TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/09 pp413–426
Volume 22, Number 4, August 2017

Distributed Algorithms for Event Reporting in Mobile-Sink WSNs for Internet of Things

Catalina Aranzazu-Suescun and Mihaela Cardei*

Abstract: Wireless Sensor Networks (WSNs) have many applications, such as climate monitoring systems, fire detection, smart homes, and smart cities. It is expected that WSNs will be integrated into the Internet of Things (IoT) and participate in various tasks. WSNs play an important role monitoring and reporting environment information and collecting surrounding context. In this paper we consider a WSN deployed for an application such as environment monitoring, and a mobile sink which acts as the gateway between the Internet and the WSN. Data gathering is a challenging problem in WSNs and in the IoT because the information has to be available quickly and effectively without delays and redundancies. In this paper we propose several distributed algorithms for composite event detection and reporting to a mobile sink. Once data is collected by the sink, it can be shared using the IoT infrastructure. We analyze the performance of our algorithms using WSNet simulator, which is specially designed for event-based WSNs. We measure various metrics such as average residual energy, percentage of composite events processed successfully at the sink, and the average number of hops to reach the sink.

Key words: composite events; distributed algorithm; energy efficiency; event-based clustering; Internet of Things; mobile sink; wireless sensor networks

1 Introduction

One of the key concepts of the Internet of Things (IoT) is to interconnect billions of devices to generate a "smart" environment. Sensor-Actuator-Internet is the framework for this smart environment[1].

Wireless Sensor Networks (WSNs) have been widely used in multiple IoT applications due to ubiquitous sensor devices[1–3]. Extensive research activities in the topics of security[4, 5], topology[6], synergies with other technologies[7], and energy consumption[1] in WSNs for IoT have been conducted in the last five years.

• Catalina Aranzazu-Suescun and Mihaela Cardei are with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA. E-mail: {caranzazusue2014, mcardei}@fau.edu.

∗ To whom correspondence should be addressed.

Manuscript received: 2016-11-20; revised: 2017-02-25; accepted: 2017-03-01

WSNs data collection and event reporting are important research topics in the IoT. The IoT is a world-wide network where all devices are interconnected and information has to be available fast and in an efficient way, thus redundancies and useless information have to be eliminated. A key approach for efficient interconnection is to give devices a "smart" behavior where they can communicate, process information, and take decisions without human intervention[8].

We classify the events to be detected by WSNs into atomic and composite events. *Atomic events* measure changes of a single attribute in the environment, for example, the temperature, while *composite events* consist of groups of atomic events. Information from the sensors is aggregated, and events of interest are reported to the sink. Aggregating data closer to the event location saves energy[6] compared to the case when data is aggregated at the sink. An event-based clustering is proposed in Ref. [9], where the Cluster Head (CH) sends a report to the sink when a composite event is detected.

The sink plays an important role in our application. The sink sends a request to the WSN with the specifications of the composite event. Actually different sinks may initiate different requests, but in this paper we deal with the presence of a single sink. We assume that the sink is mobile, and it acts as a gateway between the WSN and the IoT network. Take the forest fire application as an example. The sink could be a forest ranger equipped with a smart-phone. In this case the sink can move at a pedestrian speed or higher speeds (e.g., golf-cart or car speed). A ranger could walk through the area, collecting data from the WSN with his smart-phone, and using the IoT network he can notify the responsible entities in case of abnormal measurements.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 introduces the event model, consisting of atomic and composite events. Section 4 describes the problem definition. In Section 5 we present our distributed algorithms for event reporting to the sink. The performance of our algorithms is illustrated in Section 6, where we conduct simulations using WSNet[10]. The conclusions are stated in Section 7.

2 Related Work

WSNs have been widely used in event monitoring applications such as environment, climate, animal monitoring, and also in the medical field (e.g., body sensors) and natural disaster. Some of the networks use homogeneous sensors, where only one type of sensor is used to monitor the environment. Heterogeneous networks use different types of sensors to collect data more effectively and accurately[11, 12].

Several authors have used the composite event concept in WSNs, where variations in the sensors' measurements in the environment are collected and aggregated. A composite event is a collection of atomic events or measurements of several types of sensors.

Clustering is an effective mechanism for data aggregation in WSNs. Many clustering algorithms have been proposed in literature, such as LEACH[13] and HEED[14]. The authors of Ref. [15] used k-means and three different classifiers, Feed Forward Neural Network (FFNN), Naive Bayes, and Decision Trees, to find patterns in data and to improve detection and data aggregation for a fire event. Data is classified into two different clusters: fire event and noise. It was shown that FFNN has better prediction accuracy than the other classifiers.

Authors of Ref. [16] proposed a fuzzy logic based algorithm to choose a set of decision nodes and some sensing nodes. The decision nodes use fuzzy rules and based on the measurements of the sensing nodes they estimate the real value of the environmental events and determine if sensors have failures or not.

In Ref. [17], the authors proposed a Cluster-based Energy efficient Composite event detection (CEC) protocol. Any existing mechanism can be used for forming the clusters. All CHs form a backbone used to deliver reports to the sink. Each CH performs local data aggregation to detect events. When an event is detected, a report is sent to the sink along a backbone of CHs. Contrary to this *fixed clustering*, Ref. [9] proposes an *event-based clustering*, where a cluster is initiated by and composed of nodes that detect events.

In WSNs the sink (or sinks) can be fixed or mobile. When the sink is mobile, the trajectory of the sink can be fixed, controlled (e.g., based on some parameters such as the residual energy) or random walk[18]. The Anchor-based Voronoi-scoping Routing Protocol[19] considers several sinks that move using a random path approach. Each sink chooses an anchor from its neighbors, based on the nodes' signal strength. The anchor sends a hello message to the sink's Voronoi scope neighbors, so that they know how to send the information to the sink.

The sink sends beacons to the anchor to maintain the link. If the signal strength between the sink and the anchor is low, then the sink chooses another node to be the anchor and the process repeats. The sink has a constant speed between 1 and 10 m/s.

Two of the algorithms that we propose in this article, NewTree-based Routing and Anchor-based Routing, also use the anchor concept and assume a random walk movement of the sink. Main differences are as follows:

(1) In our algorithms, the anchor sends beacon messages to maintain the link, not the sink. This is more efficient, because when the sink ceases to receive beacon messages, it can conclude that the anchor is not in range any longer. In addition, sending of data messages can substitute the beacon messages. In Ref. [19], the sink sends the beacons, so the anchor has to ACK the receipt and then the sink determines if the anchor is still in range.

(2) In Ref. [19], the sink speed is constant, while in our case the speed varies, which is a more realistic model.

(3) Our algorithms use a cluster-based framework, where data is collected and aggregated by the cluster head and then sent to the sink.

(4) Anchor-based Routing uses up to β anchors as intermediate steps for data collection from the CHs to the sink. This is more energy efficient than having the sink choose a new anchor and broadcast the information in the whole network.

In the TRAIL protocol[19], the sink generates a trail of its movement through the network. A node that has messages for the sink uses a recent trail if it has one, or uses a random walk protocol to send data to the sink or to a sensor node that has a recent trail of the sink.

Reference [20] also uses the concept of anchor or agent, which is a node closer to the sink. If the sink moves, it waits a specific time T to receive information from the agent node. If no information is received, then the sink selects another agent node. The previous agent stores information for some time, until the sink broadcasts a message from the new location and new paths are formed. Source node paths are computed by the new agent using the Endocrine Cooperative Particle Swarm Optimization Algorithm. The algorithm computes a fitness function of the path that depends on the energy of the nodes, the distance between nodes, and the communication delay. A path with a bigger fitness function has a more optimal path from the source node to the sink. The sink moves with 5 m/s. Our anchor-based algorithms use a different framework for forwarding data from the sensor nodes to the sink, different sink speeds, and multiple anchors to avoid the expense of selecting a new anchor too often.

Another random path approach is Data Driven Routing Protocol[21]. There are three types of nodes, based on a given parameter k. $O\text{-}nodes$ are 1-hop away from the sink, $M\text{-}nodes$ are the nodes between 2- and k-hops away from the sink, and $I\text{-}nodes$ are the nodes at distance at least $(k+1)$-hops away from the sink.

While the sink moves, it sends beacon messages to its 1-hop neighbors. Beacon messages are resent in the network, so that the $O\text{-}nodes$ and the $M\text{-}nodes$ can update their routes to the sink. $I\text{-}nodes$ send data to the sink using a random walk protocol until the first $M\text{-}node$ or $O\text{-}node$ is reached. Each route has an expiration time. $M\text{-}nodes$ update their routes only if the time-stamp of the route is newer than the one stored in the memory. The nodes keep the older route in the memory as a backup, so two paths are stored in the routing table. This approach is used for one or multiple sinks.

In our approaches and the algorithms discussed so far, the sink moves using a random walk. Alternatively, the sink could move along a fixed path[22], a circular path[23, 24], or a tour[25].

In Ref. [22], the sink location is based on the energy of the network, thus saving energy. Sink movement is controlled by a genetic algorithm that calculates a population of chromosomes. Each chromosome is evaluated by a fitness equation that depends on the energy of each sensor and the distance between the sink and the sensor. A chromosome is selected if it has the highest probability, computed as the ratio of the fitness of the chromosome and the sum of the fitness of all chromosomes. In this way, the sink will move to the position of the chromosome selected.

In Ref. [23], the sink moves along a circular path, centered in the middle of the area. The sink moves only when the nearby sensor nodes have less energy than some predefined threshold. This results in less sink movements and less route update messages flowing through the network.

The protocol proposed in Ref. [24] uses three types of nodes: ring, anchor, and regular nodes. Ring nodes are located at a specific distance from the center of the network and store information about the position of the anchor. The anchor node is the node closest to the sink and it is renewed when the link quality to the sink is lower than some threshold. When a regular node wants to send information to the sink, it first requests the position of the anchor from a ring node. The ring node replies with the position of the anchor, then the regular node can send the information to the anchor node using Geographical routing. The sink moves with a speed between 0 and 5 m/s.

The sink movement in Ref. [25] follows a predefined tour. The sink visits all the nodes in the area and collects data using 1-hop transmissions. The approach sets a number of *polling points*, where the sink moves to collect data. Using these polling points, the sink must cover all sensor nodes. The authors compute the movement tour of the sink through the network using a special case of the traveling salesman problem. This heuristic approach involves building a minimum spanning tree and it runs in polynomial time.

The Virtual Grid-based Dynamic Routes Adjustment scheme[26] is an infrastructure-based approach which partitions the network into fixed cells, where the node closest to the center is the cell-header. Adjacent cell-

headers communicate via gateway nodes, which are normally located on the border of the clusters. Only cell-headers send information to the sink. When the sink moves through the network, the cell-headers adapt their path to the sink using the following procedure. The sink sends beacons to the 1-hop cell-header which becomes the Originating Cell-Header (OCH). The OCH sets the sink as its next-hop. The OCH sends a route update to its neighbors called downstream cell-headers. The sink moves outside the field in counter-clockwise direction, with constant speed of 10 m/s.

In this paper we propose two infrastructure-based algorithms, called Grid Flooding and Grid Sink-based Routing. The CH is the node with the highest residual energy, and we set-up the cell size such that any CH is 1-hop neighbor with the CHs of the nearby cells on horizontal and vertical directions. In this way we do not need gateway nodes to ensure CH connectivity. In the Grid Flooding, data are flooded along the CH backbone, and as long as the sink is connected to at least one CH, it will receive the message. In our case the sink moves inside the deployment area using a random walk, thus it will always be within communication range of at least one CH.

In the Mobile Sink based Adaptive Immune Energy Efficient Clustering Protocol[27], the network is divided into R regions, where each region has the same number of nodes. The sink passes through each region and uses the adaptive immune algorithm to find its sojourn location and the location of the optimum CH. The communication range of each node is larger than the area of the region, so each node can send the information directly to the CH. Each CH uses Time Division Multiple Access to receive information from its cluster members without collisions. In addition, Code Division Multiple Access is used to avoid inter-cell interference. To conserve energy, the nodes in the network sleep until the sink reaches its sojourn location in that region. The path of the sink through the network is fixed (circular, rectangular, or linear), depending on the distribution of the regions.

In Ref. [28], the sink has a fixed movement path. The approach uses the concept of *Rendezvous Points* (RPs), which are a subset of nodes that collect information from their neighbors. The RPs are similar to source nodes. The goal of the approach is to find a tour, using a Traveling Salesman Problem approach, which passes through all the RPs. This tour is used by the sink to collect data in the network. RPs are nodes with a larger degree, which are farther away from other RPs. An optimal tour uses less RPs and covers the network with a minimal length. The sink moves with 1 m/s.

Reference [29] proposes algorithms for data gathering when the sinks move along fixed paths and controlled paths. In the fixed path approach, the sinks move along hexagonal perimeters, and they stop periodically to collect data. In the controlled path approach, a sink moves if the energy level of its 1-hop neighbors drops under a threshold. The sinks are interconnected all the time.

3 Event Model Description

A WSN event is defined as an observable occurrence of a phenomenon or an object during a period of time in a specific area[30]. We distinguish two types of events: *atomic events* and *composite events*.

An atomic event is triggered when a single sensing value (or attribute) exceeds a given threshold. Similar to Ref. [30], we denote an atomic event by $e(t, s, R)$ where t is the time when the event occurs and it can be a specific time or an interval, s is the location of the event and it can be a point or a region, and R is a logical expression defining the conditions when the event occurs. For example, the atomic event $e(t, s, R) = (9/1/2016, (x, y), \text{temperature} > 100\,^{\circ}\text{C})$ means that the temperature at the location (x, y) on $9/1/2016$ was greater than $100\,^{\circ}$C.

To detect complex events in certain areas, variations in several attributes have to be detected, not only in one attribute. To detect a composite event, a combination of several sensing values is needed. A composite event is therefore composed of several atomic events. Similar to Ref. [30], we denote a composite event as

$$E((e_1, \delta_1), (e_2, \delta_2), \cdots, (e_k, \delta_k), C_t, C_s, \delta) = (R_1 \wedge R_2 \wedge, \cdots, \wedge R_k \wedge C_t \wedge C_s, \delta),$$

where e_i, i=1, ..., k, are the atomic events forming the composite event. δ_i with $0 \leqslant \delta_i \leqslant 1$ is the confidence of e_i, indicating the probability of E occurring when e_i occurs. R_i is a logical expression defining when e_i occurs.

C_t is the constraint on atomic events' times $t_1, t_2, \ldots, t_k$. If C_t is a specific time point t_0, then C_t is satisfied if $t_i = t_0$ for all $1 \leqslant i \leqslant k$. If C_t is a time interval I, then C_t is satisfied if $t_i \in I$ for all $1 \leqslant i \leqslant k$.

C_s is the constraint on atomic events' locations $s_1, s_2, ..., s_k$. C_s can be a location point or a region. If C_s is a specific location point s_0, then C_s is satisfied

if $s_i = s_0$ for all $1 \leqslant i \leqslant k$. If C_s is a region R, then C_s is satisfied if $s_i \in R$ for all $1 \leqslant i \leqslant k$.

Usually the confidence δ of the composite event is defined as $\delta = \delta_1 + \delta_2 + \cdots + \delta_k$, and it is expected to satisfy the property $\delta_1 + \delta_2 + \cdots + \delta_k = 1$[30].

As an example, consider the composite event *forest fire detection* which is defined using three atomic events: $e_1(t_1, s_1, \text{temperature} > \text{th}_1)$, $e_2(t_2, s_2, \text{light} > \text{th}_2)$, and $e_3(t_3, s_3, \text{smoke} > \text{th}_3)$. The threshold values th_1, th_2, th_3 as well as the attributes constituting the composite event are assigned by experts in the field.

The composite event forest fire detection can be defined as $E((e_1, 0.5), (e_2, 0.3), (e_3, 0.2), I, R, \delta) = (\text{light} > \text{th}_1 \wedge \text{temperature} > \text{th}_2 \wedge \text{smoke} > \text{th}_3 \wedge t_1, t_2, t_3 \in I \wedge s_1, s_2, s_3 \in R, \delta = 0.5 + 0.3 + 0.2)$.

In this example, the confidence of forest fire occurring when e_1 is detected is 0.5 or 50%. If both e_1 and e_2 occur, then the confidence increases to 0.8. If all three atomic events occur, then the confidence of forest fire event is 1 or 100%.

4 Problem Definition

We consider a WSN consisting of n heterogeneous nodes N_1, N_2, ..., N_n and a sink S. We assume that the nodes are densely deployed and they are connected to the sink. All the nodes have the same communication range R_c and the same initial energy E_{init}.

The nodes are heterogeneous, since each node is equipped with one or multiple sensing components from the set $\{s_1, s_2, ..., s_m\}$. Each sensing component can be used to detect an atomic event for that attribute. For example, the Waspmote events sensor board[31] can detect temperature, humidity, vibration, and water, and measurement values can be sent using 802.15.4/ZigBee radio. There are few reason that nodes have different sets of sensing components[32]:

- Nodes may be manufactured with different sensing capabilities.
- Some nodes may have purposely turned off some sensing components due to energy constraints.
- Some sensing components may fail over time.
- Some of the sensing components cannot be used due to lack of memory for storing data.

Nodes in WSN are resource constraint in terms of power, bandwidth, memory, and computing capabilities. Since WSNs are sometimes deployed in hostile environments where human access is limited, and recharging or replacing wireless nodes is prohibited in such situations, mechanisms for event detection and reporting have to minimize power consumption in order to prolong network lifetime[33].

Table 1 shows the main notations used in this paper. In Fig. 1 we illustrate an example with 40 nodes. There are $m = 2$ types of sensing components. For each node N_i, $1 \leqslant i \leqslant 40$, we indicate the sensing components that the node is equipped with. The problem definition is presented next.

Problem Definition—Composite Event Detection and Reporting (CEDR) in Mobile-Sink WSNs: Given a WSN deployed in an area A, consisting of n nodes with different sensing components from the set $\{s_1, s_2, ..., s_m\}$ and a mobile sink S, design an energy-efficient distributed algorithm for detecting and reporting a composite event E inquired by the sink S. The composite event E is defined using some or all of the m atomic events corresponding to the attributes

Table 1 Notations.

E	Composite event
δ	Confidence of the composite event
e_i	Atomic event i
δ_i	Confidence of atomic event i
n	Number of nodes
m	Maximum number of sensing components
T	Convergecast tree rooted at S
N_j	Node j, $1 \leqslant j \leqslant n$
$N_j.\{s_{j_1}, s_{j_2}, ..., s_{j_k}\}$	Sensing components of node N_j, $1 \leqslant k \leqslant m$
$N_j.E_{\text{residual}}$	Residual energy of node N_j
$N_j.tp$	Parent of node N_j in T
R_c	Node communication range
A	Deployment area
$A.L$	Length of the side of the deployment area
E_{init}	Initial energy of each node

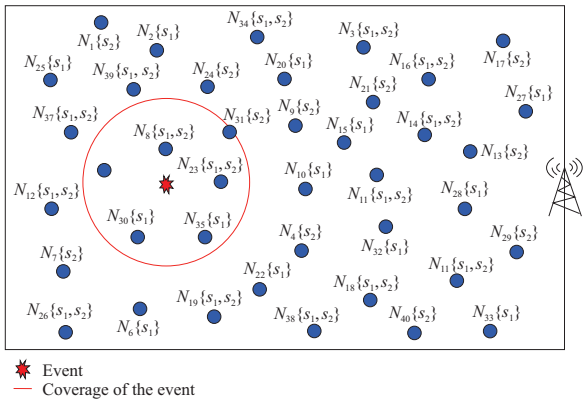

Fig. 1 Example of network deployment.

measured by the sensing components $\{s_1, s_2, ..., s_m\}$.

5 Distributed Algorithms for CEDR in Mobile-Sink WSNs

In this paper we assume that only one composite event is requested by the sink at a time. We propose the network protocol shown in Fig. 2.

An energy-efficient mechanism to aggregate data or events is using clustering. There are several approaches proposed in literature for clustering in WSNs[9, 13–15, 17]. We distinguish *fixed clustering* and *event-based clustering*.

5.1 CEDR for fixed clustering

In this case we assume that the clusters are fixed and they are constructed after the network is deployed, that means before Phase 1 begins. Therefore, in this case, the clustering does not account for the type of event to be monitored.

The area is divided into a grid, see Fig. 3, where the grid cell size is at most $R_c/\sqrt{5}$. In this way any two nodes from neighboring horizontal or vertical cells can communicate directly. The nodes in each grid cell form a cluster. We assume that the network is dense, so that each grid cell has at least one sensor node. The nodes are aware of their location. After the nodes are deployed, they can compute their grid cell based on their location. The CH is the node with the largest residual energy and in case of a tie, the one with the smallest ID becomes the CH. This can be implemented using the following simple protocol. After deployment, nodes send a *Hello* message containing their ID, location, and residual energy. The CH node, selected as specified previously, sends a message *JoinCluster* and all the nodes in the cluster reply with an *ACK* message containing their ID.

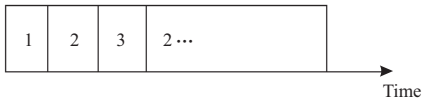

Fig. 2 Network organization.

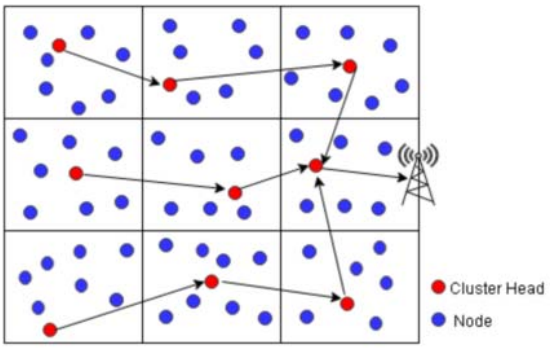

Fig. 3 Network deployment for fixed clustering.

The CHs form a connected backbone, see Fig. 3, used to communicate with the sink. The sink is always within communication range of at least one CH.

We propose two mechanisms for event reporting, *Grid Flooding* and *Grid Sink-based Routing*. For each mechanism we describe the three phases from Fig. 2.

5.1.1 Grid flooding

In Phase 1, the sink S broadcasts a message *CompositeEventRequest*(S, E, δ_{th}) along the CH backbone. E is the composite event with the fields $E((e_1, \delta_1), (e_2, \delta_2), ..., (e_k, \delta_k), C_t, C_s, \delta)$, as specified in Section 3. The field δ_{th} is the required minimum confidence, that means if $\delta \geqslant \delta_{th}$ then the composite event is detected successfully. The composite event E involves atomic events based on the sensing components from the set $\{s_1, s_2, ..., s_m\}$ and $k \leqslant m$.

The *CompositeEventRequest* message is flooded in the network along the backbone of CHs. More specifically, when a CH receives a *CompositeEventRequest* message for the first time, it will broadcast the message once. At that time the nodes in the cluster store the specifications of the request. The non-CH nodes do not re-transmit this message.

In Phase 2, the nodes that satisfy the location requirement C_s and are equipped with sensing components needed to detect one or more atomic events $e_1, ..., e_k$, start the detection process for a time duration C_t.

Phase 3 deals with event detection and event reporting. Phase 3 starts when one or more nodes detect variations in one or more attributes currently monitored. If an attribute value exceeds the threshold value, then an atomic value is detected.

We note that the event may span few grid cells, therefore one or more CHs are involved in the event reporting. There are instances when only the sink has all the information needed to detect the composite event, therefore all aggregated atomic events are reported to the sink.

When a sensor detects an atomic event it sends an *atomicEvent* message to its CH. The CH aggregates the atomic events received from its cluster and sends

an aggregated message *eventReport* to the sink by flooding, along the CH backbone. The sink is mobile, but as long as it is within communication range of at least one CH, it will receive the *eventReport* message. Note that the sink may be within communication range of multiple CHs, and in that case it will drop the duplicate messages. The pseudocode of the event reporting mechanism is presented in Algorithm 1.

5.1.2 Grid Sink-based Routing

In *Grid Sink-based Routing*, the report message is sent along a path of CHs rather than flooding.

In Phase 1, the sink S selects the closest CH as the root, denoted by R. This can be done using a simple protocol: the sink broadcasts *FindClosestCH* and the CHs in range reply with their ID and residual energy after a small random delay. The sink chooses the closest CH based on the signal strength and in the case of a tie chooses the CH based on the residual energy and the smallest ID. Then the sink sends the request *SinkInitiatedRequest*(S, R, E, δ_{th}). Similar to the previous algorithm, E is the composite event with the fields $E((e_1, \delta_1), (e_2, \delta_2), ..., (e_k, \delta_k), C_t, C_s, \delta)$. The field δ_{th} is the required minimum confidence, that means if $\delta \geqslant \delta_{\text{th}}$ then the composite event is detected successfully.

The root R then broadcasts a message *CompositeEventRequest*(R, E, δ_{th}, hops = 0) along the CH backbone. As the *CompositeEventRequest* message is flooded along the backbone of CHs, a convergecast tree T is formed, where R is the root. The tree T contains only CH nodes.

Each CH node N_j that receives the message for the first time, increments the *hops* field, sets the sending CH node as its parent in T, stored in the field $N_j.tp$, and sends a message *CompositeEventRequest*(N_j, E, δ_{th}, hops).

Note that the sink is mobile, thus the convergecast CH-tree T must change when S is not within R's communication range. Sometimes S may be stationary for some time, or move with a slow speed. T will not change as long as S is within R's communication range.

The following mechanism is used to update T. The root R sends periodically a beacon message. If R reports data, then the beacon is omitted that period. If the sink S does not hear a beacon (or data) from R for α periods (e.g., $\alpha = 2$), then a new convergecast tree T is formed. S chooses a new root R using the same mechanism, and the process of forming a new tree T is repeated as described previously.

Phase 2 is the same as in the *Grid Flooding* mechanism. Event reporting in Phase 3 is done along the parent path in the convergecast tree T. When a CH node N_j receives an *eventReport* message for the first time, it sends it to its parent $N_j.tp$. The pseudocode of the event reporting mechanism is presented in Algorithm 2.

5.2 CEDR for event-based clustering

In this section we propose mechanisms for *event-based*

Algorithm 1 Grid Flooding — Event Reporting (node N_j)

```
1:  if N_j.isCH == true then
2:      if first atomicEvent received from a node in N_j 's cluster then
3:          start timer t_1
4:          while t_1 > 0 do
5:              aggregate atomic events received from nodes in N_j 's cluster
                into atomicEventList
6:          end while
7:          if t_1 == 0 then
8:              send eventReport(N_j , atomicEventList) after a small random
                delay
9:          end if
10:     end if
11:     if eventReport message initiated by another CH is received for the
        first time then
12:         send eventReport message after a small random delay
13:     end if
14: end if
15: if N_j.isCH == false then
16:     if N_j detects one or more atomic events then
17:         send atomicEvent(N_j , N_j .CH, atomicEventList) after a small
            random delay
18:     end if
19: end if
```

Algorithm 2 Grid Sink-based Routing — Event Reporting (node N_j)

```
1:  if N_j .isCH == true then
2:      if first atomicEvent received from a node in N_j 's cluster then
3:          start timer t_1
4:          while t_1 > 0 do
5:              aggregate atomic events received from nodes in N_j 's cluster
                into atomicEventList
6:          end while
7:          if t_1 == 0 then
8:              send eventReport(N_j , N_j .tp, atomicEventList) after a small
                random delay
9:          end if
10:     end if
11:     if receive eventReport(N_i , N_j , atomicEventList) then
12:         send eventReport(N_i , N_j .tp, atomicEventList) after a small
            random delay
13:     end if
14: end if
15: if N_j .isCH == false then
16:     if N_j detects one or more atomic events then
17:         send atomicEvent(N_j , N_j .CH, atomicEventList) after a small
            random delay
18:     end if
19: end if
```

clustering. Rather than using fixed (or predetermined) clustering, in this case the cluster is initiated by nodes that detect events.

We note few drawbacks to the fixed clustering approach. First of all, communication consumes high energy in WSNs and more CHs need to report the event if it spans multiple grid cells. For example, even a small event located at the border of multiple grid cells will trigger event detection in multiple grid cells. Second, if the composite event involves sensing components from nodes located in neighboring grid cells, then only the sink has all the information needed to detect the composite event. Event-based clustering is expected to detect the composite event earlier, as the result of aggregation.

In the *event-based clustering* mechanisms, one or more clusters are formed by the nodes with sensing components that detect atomic events. A CH collects and aggregates information from the nodes in its cluster and then reports the event to the sink.

Clusters are formed in Phase 3 of the algorithm. We proposed two mechanisms for event reporting, *Anchor-based Routing* and *NewTree-based Routing*.

5.2.1 Anchor-based Routing

The maximum number of anchors β is a given argument and it is expected to have a small value such as $\beta = 3$.

In Phase 1 the sink S selects the closest node as the first anchor, denoted A_1, using the following mechanism. S broadcasts *FindClosestNode* and the nodes in range reply with their ID and residual energy after a small delay. The sink chooses the closest node based on the signal strength, and in case of a tie the residual energy and the smallest ID criteria are used. The sink S sends the request $SinkInitiatedRequest(S, A_1, E, \delta_{\text{th}})$, where E is the composite event and δ_{th} is the threshold parameter for the composite event.

A_1 then broadcasts a message *CompositeEventRequest*(A_1, E, δ_{th}, hops = 0) in the whole network. As the *CompositeEventRequest* message is flooded, a convergecast tree T is formed, where A_1 is the root. Each node N_j that receives the message for the first time, increments the $hops$ field, sets the sending node as its parent in T, stored in the field $N_j.tp$, and sends a message *CompositeEventRequest*(N_j, E, δ_{th}, hops).

Event reports will flow from the nodes to CH, from CH to A_1 along T, and from A_1 to S. As long as S is within communication range of A_1, no change is needed. To determine this, A_1 sends beacons (or data) periodically. If S does not hear a beacon (or data) from A_1 for α periods (e.g., $\alpha = 2$), then a mechanism for selecting a new anchor A_2 is initiated as follows. S broadcasts a message $NewAnchorRequest(S, A_1)$. Nodes which receive both A_1's beacons (or data) and S's message $NewAnchorRequest(S, A_1)$ are candidates to become the second anchor A_2, since they are connected to both A_1 and S. Such a node N_j waits a time based on the signal strength of the message $NewAnchorRequest(S, A_1)$, and sends a message $NewAnchorReply(S, A_1, N_j)$. The waiting time is smaller when the signal strength is higher. When the first message is received by the sink, S replies with $NewAnchorAck(S, A_1, N_j)$, and N_j becomes the second anchor.

A_2 sends now beacons (or data) periodically. The events flow along the path node $\rightarrow$ CH $\rightarrow A_1 \rightarrow A_2 \rightarrow S$. On the other hand, if no *NewAnchorReply* message is received by the sink, then the anchor selection process is reset, that means S selects a new first anchor A_1 and broadcasts *CompositeEventRequest*(A_1, E, δ_{th}, hops = 0) in the whole network.

If S moves out of the range of A_2, then the process repeats and a new anchor A_3 is selected. After the maximum number of anchors β is reached, the anchor selection process resets, that means a new anchor A_1 is selected.

In Phase 2, the nodes that satisfy the location requirement C_s and are equipped with sensing components needed to detect one or more atomic events $e_1, ..., e_k$, start the detection process for a time duration C_t.

Phase 3 deals with event detection and event reporting. We use the *event-based clustering* mechanism from Ref. [9]. We give here a brief overview. The cluster contains nodes that detect atomic events part of the composite event requested by the sink. The cluster may also contain some relay nodes which are only involved in connecting the sensing nodes to the CH. A node can become CH only if it detects at least one atomic event and if its residual energy is larger than a predefined threshold. Based on the residual energy and ID which is used for breaking ties, a node proclaims itself CH and sends a message *JoinCluster* over h_{cluster} hops. The nodes in the cluster form a cluster tree T_{cluster} rooted at CH. T_{cluster} is expected to have a small height h_{cluster}, such as 2 or 3. Since there is no guarantee that all the nodes

detecting atomic events are within h_{cluster}-hops of the CH, additional clusters may form.

A CH receives atomic events from cluster members, which are sent along T_{cluster}. As messages are sent from cluster members to the CH, the aggregation is performed.

The event reporting mechanism is as follows. The event is reported from CH to the anchor node A_1 along the tree T using the *tp* parent attribute. From A_1 the event is reported directly to the sink (if A_1 is the last anchor) or is using a path of at most β anchors to reach the sink S. The attribute *ap* stores the next anchor in the path to the sink. For example, for $\beta = 3$, $A_1.ap = A_2$, $A_2.ap = A_3$, and $A_3.ap = S$.

The pseudocode of the event reporting mechanism is given in Algorithm 3.

5.2.2 NewTree-based Routing

The *NewTree-based Routing* mechanism is the same as *Anchor-based Routing* for $\beta = 1$ anchor, and it follows the same framework. More specifically, in Phase 1 the sink S selects the closest node as the root, denoted R. The sink sends $SinkInitiatedRequest(S, R, E, \delta_{\text{th}})$ and then R broadcasts a message $compositeEventRequest(R, E, \delta_{\text{th}}, \text{hops} = 0)$ in the whole network. A convergecast tree T is formed, where R is the root.

Event reports flow from the cluster nodes to CH, from CH to R along T, and from R to S. As long as S is within communication range of R, no change is needed. R sends beacons (or data) periodically. If S does not receive a beacon (or data) from R for α periods (e.g., $\alpha = 2$), then a new root R is selected, and a new convergecast tree T rooted at the new root R is formed.

Phase 2 is similar to *Anchor-based Routing*. In Phase 3, the *event-based clustering* mechanism is used to build one or more clusters. The event reporting mechanism is briefly described next. Events are reported from cluster members to CH along T_{cluster}. From CHs, events are reported to the root R along the convergecast tree T, using the tp parent attribute. From R, the event is reported to the sink S. The pseudocode of the event reporting mechanism is presented in Algorithm 4.

Algorithm 3 Anchor-based Routing — Event Reporting (Node N_j)

```
1:  if N_j ∈ T_cluster then
2:      set timer t_1 based on the height of N_j in T_cluster
3:      start timer t_1
4:      while t_1 > 0 do
5:          aggregate atomic events received from its children in T_cluster into
            atomicEventList
6:      end while
7:  end if
8:  if N_j ∈ T_cluster and N_j.isCH == false then
9:      if t_1 == 0 then
10:         send atomicEvent(N_j, N_j.cp, atomicEventList) after a small
            random delay
11:     end if
12: end if
13: if N_j.isCH == true AND t_1 == 0 then
14:     if N_j.isFirstAnchor == true then
15:         send eventReport(N_j, N_j.ap, atomicEventList,
            firstAnchorReached = true) after a small random delay
16:     else
17:         send eventReport(N_j, N_j.tp, atomicEventList,
            firstAnchorReached = false) after a small random delay
18:     end if
19: end if
20: if receive eventReport(N_i, N_j, atomicEventList, firstAnchorReached)
    then
21:     if (firstAnchorReached == false AND N_j.isFirstAnchor == true)
        OR (firstAnchorReached == true) then
22:         send eventReport(N_j, N_j.ap, atomicEventList,
            firstAnchorReached = true) after a small random delay
23:     else
24:         send eventReport(N_j, N_j.tp, atomicEventList,
            firstAnchorReached = false) after a small random delay
25:     end if
26: end if
```

Algorithm 4 NewTree-based Routing — Event Reporting (Node N_j)

```
1:  if N_j ∈ T_cluster then
2:      set timer t_1 based on the height of N_j in T_cluster
3:      start timer t_1
4:      while t_1 > 0 do
5:          aggregate atomic events received from its children in T_cluster into
            atomicEventList
6:      end while
7:  end if
8:  if N_j ∈ T_cluster and N_j.isCH == false then
9:      if t_1 == 0 then
10:         send atomicEvent(N_j, N_j.cp, atomicEventList) after a small
            random delay
11:     end if
12: end if
13: if N_j.isCH == true AND t_1 == 0 then
14:     if N_j.isRoot == true then
15:         send eventReport(N_j, S, atomicEventList) after a small random
            delay
16:     else
17:         send eventReport(N_j, N_j.tp, atomicEventList) after a small
            random delay
18:     end if
19: end if
20: if receive eventReport(N_i, N_j, atomicEventList) then
21:     if N_j.isRoot == true then
22:         send eventReport(N_j, S, atomicEventList) after a small random
            delay
23:     else
24:         send eventReport(N_j, N_j.tp, atomicEventList) after a small
            random delay
25:     end if
26: end if
```

6 Simulations

We conducted simulations using WSNet[10], an open source event-based simulator for WSNs. WSNet was developed by the Center of Innovation in Telecommunication CITI Laboratory associated with INSA Lyon France. WSNet uses object-oriented C++ language, Linux operating system, and provides a platform where new modules can be developed. In addition, it provides support for energy model and event modeling, features which are important in WSNs. In this section we compare the performance of the four event-reporting mechanisms presented in Section 5.

6.1 Simulation environment

The main parameters used in simulations are listed in Tables 2 – 4.

The WSN is deployed into a square area A, where the square length A.L takes values between 440 m and 1100 m, see Table 3. The values have been selected such that the area can be divided into a grid, with cell size of 44 m. In this way, any sensors in horizontally or vertically adjacent cells can communicate directly. This feature is useful for our grid-based algorithms.

Table 2 Simulation parameters.

Simulation time	1 h
Antenna type	Omnidirectional
MAC layer	802.11
E_{init}	1 J
Node communication range R_{c}	100m
Packet length	132 bytes
Confidence threshold δ_{th}	0.75

Table 3 Network deployment parameters.

Number of rows	Number of cells	A.L (m)	Number of nodes, n
10	100	440	500
15	225	660	1125
20	400	880	2000
25	625	1100	3125

Table 4 Sink speed.

Average speed (m/s)	Maximum speed (m/s)
1.0	2
2.5	5
5.0	10
7.5	15
10.0	20
12.5	25

The nodes are deployed as follows. First, one node is deployed randomly in each cell. Then the remaining nodes are deployed randomly in A. In this way the resulting network is connected and each cell has one node, thus each cell can select a CH in the grid-based algorithms.

Initially, the sink S is located in the middle of the right side of A, see Fig. 3. The sink moves in the area A using a random walk, with the average and maximum speeds indicated in Table 4. The sink pauses for some time, then it moves with a speed between 0 and the maximum value for a random time. The direction angle has a random value between 0° and 360°.

The maximum number of sensing components is $m = 5$. Each node is equipped randomly with sensing components. We define a composite event with five atomic events. The sensing components involved, with confidence and threshold values, are presented in Table 5.

The five atomic events are defined as follows:

- $e_1(t_{\text{s}}, A, \text{temperature} > 150)$;
- $e_2(t_{\text{s}}, A, \text{pressure} > 50)$;
- $e_3(t_{\text{s}}, A, \text{humidity} > 10)$;
- $e_4(t_{\text{s}}, A, \text{smoke} > 100)$;
- $e_5(t_{\text{s}}, A, \text{light} > 80)$.

t_{s} is the simulation time after the request is sent by the sink S and A is the deployment area. The composite event that the WSN monitors is defined as $E((e_1, 0.35), (e_2, 0.1), (e_3, 0.15), (e_4, 0.3), (e_5, 0.1), t_s, A, \delta)$.

In each simulation run, we generate an event which has a circular coverage, see Fig. 1. The center is generated randomly. Three types of events are used in the simulations:

- *Small events*, where radius has a random value between 10% and 20% of A.L.;
- *Medium events*, where radius has a random value between 20% and 40% of A.L;
- *Large events*, where radius has a random value between 40% and 60% of A.L.

The nodes located in the event area, equipped with the corresponding sensing components, detect an

Table 5 Types of sensors used.

Sensor type	Confidence	Threshold
Temperature	0.35	150
Pressure	0.1	50
Humidity	0.15	10
Smoke	0.3	100
Light	0.1	80

atomic event with probability 95%.

The initial energy of each node is $E_{\text{init}} = 1$ J. To measure the energy consumed, we implemented the energy model from LEACH[13]. The energy consumed to transmit/receive an l-bit message over a distance d is computed as

$$E_{Tx}(l, d) = E_{\text{elec}} \times l + \epsilon_{\text{amp}} \times l \times d^2,$$
$$E_{Rx}(l) = E_{\text{elec}} \times l,$$

where $E_{\text{elec}} = 50$ nJ/bit and $\epsilon_{\text{amp}} = 100$ pJ/(bit $\cdot$ m^2). We do not take into account the energy consumed on sensing, since it is negligible compared with the energy consumed on transmitting and receiving messages.

We run each simulation scenario 5 times using different seed values to generate random numbers and report the average values in the graphs.

The simulation time for each algorithm is 1 h. If events are detected, then they are reported to the sink S every 5 s. More specifically, every 5 s nodes that detect atomic events send a report to the CH, and from here to the sink according to the rules presented in each algorithm. Based on the messages received, the sink computes the confidence to determine if the composite event was detected or not.

6.2 Simulation results

Figure 4 shows the residual energy of the network for $n = 3125$ nodes, A.L = 1100 m, and medium size events. The average sink speed is 1 m/s, 5 m/s, and 12.5 m/s, respectively. We observe that Grid Flooding consumes the most energy, since all CHs resend all event reports. The Grid Sink-based Routing consumes the least energy, since event reports are sent along a path of CHs, so fewer nodes are involved. Also, compared to NewTree-based Routing, updating the convergecast tree is done using only CH nodes.

NewTree-based Routing and Anchor-based Routing have comparable results when the average sink speed is small (see Fig. 4a), since in this case the process of building a new convergecast tree or adding a new anchor is not used too often. On the other hand, for higher speeds (see Figs. 4b and 4c) we observe that the Anchor-based Routing consumes less energy. When the sink is not in the range of the anchor and the maximum number of anchors β has not been reached, then another anchor is selected, thus avoiding to spend energy on building a new convergecast tree in the whole network.

In Fig. 5, the average sink speed is 5 m/s and the number of nodes is $n = 3125$. Results are measured for small, medium, and large events. The results on energy consumed by the four algorithms are consistent with those from Fig. 4. In addition, the larger the event, more energy is spent by the network on data reporting. This

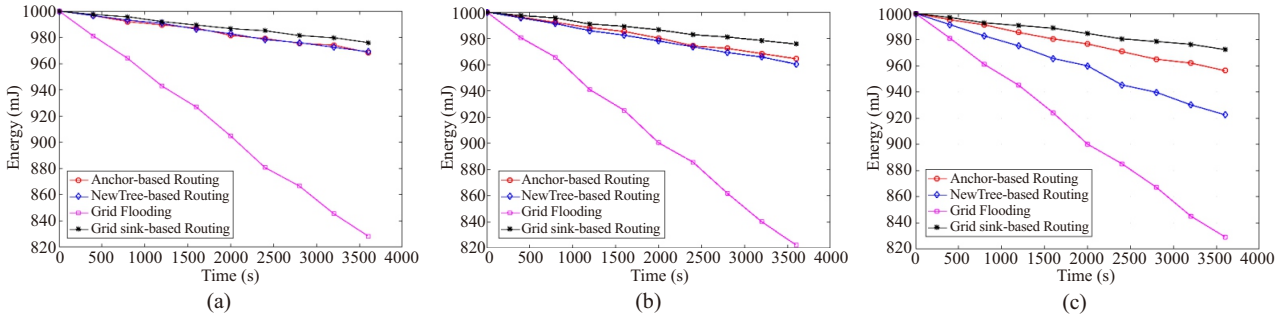

Fig. 4 Average residual energy of the network: (a) Average sink speed 1 m/s, (b) Average sink speed 5 m/s, (c) Average sink speed 12.5 m/s.

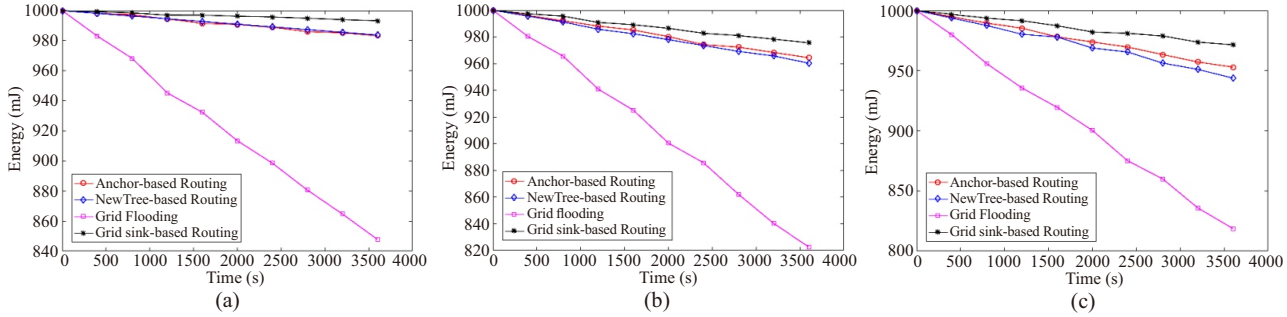

Fig. 5 Average residual energy of the network: (a) Small events, (b) Medium events, (c) Large events.

is because more nodes detect the event and participate in data reporting. Also, more clusters will be formed.

Figure 6 measures the percentage of composite events processed successfully at the sink, for small and large events. The number of nodes is $n = 3125$, and the sink average speed varies between 1 m/s and 12.5 m/s, using the values from Table 4. Except for Grid Flooding, in all algorithms the percentage of composite events processed successfully at the sink decreases as the average speed of the sink increases. In Grid Flooding, all CHs resend the event report messages, therefore the sink receives all messages regardless of its speed. We note that the sink moves in the deployment area, thus it will be within the communication range of at least one CH at all the times. The algorithm which is mostly affected by an increase in sink speed is Grid Sink-based Routing. The reason is that multiple nearby clusters are likely to use the same CH path to the root R, thus when the number of packets increases, some are lost due to contentions and collisions.

NewTree-based Routing gets slightly better results than Anchor-based Routing. The difference is less than 1% in our results. For a large number of event reports, some may be lost when they reach the root R or the first anchor A_1, which may be seen as a bottleneck. For Anchor-based Routing, if the path of anchors and the sink overlap with the part of network actively involved in event reporting, then additional contentions and collisions may result, thus more packets are dropped. On the other hand, NewTree-based Routing reconstructs the convergecast tree, thus this situation does not occur.

From the results of Fig. 6 we can also see that the percentage of composite events processed successfully by the sink is slightly larger for large events. This is because more sensors detect the event, thus the redundancy in event reporting help alleviate the impact of packet dropping.

Figure 7a shows the average number of CHs for the Event-based Clustering and Fixed Clustering, when n varies between 500 and 3125, using the values from Table 3. NewTree-based Routing and Anchor-based Routing are using Event-based Clustering, while Grid Flooding and Grid sink-based Routing are using Fixed

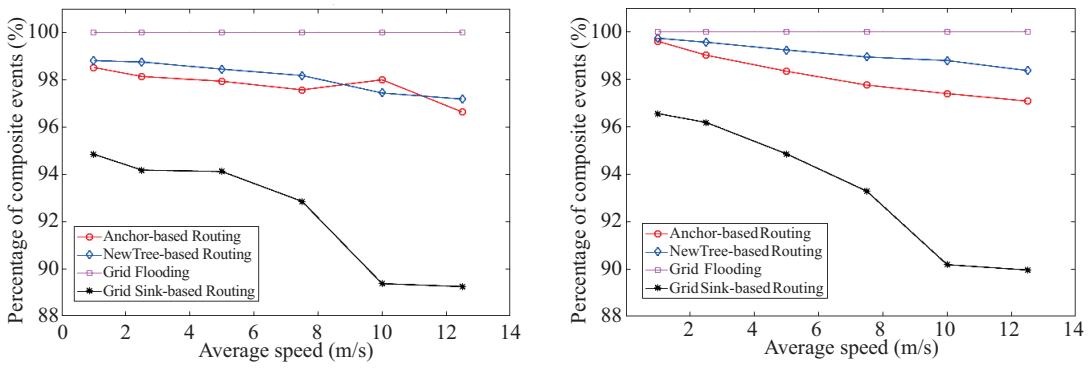

Fig. 6 Percentage of composite events processed successfully at the sink: (a) Small events, (b) Large events.

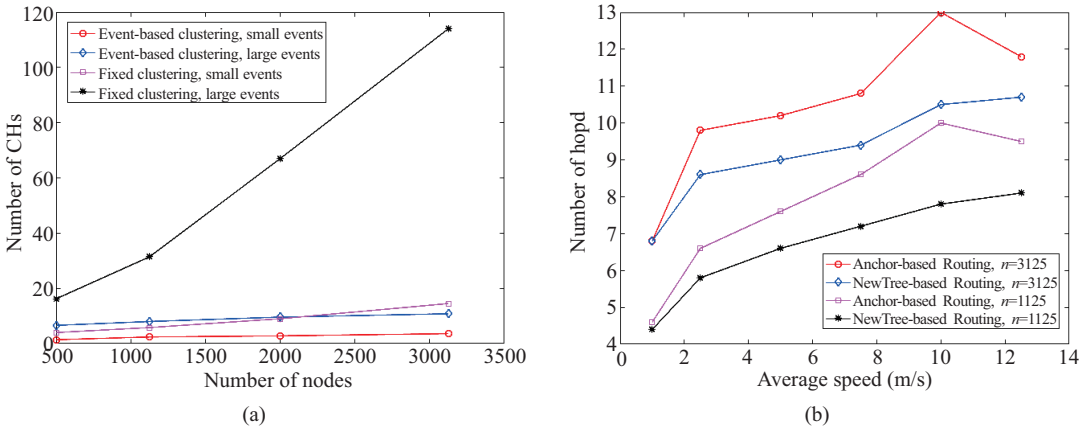

Fig. 7 (a) Average number of CHs, (b) Average number of hops to reach the sink.

Clustering. Based on the size of the event, one or more clusters are formed. This figure show comparatively the number of clusters for small and large events. As expected, more clusters are formed for large events. We also observe that the average number of clusters in Fixed Clustering is larger than those in Event-based Clustering.

Figure 7b illustrates the average number of hops between event-reporting CHs and the sink, for NewTree-based Routing and Anchor-based Routing. We considered medium size events, when the sink speed varies between 1 m/s and 12.5 m/s. We take two cases: A.L $= 660$ m, $n = 1125$, and A.L $= 1100$ m, $n = 3125$. The Anchor-based Routing has a larger number of hops compared to NewTree-based Routing. When the sink moves closer to the event, Anchor-based Routing routes the messages through the first anchor A_1, which may result in longer paths. On the other hand, a new convergecast tree is initiated by the NewTree-based Routing every time the sink moves out of R's range, thus events are reported on shorter paths. As expected, a larger network size with a larger A.L results in longer delivery paths.

7 Conclusion

This paper presents four distributed algorithms for event detection and reporting in mobile-sink WSNs. Fixed clustering algorithms, Grid Flooding, and Grid Sink-based Routing, require nodes to know their location, using GPS or running a location computation mechanism. Grid Flooding consumes the most energy, but achieves the highest percentage of composite events processed successfully at the sink. Grid Sink-based Routing consumes the least energy among the four algorithms, but achieves the lowest percentage of composite events processed successfully at the sink.

NewTree-based Routing and Anchor-based Routing are using event-based clustering, where nodes do not need to know their location. Anchor-based Routing is more energy-efficient than NewTree-based Routing, but it may result in longer paths, and has a slightly less percentage of composite events processed successfully at the sink.

Based on our results, we conclude that the network designer should select the event reporting algorithm based on the performance metrics he tries to optimize. For WSNs, nodes which have limited energy resources and using GPS is considered too expensive, the Anchor-based algorithm with a small number of anchors β is the recommended energy-efficient approach. If node energy is not a concern, and getting a high percentage of composite events processed successfully at the sink is a priority, then Grid Flooding algorithm can be used.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, Internet of Things (IoT): A vision, architectural elements and future directions, *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1645–1660, 2013.

[2] S. D. Tebje Kelly, N. Kumar Suryadevara, and S. Chandra Mukhopadhyay, Towards the implementation of IoT for environmental condition monitoring in homes, *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3846–3853, 2013.

[3] R. Fisher, L. Ledwada, G. Hancke, and C. Kruger, Open hardware: A role to play in wireless sensor networks? *Sensors*, vol. 15, no. 3, pp. 6818–6844, 2015.

[4] R. Roman, P. Najera, and J. Lopez, Securing the Internet of Things, *IEEE Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[5] M. Turkanivic, B. Brumen, and M. Hlbl, A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion, *Ad Hoc Networks*, vol. 20, no. 2, pp. 96–112, 2014.

[6] D-G. Zhang, Y-N. Zhu, Ch-P. Zhao, and W-B. Dai, A new construction approach for a weighted topology of wireless sensor networks based on local-world theory for the Internet of Things (IoT), *Computer and Mathematics with Applications*, vol. 64, no. 5, pp. 1044–1055, 2012.

[7] P. Rawat, K. Deep Singh, H. Chaouchi, and J. M. Bonnin, Wireless sensor networks: A survey on recent developments and potential synergies, *Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[8] D. Morandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, Internet of Things: Vision, applications and research challenges, *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[9] C. Aranzazu Suescun and M. Cardei, Event-based clustering for composite event detection in wireless sensors networks, in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)* (2016), 2016.

[10] WSNet—An event driven simulator for large scale wireless networks, Available: http://wsnet.gforge.inria.fr/, Accessed on Feb. 10, 2017.

[11] H. Wu, J. Cao, and X. Fan, Dynamic collaborative in-network event detection in wireless sensor networks, *Telecommunication Systems*, doi: 10.1007/s11235-015-9981-0.

[12] C. T. Vu, R. A. Beyah, and Y Li, Composite event detection in wireless sensor networks, in *IEEE International Performance, Computing, and Communications Conference*, 2007.

[13] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan,

Energy efficient communication protocol for wireless microsensor networks, in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[14] O. Younis and S. Fahmy, HEED: A hybrid, energy-efficient distributed clustering approach for ad hoc sensor networks, *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[15] M. O. Oladimeji, M. Turkey, M. Ghavami, and S. Dudley, A new approach for event detection using K-means clustering and neuronal networks, *in 2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.

[16] S. Zhang, H. Chen, Q. Zhu, and J. Jia, A fuzzy-decision based approach for composite event detection in wireless sensor networks, *The Scientific World Journal*, p. 816892, 2014.

[17] I. Memon and T. Muntean, Cluster-based energy-efficient composite event detection for wireless sensor networks, in *SENSORCOMM 2012*, 2012.

[18] A. W. Khan, A. H. Abdullah, M. H. Anisi, and J. I. Bangash, A comprehensive study of data collection schemes using mobile sinks in wireless sensor networks, *Sensors*, vol. 14, no. 2, pp. 2510–2548, 2014.

[19] K. Tian, B. Zhang, K. Huang, and J. Ma, Data gathering protocols for wireless sensor networks with mobile sinks, in *Proceedings IEEE GLOBECOM10*, 2010.

[20] Y-F. Hu, Y-S. Ding, L-H. Ren, K-R. Hao, and H. Han, An endocrine cooperative particle swarm optimization algorithm for routing recovery problem of wireless sensor networks with multiple mobile sinks, *Information Sciences*, vol. 300, pp. 100–113, 2015.

[21] L. Shi, B. Zhang, H. T. Mouftah, and J. Ma, DDRP: An efficient data-driven routing protocol for wireless sensor networks with mobile sinks, *International Journal of Communication Systems*, doi: 10.1002/dac.2315.

[22] B. Patel and D. Bhagat, Shifting of sink position in wireless sensor network, *International Journal of Engineering Development and Research*, vol. 2, no. 2, pp. 2566–2571, 2014.

[23] M. I. Khan, W. N. Gansterer, and G. Haring, Static vs mobile sink: The influence of basic parameters on energy efficiency in wireless sensor networks, *Computer Communications*, vol. 36, no. 5, pp. 965–978, 2012.

[24] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink, *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1947–1960, 2015.

[25] M. Ma, Y. Yang, and M. Zhao, Tour planning for mobile data-gathering mechanisms in wireless sensor networks, *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1472–1483, 2013.

[26] A. W. Khan, A. H. Abdullah, M. A. Razzaque, and J. I. Bangash, VGDRA: A virtual grid-based dynamic routes adjustment scheme for mobile sink-based wireless sensor networks, *IEEE Sensors Journal*, vol. 15, no. 1, pp. 526–534, 2015.

[27] M. Abo-Zahhad, S. M. Ahmed, N. Sabor, and S. Sasaki, Mobile sink based adaptive immune energy efficient clustering protocol for improving the lifetime and stability period of wireless sensor networks, *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4576–4586, 2015.

[28] H. Salarian, K-W. Chin, and F. Naghdy, An energy-efficient mobile-sink path selection strategy for wireless sensor networks, *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2407–2419, 2014.

[29] M. Marta and M. Cardei, Improved sensor network lifetime with multiple mobile sinks, *Pervasive and Mobile Computing*, vol. 5, no. 5, pp. 542–555, 2009.

[30] J. Gao, J. Li, Z. Cai, and H. Gao, Composite event coverage in wireless sensor networks with heterogeneous sensors, in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[31] Waspmote—Open source sensor node, Available: http://www.libelium.com/products/waspmote/, Accessed on Feb. 10, 2017.

[32] M. Marta Y. Yang, and M. Cardei, Energy-efficient composite event detection in wireless sensor networks, in *International Conference on Wireless Algorithm*, 2009.

[33] Y. Yang, A. Ambrose, and M. Cardei, Coverage for composite event detection in wireless sensor networks, *Wireless Communications and Mobile Computing*, vol. 11, no. 8, pp. 1168–1181, 2010.

Mihaela Cardei is a professor with the Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, Florida, USA. She received the PhD and MS degrees in computer science from the University of Minnesota, Twin Cities, in 2003 and 1999, respectively. Her research interests include wireless networking, wireless sensor networks, network protocol and algorithm design, and resource management in computer networks. She has over 90 publications with over 6300 citations on Google Scholar, and 3 Best Paper awards. Dr. Cardei is a recipient of an NSF CAREER Award and the recipient of the 2007 Researcher of the Year Award at Florida Atlantic University. She is a senior member of IEEE.

Catalina Aranzazu-Suescun is a PhD candidate at the Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, Florida, USA. She received the MS degree in telecommunications from the Pontificia Bolivariana University and the bachelor degree in electronic engineering from the University of Antioquia, Medellin-Colombia, in 2011 and 2007, respectively. Her research interests include wireless sensor networks, communication protocols, optimization of algorithms, and networking planning. She has 5 publications and has received the Best Paper Award at IEEE IPCCC 2016. She is an IEEE student member.