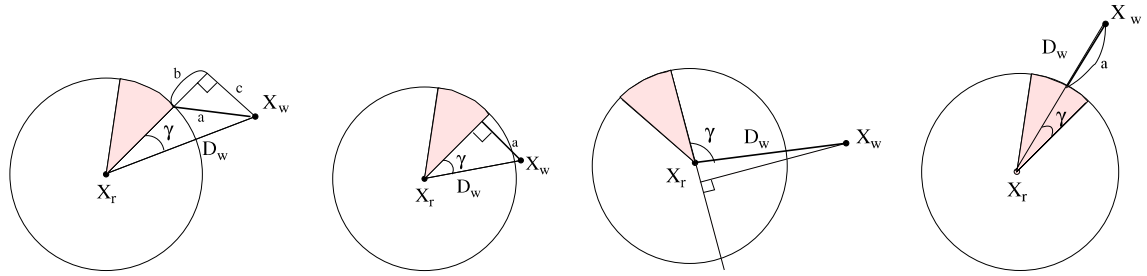
We propose Algorithm *Choose*-1-*out-of*-*M*. Executed at the requester node r, it assigns to each circular sector an available sensor with the lowest cost. Assigning sensors to concentric circles can be done in similar manner, where CS_is are considered instead of S_is. The proposed algorithm can be easily extended to select D sensors out of M for each region, or to assign different number of sensors to different regions.

Algorithm *Choose*-1-*out-of*-*M* uses the following variables. For each sensor i, $1 \leq i \leq M$, the variable $x_i \in \{0, 1, \ldots, K\}$ indicates either the circular sector for which sensor i has been selected ($x_i \geq 1$) or that the sensor i is still available ($x_i = 0$). For each circular sector j, $1 \leq j \leq K$, the variable $y_j \in \{0, 1, \ldots, M\}$ indicates either the sensor assigned to it ($y_j \geq 1$) or that the circular sector is still available ($y_j = 0$).

The algorithm has at most $K+1$ steps. Step 0 is the initialization. The x_is and y_js are set to zero. For each of the M sensors and K circular sectors, we compute $d(X_i; S_j)$, the minimum distance sensor i needs to move in order to reach the boundary of the desired region S_j, following the method described in Section 4.3. We then compute $cost(i, j)$, the cost of moving sensor i to region S_j as the product of the distance $d(X_i; S_j)$ and the cost per unit distance $cpud(i)$ of sensor i, reported by sensor i in the message *Available*.

At each step k, $1 \leq k \leq K$, the algorithm selects an available sensor that has the minimum cost for some region that has not been occupied and invites that sensor to move in that region. An optimal solution for Algorithm *Choose*-1-*out-of*-*M* seeks to select moves of the minimum cost, ignoring the moving time. Alternatively, we can consider moves of the shortest time as an alternative for the objective function; in that case we will compute $time(i, j)$ to be the time it takes for sensor i to move to region S_j, which is simply the ratio between $d(X_i; S_j)$ and the speed $speed(i)$ reported by sensor i in the message *Available*.

Theorem 2. *Algorithm Choose*-1-*out-of*-*M assigns* $\min\{M, K\}$ *sensors, one per circular sector.*



(a) Case 1.1. (b) Case 1.2. (c) Case 2. (d) Case 3.

Fig. 5. Moving node w into the sectors of the desired region C.

Algorithm 4.1 Algorithm *Choose*-1-*out-of-M*

Input:

M	total number of sensors
K	total number of sectors
X_r	location of the requester node
$\{X_i \mid 1 \le i \le M\}$	location of sensor i
$cpud(i), 1 \le i \le M$	cost per unit distance of sensor i

Step 0 *(Initialization):*
for $i = 1$ to M do set $x_i = 0$
for $j = 1$ to K do set $y_j = 0$
for $i = 1$ to M do
for $j = 1$ to K do
compute $d(X_i; S_j)$
compute $cost(i, j) = d(X_i; S - j) \times cpud(i)$
Step k, $1 \le k \le K$*:*
L1 select i_0 and j_0 such that
$cost(i_0, j_0) = min\{cost(i, j) | (1 \le i \le M) \wedge x_i = 0$
$\wedge (1 \le j \le K) \wedge y_j = 0\}$
L2 if no sensor is available then STOP
L3 set $x_{i_0} = j_0$; set $y_{j_0} = i_0$ // assign sensor to that region

Proof. After step 0, the condition $x_i = 0 \wedge y_j = 0$ is *true* for all sensors i, $1 \le i \le M$, and all circular regions j, $1 \le j \le K$.

In step 1, the instruction $L1$ selects the minimum cost value among all sensors and all circular sectors. The sensor i_0 and the circular sector j_0 with that minimum cost will be assigned to each other by the instruction $L3$. Neither the sensor i_0 nor the circular sector j_0 will be considered for future selection by the instruction $L2$ executed during some step k, $1 < k \ge K$, since $x_{i_0} = j_0$ and $j_0 \ge 1$, and $y_{j_0} = i_0$ and $i_0 \ge 1$.

At each step k, $1 \le k \le K$, one sensor gets assigned to a single circular region, so the number of available sensors is decremented by one. If M, the number of sensors, is less than or equal to K, then the instruction $L1$ will be executed exactly M times and all the M sensors will be assigned to M regions. The instruction $L2$ will stop the algorithm in case $M < K$. If $M > K$ then the instruction $L1$ will be executed exactly K times and K sensors will be selected and assigned to K circular sectors. □

The following communication occurs:

- On receiving one or more messages *Available*, the IS decides which nodes are to be invited to move and where they will be placed.
- The MDSTC of a node that receives the message *moveNode* not addressed to it forwards it to the neighbor that has forwarded the message *Available* to it.
- On receiving the message *moveNode*, the MDSTC of the desired node arranges for the node to move.

5. Application—tracking a mobile target in an urban scenario

In an urban environment, the ambient noise for certain type of sensors is more prevalent than in an open field or underwater due to buildings, people, motor vehicles, and so on. We simulated the sensor network in an urban environment in NS-2, using sensory data collected during training experiments executed in our laboratory for locating and tracking a Segway RMP robot (as the moving object). Each sensor from the laboratory has been simulated in NS-2 and its readings were emulated using the readings of that sensor in the lab during the training phase. The spatial-temporal information of the Segway's movement collected by individual sensors are fused in NS-2 by clustering the sensors along the estimated path of the Segway. The cluster heads of each of the formed clusters then estimate the Segway's position and velocity. The urban scenario in NS-2 simulates the positioning of the sensors in the lab as follows. Three types of sensors were deployed on roads arranged as a Manhattan-like grid (see Fig. 6): pressure sensors, video sensors (cameras), and magnetic sensors. A large brown block represents a building and a yellow dashed line is a road marking for a 20-feet wide two-way street.

312 fixed pressure sensors are embedded in the ground of two-way streets. Each pressure sensor generates an analog voltage due to the pressure applied when an object goes over it. This voltage is fed into a 10-bit A/D channel and thereby the output ranges from 0 to 1023. Video sensors, mounted on the buildings, have adjustable view angles so that the sensing coverage can be easily changed; there are six of them on each building. A video sensor takes snapshots of the scene within its view at a certain frequency and these images are used to construct PFSA. 72 magnetic sensors are initially positioned close to the street intersections and are mounted on some mobile platforms that can move freely along the streets; they can be quickly relocated if necessary.

Nodes communicate wirelessly using the two-ray ground reflection radio model. Each sensor node uses the 802.11 channel access scheme. The AODV algorithm is employed as the routing protocol for the entire network. The transmission power of a single node is 0.3 W and the reception power is 0.2 W. We recall that a sensor broadcasts its observed subpattern only if the stimulus created by the target is above a certain threshold. To save more energy, we consider a sleep-and-awake schedule for the sensor nodes, A-SAS, proposed by [26]. For a 1500 second-simulation, around 50% of the nodes are awake at any moment. Table 2 gives the energy consumption of the nodes when A-SAS is used or not. We note that the nodes that have joined a cluster consume more energy than the sensing only nodes. We have incorporated and run the rest of the simulations using the A-SAS schedule.

In Section 3 we showed how subpatterns for pressure, magnetic, and video sensors are constructed during the training phase. We consider composite patterns that are 3-tuple of subpatterns, one from each modality. During the operational phase we locate and track vehicles carrying large amounts of metallic material.

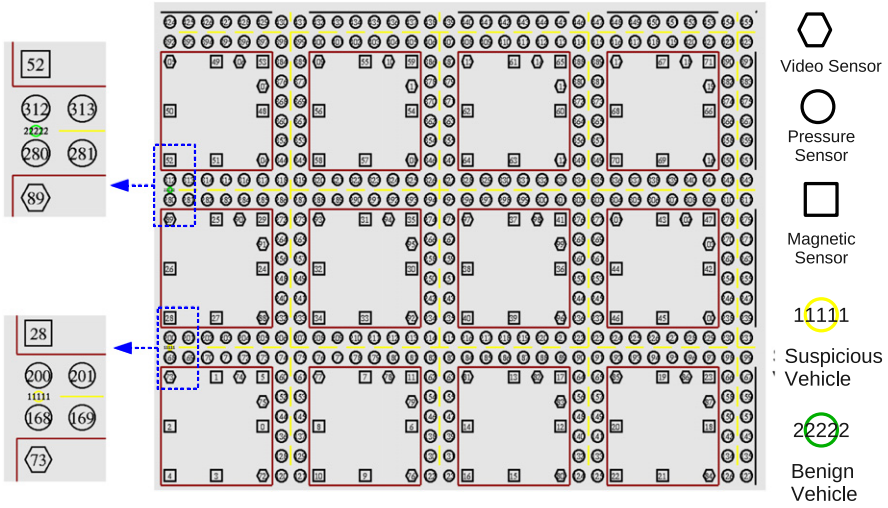


Fig. 6. Urban multi-modal sensor network.

Table 2
Average energy consumption per sensor node during a 1500 -second simulation (in J).

	Sensing only nodes	Nodes once in a cluster
Without A-SAS	0.2194	0.2905
With A-SAS	0.1052	0.2068

Such a vehicle will trigger subpattern matches at the pressure, video, and magnetic sensors that are geographically closer to its moving location. Thus a composite pattern that involves PFSA from these multi-modal sensors could be identified and a heterogeneous cluster that contains the sensors that matched the subpattern be formed. Such a vehicle is deemed by our sensor network as suspicious and is marked with yellow color in Fig. 6. Vehicles that do not carry large amounts of metal will not trigger a subpattern match at any magnetic sensor, thus a composite pattern that involves PFSA from pressure, video and magnetic sensors cannot be identified. Such a vehicle is considered by our sensor network as benign and is marked with green in Fig. 6.

Based on the speed of the robot and the fact that a cluster needs some time to form, we fix the maximum lifetime of any cluster to be 80 s and the minimum lifetime of any cluster to be 50 s.

Target location estimation is done at the cluster head after the cluster is formed. We use a triangulation scheme based on the location of the cluster members to estimate the target location which works as follows. When a cluster has been formed, each cluster member starts estimating its distance to the target and sends that distance to the cluster head. Then the cluster head collects these local estimators including its own estimator and finalizes a localization via a non-linear least squares triangulation algorithm [13]. Such a triangulation algorithm requires at least three estimators from the nodes in a cluster. In general, more nodes lead to better localization. If the size of a cluster is less than three, a localization cannot be properly done. We consider only that the magnetic and the pressure sensors contribute to the triangulation for heterogeneous clusters. This is justified on the basis that while a video sensor may be more important for long distance surveillance, for target localization the utility of magnetic sensor data and the pressure data is much higher. Both magnetic and pressure sensors are assumed to be isotropic, which means that the sensor measurement depends only on the distance to the target.

The triangulation scheme used by the cluster head has a smaller error than individual estimations sent by individual sensor nodes, since, without clustering, the localization error of a pressure sensor could be of 1 m.

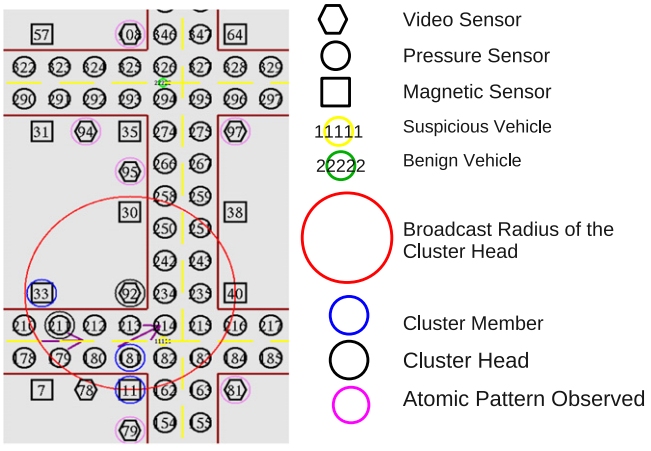


Fig. 7. Tracking a suspicious vehicle using composite patterns.

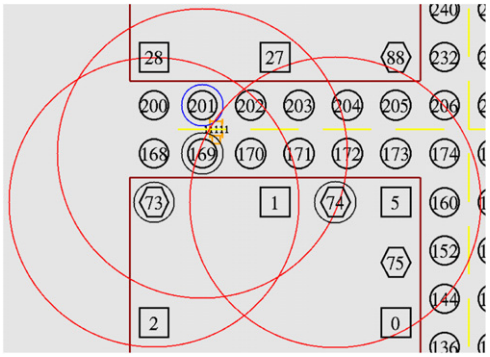
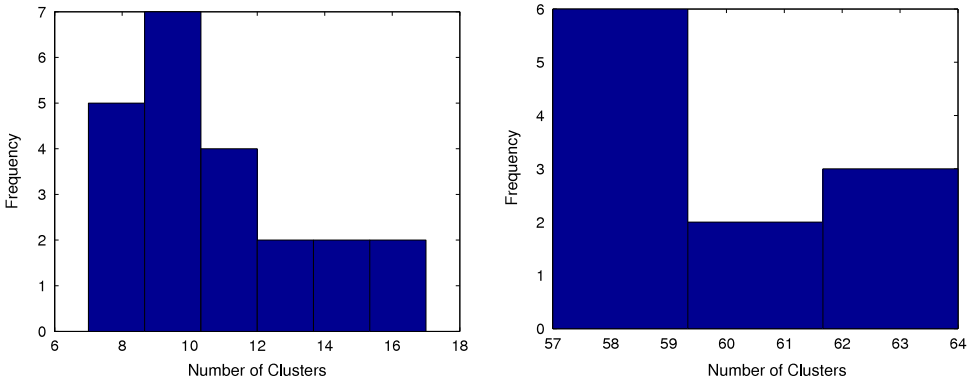


Fig. 8. Snapshot of homogeneous clustering.

Fig. 7 shows a snapshot of the NS-2 simulation for heterogeneous clustering. Arrows indicate the estimated direction of the target (estimated velocity) and the origin of the arrow indicates the estimated position of the target. Homogeneous clustering using the same sensor network is shown in Fig. 8; three homogeneous clusters, corresponding to the three modality sensors, form around the target. However, only the cluster of pressure sensors generates target location estimation useful for tracking, because the other two clusters do not have good location estimation.



(a) Heterogeneous clusters. (b) Homogeneous clusters.

Fig. 9. Histogram of the number of clusters formed.

Using only homogeneous clusters, it is impossible to do a proper classification when two targets can trigger similar subpatterns for some modality sensors but not for all, unless we add another layer of inter-cluster communication. Thus the main disadvantage of a homogeneous cluster is that it cannot properly distinguish benign targets and malicious targets, except for the cluster of magnetic sensors, and the network cannot selectively track the targets, which leads to unnecessary cluster formation and therefore wastes the network resources. In Fig. 7, when two vehicles are present, one labeled as "suspicious" and one labeled as "benign", the benign vehicle carries little metal and will not trigger subpattern generation at the magnetic sensors, due to the fact that its magnetic sensory data does not reach the threshold imposed (see Section 3).

To compare the localization performance, we modify MDSTC to form either heterogeneous or homogeneous clusters in two scenarios: (1) homogeneous clustering (*homog_Clus*), (2) heterogeneous clustering (*heter_Clus*).

For both scenarios, only one target is included since a homogeneous cluster cannot distinguish between the benign and malicious target. We consider different paths for the target. The path parameter p gives the ratio of the distance between the target and the right building and the width of the road. For example, when p is 0.5, the target moves exactly in the middle of the street. We range p from 0.4 to 0.6 by the increment of 0.05 in the simulation. We conduct a 1500-second simulation for each p in each scenario.

Fig. 9 gives the histogram of the number of clusters formed during the 1500-second simulation for homogeneous clusters and heterogeneous clusters. The average number of clusters is around ten or eleven, while in the case of homogeneous clustering the average is over 50.

Table 3 gives the average error of position estimation for various path parameters using heterogeneous and homogeneous clustering. We note that the heterogeneous clustering provides a better localization for the malicious target than the homogeneous one. The reason is that in the heterogeneous scenario, the number of clusters formed is smaller than the homogeneous scenario, but the cluster size is larger. A larger cluster results in a better triangulation estimation. We also note that for some path parameters, the homogeneous clustering does not give any estimation. This is due to the fact that the cluster size should be at least 3 for the triangulation scheme to work and in the homogeneous scenario, either the magnetic sensors are geographically far away from each other or the pressure sensors have a relatively short sensing range. It is very difficult for the target to trigger three magnetic sensors or three pressure sensors roughly at the same time.

Table 4 gives the total number of estimations done for both scenarios. We note that the heterogeneous clustering does many more estimations than the homogeneous one because it allows different types of sensors to be included in one cluster. Hence our heterogeneous clustering makes the tracking more robust.

Table 3
Average localization estimation error in meters for homogeneous and heterogeneous scenarios and various path parameter values. (None means there is no estimation during the simulation.)

Scenarios\path p	0.4	0.45	0.5	0.55	0.6
heter_Clus	0.5917	0.5644	0.5443	0.6050	0.6822
homog_Clus	None	0.7950	0.6468	None	None

Table 4
Number of estimations done in both scenarios and various path parameter values.

Scenarios\path p	0.4	0.45	0.5	0.55	0.6
heter_Clus	35	28	59	11	14
homog_Clus	0	6	39	0	0

All three types of sensors must be present in the vicinity of the target to be able to form a heterogeneous cluster. If the sensor field is not dense enough, more likely a malicious target will be missed. To show the need of moving sensors and the purpose of our proposed moving algorithm, we purposely put some magnetic sensors to sleep. In this way we create a local area sparse in magnetic sensors. As shown in Fig. 10, sensors number 5 and 34 (gray dash boxes) are sleeping and no magnetic sensor monitors that intersection. Thus the track of the malicious target will be lost when it reaches that intersection since no cluster will be formed without at least one magnetic sensor. We have implemented the first method of dividing the region of interest using concentric circles and the placement algorithm. That will trigger the magnetic sensor number 27 to move to that intersection, as shown in Fig. 11. For our simulations, since we consider that every composite pattern needs at least one node of each modality, we have a degenerated case of the method using concentric circles, since only one magnetic sensor is requested to move. A heterogeneous cluster is formed in the vicinity of the intersection around the malicious target and the track of the target is followed further.

6. Conclusion and future work

We proposed a network controller that is responsible for clustering multi-modal sensor nodes only in the space–time vicinity of a moving object, thus improving data fusion and localizing and tracking the object that has triggered the composite pattern. We showed that in general composite-base estimation is more accurate than individual estimation, i.e. multi-modal (heterogeneous) clusters offer better estimation than homogeneous clusters. Thus

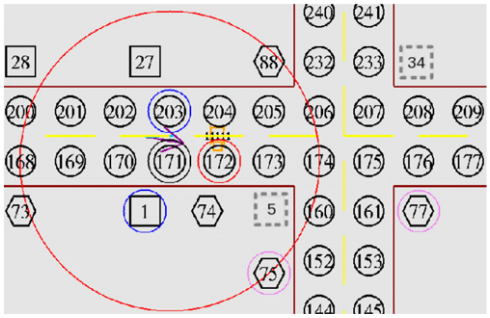


Fig. 10. Sparse sensing area at some intersection.

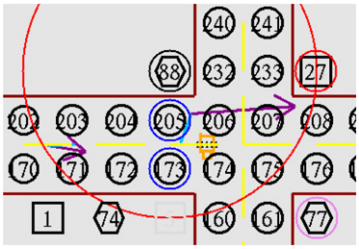


Fig. 11. A magnetic sensor moves to a sparse sensing area.

multi-modal clusters are better for estimation, as well as for classification.

Our multi-modal clustering algorithm MDSTC considers only one-hop clusters. In order to save energy, sensor nodes may reduce their transmission power so they may need to form multi-hop clusters in order to group enough sensors of various modalities. We plan to use the work of [17,11] to extend the results of a one-hop cluster to a multi-hop cluster where the transmission schedule of the nodes within the cluster plays a much more important factor than it does in an one-hop cluster. We plan also to design a traffic analyzer that observes the local traffic at a node in order to predict future events and alter the transmission schedule based on patterns of communication detected in the past and stored locally at the node.

References

[1] I. Amundson, B. Kusy, P. Volgyesi, X. Koutsoukos, A. Ledeczi, Time synchronization in heterogeneous sensor networks, in: Distributed Computing in Sensor Systems, DCOSS, 2008, pp. 17–31.

[2] R. Brooks, D. Friedlander, J. Koch, S. Phoha, Tracking multiple targets with self-organizing distributed ground sensors, Journal of Parallel and Distributed Computing 64 (2005) 874–884.

[3] R. Brooks, M. Zhu, J. Lamb, S. Iyengar, Aspect-oriented design of sensor networks, Journal of Parallel and Distributed Computing 64 (2004) 853–865.

[4] Q. Cao, T. Abdelzaher, T. He, J. Stankovic, Towards optimal sleep scheduling in sensor networks for rare-event detection, in: 4th International Symposium on Information Processing in Sensor Networks, IEEE Press, 2005, pp. 20–27.

[5] K. Chakrabarty, S. Iyengar, H. Qi, E. Cho, Grid coverage for surveillance and target location in distributed sensor networks, IEEE Transactions on Computers 51 (2002) 1448–1453.

[6] I. Chattopadhyay, A. Ray, Structural transformations of probabilistic finite state machines, International Journal of Control 81 (5) (2008) 820–835.

[7] J. Cortez, S. Martinez, T. Karatas, F. Bullo, Coverage control for mobile sensing networks, IEEE Transactions on Robotics and Automation 20 (2) (2004) 243–255.

[8] D. Friedlander, C. Griffin, N. Jacobson, S. Phoha, R. Brooks, Dynamic agent classification and tracking using an ad hoc mobile acoustic sensor network, EURASIP Journal on Applied Signal Processing 4 (2003) 371–377.

[9] D. Friedlander, S. Phoha, Semantic information fusion for coordinated signal processing in mobile sensor networks, International Journal of High Performance Computing Applications 16 (3) (2002) 235–241. Special Issue on Sensor Networks.

[10] S. Ghosh, P. Veeraraghavan, S. Singh, L. Zhang, Performance of a wireless sensor network MAC protocol with a global sleep schedule, International Journal of Multimedia and Ubiquitous Engineering 4 (2) (2009) 99–114.

[11] C. Joo, X. Lin, N. Shroff, Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks, IEEE/ACM Transactions on Networking 17 (4) (2009) 1132–1145.

[12] R. Kannan, S. Sarangi, S. Iyengar, Sensor-centric energy-constrained reliable query routing for wireless sensor networks, Journal of Parallel and Distributed Computing 64 (2004) 839–852.

[13] C. Kelley, Iterative Methods for Optimization, in: SIAM Frontiers in Applied Mathematics, 1996.

[14] L. Lazos, R. Poovendran, J. Ritcey, Probabilistic detection of mobile targets in heterogeneous sensor networks, in: International Conference on Information Processing in Sensor Networks, IPSN, 2007, pp. 519–528.

[15] A. Lim, Q. Yang, K. Casey, R.-K. Neelisetti, Real-time target tracking with CPA algorithm in wireless sensor networks, in: IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2008, pp. 305–313.

[16] H. Lin, J. Rushing, S. Graves, E. Criswell, A data fusion algorithm for large heterogeneous sensor networks, in: International Conference Wireless Algorithms, Systems, and Applications, 2007, pp. 225–230.

[17] X. Lin, N. Shroff, The impact of imperfect scheduling on cross-layer congestion control in wireless networks, IEEE/ACM Transactions on Networking (2006) 302–315.

[18] K. Ma, Y. Zhang, W. Trappe, Managing the mobility of a mobile sensor network using network dynamics, IEEE Transactions on Parallel and Distributed Systems 19 (1) (2009) 106–120.

[19] S. Phoha, N. Jacobson, D. Friedlander, R. Brooks, Sensor network based localization and target tracking through hybridization in the operational domains of beamforming and dynamic space–time clustering, in: IEEE Global Telecommunications Conference, vol. 5, 2003, pp. 2952–2956.

[20] S. Phoha, J. Koch, E. Grele, C. Griffin, B. Madan, Space–time coordinated distributed sensing algorithms for resource efficient narrowband target localization and tracking, International Journal of Distributed Sensor Networks 1 (2005) 81–99.

[21] S. Phoha, T.L. Porta, C. Griffin (Eds.), Sensor Network Operations, Wiley-IEEE Press, 2006, Ch. 15, Noise-adaptive sensor network for vehicle tracking in the desert, S. Phoha, E. Grele, C. Griffin, J. Koch, B.B. Madan, pp. 705–716.

[22] S. Phoha, A. Ray, Dynamic information fusion driven design of urban sensor networks, in: IEEE International Conference on Networking, Sensing and Control, 2007, pp. 1–6.

[23] S. Phoha, A. Ray, I. Chattopadhyay, G. Mallapragada, Y. Wen, Mathematical modeling of sensor network dynamics for control and stability, Tech. Rep. TR eSensIF-08-01, Applied Research Laboratory, Penn State University (September 2008). http://strange.arl.psu.edu/eSensIF/Publications/TR%20eSenseIF-08-01.pdf .

[24] A. Ray, Symbolic dynamic analysis of complex systems for anomaly detection, Signal Processing 84 (7) (2004) 1115–1130.

[25] N. Shrivastava, R. Mudumbai, U. Madhow, S. Suri, Target tracking with binary proximity sensors: fundamental limits, minima descriptions, and algorithms, in: International Conference on Embedded Networked Sensors Systems, SenSys, 2006, pp. 251–264.

[26] A. Srivastav, A. Ray, S. Phoha, Adaptive sensor activity scheduling in distributed sensor networks: a statistical mechanical approach, International Journal of Distributed Sensor Networks 5 (3) (2009) 242–261.

[27] D. Tran, T. Nguyen, Localization in wireless sensor networks based on support vector machines, IEEE Transactions on Parallel and Distributed Systems 19 (7) (2008) 981–994.

[28] Y. Wang, L. Cao, T. Dahlberg, F. Li, X. Shi, Self-organizing fault-tolerant topology control in large-scale three dimensional wireless networks, ACM Transactions on Autonomous and Adaptive Systems 4 (3) (2009).

[29] Y. Wen, D. Bein, S. Phoha, Middleware for heterogeneous sensor networks in urban scenarios, in: ITNG 2010, 2010, pp. 1–6.

[30] Q. Yang, A. Lim, K. Casey, R.-K. Neelisetti, An enhanced CPA algorithm for real-time target tracking in wireless sensor networks, International Journal of Distributed Sensor Networks 5 (5) (2009) 619–643.

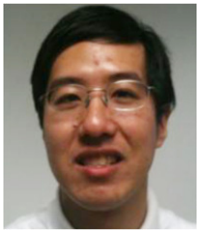
[31] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, S. Singh, Exploiting heterogeneity in sensor networks, in: IEEE Conference on Computer Communications, INFOCOM, 2005, pp. 366–377.

[32] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: IEEE Conference on Computer Communications, INFOCOM, 2002, pp. 1567–1576.

[33] W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Transactions on Networking 12 (2004) 493–506.

[34] Y. Zou, K. Chakrabarty, Uncertainty-aware and coverage-oriented deployment for sensor networks, Journal of Parallel and Distributed Computing 64 (2004) 788–798.

Doina Bein has received her Ph.D. in Computer Science from the University of Nevada, Las Vegas, USA. She is a research faculty member in the Applied Research Laboratory at The Pennsylvania State University. Her current research is focused on the complexities that are derived from advanced interaction of computationally powerful devices especially in communication and novel innovative approaches such as the use of randomization in wireless networks. Her publications are in energy-efficient communication on wireless networks, routing and broadcasting in wireless networks, fault tolerant coverage, self-organizing wireless networks, fault tolerant systems and self- stabilizing algorithms. She is an IEEE and SIAM member.



Yicheng Wen has received his Masters degree in Mechanical Engineering from The Pennsylvania State University. He is currently a Ph.D. candidate in Mechanical Engineering and holds a research assistantship position in the Networked Robotics and Sensors Laboratory at The Pennsylvania State University. His current research is focused on formal-language-theoretic approach to hierarchical data fusion including modeling, pattern classification, adaptive data routing and optimal decision making in heterogeneous sensor networks.

Shashi Phoha is a Professor of Electrical Engineering and the Director of Information Sciences and Technology Division of the Applied Research Laboratory at Penn State University. She has led multi-organizational advanced research programs and laboratories in major US industrial and academic institutions. She was the Director of the Information Technology Laboratory of the NIST. Her research focuses on information sciences, which bring together ideas from logic, operations research, formal languages, automata theory, dynamic systems, and control theory in diverse applications. She has forged new research directions in computational sciences that enable dependable distributed automation of multiple interacting devices. Drawing on her multidisciplinary research innovations, she pioneered the use of formal methods for the scientific analysis of distributed information for decision support, multi-stage coordination and intelligent control. She has over 200 publications, three books, and several patents licensed to industry. She is an editor of the International Journal of Distributed Sensor Networks.

Bharat B. Madan is with the Applied Research Laboratory of The Pennsylvania State University as Head of the Distributed Systems Department. His research interests are in the areas of distributed sensor systems, sensor data and information fusion, wireless networking protocols, intrusion tolerant secure systems and signal processing. Over the last six years, he has worked on a number of DARPA/DoD sponsored research projects dealing with intrusion tolerance, sensor networks and semantics driven service oriented network of sensors & platforms. He is also the PI for a DTRA funded project on the design and analysis of networks subject to large cascading failures. Prior to this, he was a faculty at the Indian Institute of Technology, New Delhi as a Professor in the Computer Science & Engineering Department, where he taught and conducted research in the areas of distributed systems, high performance computing, computer networks, data structures and signal processing. He has also held visiting positions at Loughborough University, Naval Postgraduate School, University of Delaware and the Old Dominion University. During the years 2002–2004, he was a Visiting Scholar with the Electrical & Computer Engineering Department, Duke University, where he was involved in teaching graduate courses and conducting research in the areas of stochastic modeling, computer security and intrusion tolerant systems and voice/data wireless systems.

Asok Ray earned the Ph.D. degree in Mechanical Engineering from Northeastern University, Boston, MA in 1976, and also graduate degrees in each of the disciplines of Electrical Engineering, Computer Science, and Mathematics. Dr. Ray joined The Pennsylvania State University in July 1985, and is currently a Distinguished Professor of Mechanical Engineering and a Graduate Faculty of Electrical Engineering. Prior to joining Penn State, Dr. Ray held research and academic positions at Massachusetts Institute of Technology and Carnegie–Mellon University as well as research and management positions at GTE Strategic Systems Division, Charles Stark Draper Laboratory, and MITRE Corporation. Dr. Ray is a Fellow of IEEE, a Fellow of ASME, and a Fellow of World Innovation Foundation (WIF). Further details are available at http://www.mne.psu.edu/Ray/.