EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

A virtual infrastructure based on honeycomb tessellation for data dissemination in multi-sink mobile wireless sensor networks

Ayşegül Tüysüz Erman*, Arta Dilo and Paul Havinga

Abstract

A new category of intelligent sensor network applications emerges where motion is a fundamental characteristic of the system under consideration. In such applications, sensors are attached to vehicles, or people that move around large geographic areas. For instance, in mission critical applications of wireless sensor networks (WSNs), sinks can be associated to first responders. In such scenarios, reliable data dissemination of events is very important, as well as the efficiency in handling the mobility of both sinks and event sources. For this kind of applications, reliability means real-time data delivery with a high data delivery ratio. In this article, we propose a *virtual infrastructure* and a *data dissemination protocol* exploiting this infrastructure, which considers dynamic conditions of multiple sinks and sources. The architecture consists of '*highways*' in a *honeycomb tessellation*, which are the three main diagonals of the honeycomb where the data flow is directed and event data is cached. The highways act as rendezvous regions of the events and queries. Our protocol, namely *hexagonal cell-based data dissemination (HexDD)*, is fault-tolerant, meaning it can bypass routing holes created by imperfect conditions of wireless communication in the network. We analytically evaluate the communication cost and hot region traffic cost of HexDD and compare it with other approaches. Additionally, with extensive simulations, we evaluate the performance of HexDD in terms of data delivery ratio, latency, and energy consumption. We also analyze the hot spot zones of HexDD and other virtual infrastructure based protocols. To overcome the hot region problem in HexDD, we propose to resize the hot regions and evaluate the performance of this method. Simulation results show that our study significantly reduces overall energy consumption while maintaining comparably high data delivery ratio and low latency.

1 Introduction

Based on recent technological advances in wireless communication, low-power microelectronics integration and miniaturization, the manufacturing of a large number of low cost wireless sensors became technically and economically feasible. Wireless sensors are constrained devices with relatively small memory resource, restricted computation capability, short range wireless transmitter-receiver and limited built-in battery. Hundreds or thousands of these devices can potentially be networked as a wireless sensor network (WSN) for many applications that require unattended, long-term operations. Consequently, WSNs have emerged as a promising technology with various applications, such as activity recognition [1], intrusion detection [2], structural health monitoring [3], disaster management, etc.

In all these applications, the primary goal of a WSN is to collect useful information by monitoring phenomena in the surrounding environment. Common sensing tasks are heat, pressure, light, sound, vibration, presence of objects, etc. In WSNs, each sensor individually senses the local environment, but col-laboratively achieves complex information gathering and dissemination tasks. Typically a WSN follows the communication pattern of convergecast, where sensors *-source* nodes- generate data about a phenomenon and relay streams of data to a more resource rich device called *sink*. This procedure is called *data dissemination*, which is a preplanned way of distributing data and queries of sinks among the sensors.

Traditional static WSN systems use a *n-to-*1 communication paradigm in which sensors forward their data

* Correspondence: a.tuysuz@utwente.nl
Pervasive Systems Research Group, Department of Computer Science, University of Twente, Enschede, The Netherlands

towards a common static sink. However, deploying one static sink limits the network lifetime as the close neighbors of the sink can become the bottlenecks of the network. Multiple sinks deployment helps to spread load over the network, while mobility of sinks reduces the bottleneck problem of static sinks. Exploiting *multiple*, *mobile* sinks in a WSN, instead of static ones, is thus an interesting concept to enhance the network lifetime by avoiding excessive transmission at the nodes that are close to the location of the static sink.

The study presented in this article is motivated by disaster management scenarios where we have a mobile multi-sink WSN in which the deployment of sensors is performed in a random fashion, e.g., dropping sensors from helicopters flying above the field [4]. As shown in Figure 1, in this mobile multi-sink WSN, unmanned aerial vehicles (UAVs), emergency responders, e.g., firefighters, or vehicles, e.g., firetrucks, carry *sink* nodes onboard. These mobile sinks are used to collect more reliable data about the event in the dangerous/inaccessible regions. In this scenario, both the number of sources and that of mobile sinks may vary over time. The speed of sources and sinks also vary from a typical pedestrian to a flying UAV.

Sink mobility brings new challenges to data dissemination in WSNs. Since the location of the sink changes in time, the difficulty for sensor nodes is to efficiently track the location of the mobile sink to report the collected measurements about the event. Although several data dissemination protocols have been proposed for sensor networks, e.g., Directed Diffusion [5], they all suggest that each mobile sink needs to periodically flood its location information through the sensor field, so that each sensor is aware of the sink location for sending future events and measurements. However, such a strategy leads to increased congestion and collisions in the wireless transmission and is thus mainly suited for (semi) static setups.

Flat networks, where each node typically plays the same role, and flooding-based protocols do not scale due to frequent location updates from multiple sinks. Therefore, overlaying a virtual infrastructure over the

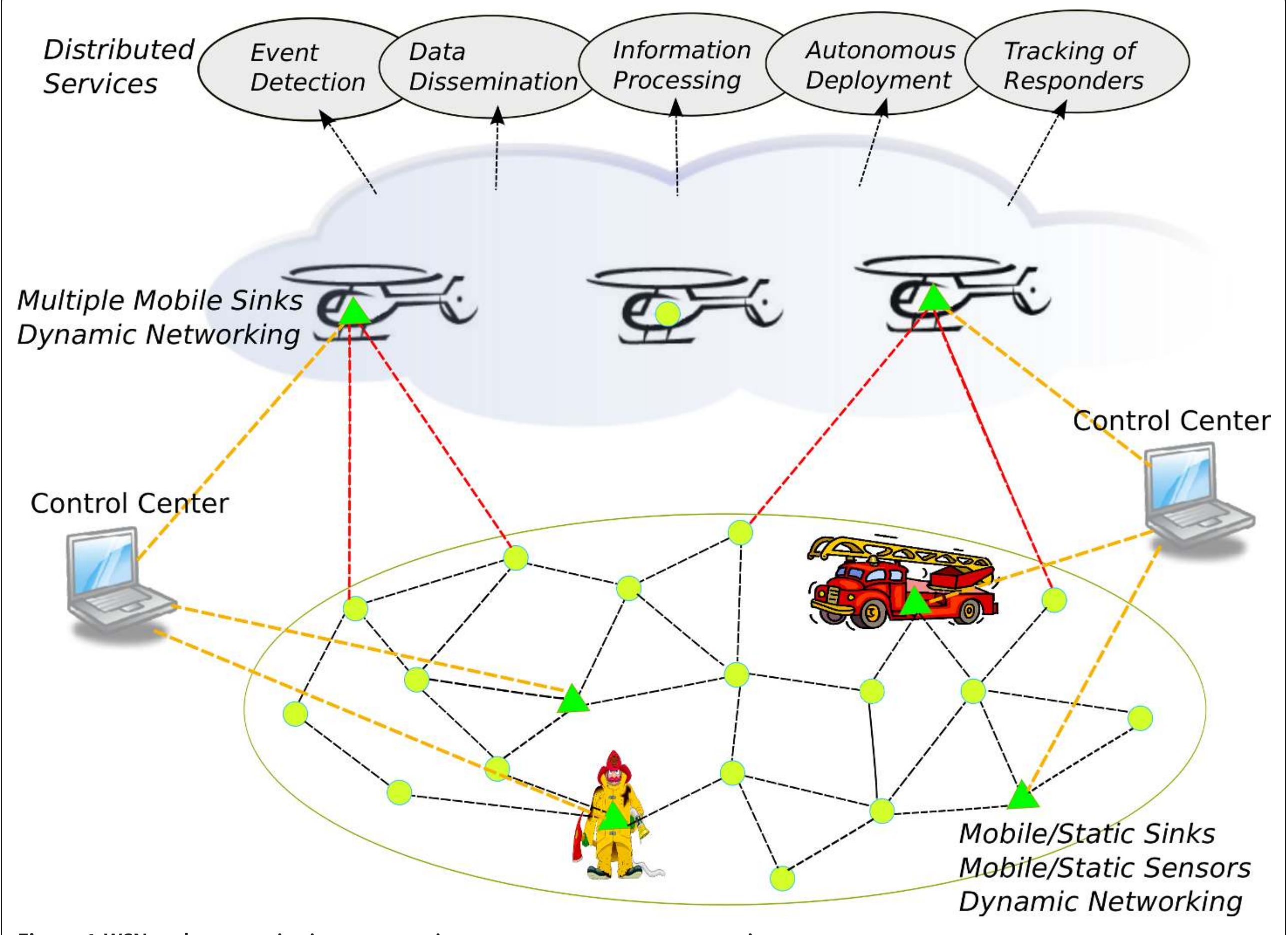


Figure 1 WSN and communication structure in an emergency response scenario.

physical network has been investigated as an efficient strategy for data dissemination towards mobile sinks [6]. In this article, we investigate the use of virtual infrastructures to support mobile sinks in WSNs. Once a virtual infrastructure is overlaid onto the physical network, it acts as a *rendezvous* region for storing and retrieving collected event data. Sensor nodes in the rendezvous region store the generated data during the absence of the sink. When the mobile sink crosses the network, the sensors in the rendezvous region are queried to notify of the event data.

We first present the advantages and challenges of using mobile sinks in WSNs. Next, we introduce our virtual infrastructure based on honeycomb tessellation and the protocol based on it, *hexagonal cell-based data dissemination (HexDD)*. HexDD is a geographical routing protocol based on this virtual infrastructure concept, proposing rendezvous regions for events (data caching) and queries (look-up). It is designed to improve network performance in terms of data delivery ratio and latency, besides meeting the traditional requirements of WSNs, such as energy efficiency.

In contrast to the rich literature on virtual infrastructure based data dissemination, especially those using greedy forwarding (GF) to send data from sources to rendezvous region, in our previous study [7] we proposed to forward data generated by sources along predefined regions called *highways*, which are the rendezvous regions in HexDD. The main contribution of this article is to improve our data dissemination protocol, HexDD with a fault-tolerance mechanism that does not require additional networking overhead, such as extra messaging to find alternative paths. The following are the key highlights of this study:

(i) We discuss the advantages and challenges of mobile sinks and present a review of existing virtual infrastructure based data dissemination protocols for mobile multi-sink WSNs.
(ii) We present our previously proposed HexDD protocol that accommodates the dynamics of the WSN such as stimulus and sink mobility, in such a way that it avoids excessive updates caused by frequently changing environment.
(iii) We enhance the HexDD protocolby proposing a complete fault-tolerance algorithm that detects routing holes, and calculates and establishes alternative forwarding paths.
(iv) We evaluate analytically the communication cost and hot region traffic cost of HexDD and compare it with other approaches.
(v) We evaluate the performance of HexDD with extensive simulations in NS2, and present a large study of comparisons with two other virtual infrastructure based protocols. The protocols with different virtual infrastructures allow us to study the effects of the virtual infrastructure shape and the data dissemination strategy on the networking performance.
(vi) We show the "hot spot" regions (i.e., heavily loaded nodes around rendezvous areas) that are created by different virtual infrastructure based protocols. We present a method for resizing of rendezvous region in HexDD to alleviate hot spot problem in the network.

The highlights (i), (iii), (iv), and (vi) are extensions to our previous studies [7,8] while the treatment of all (i)-(vi) in this article provides a comprehensive discussion of the protocol. The rest of this article is organized as follows: The related studies are introduced with their strengths and weaknesses in Section 2. Section 3 motivates the use of mobile sinks in WSNs. Section 4 introduces the honeycomb tessellation and HexDD protocol. Section 5 provides analytical studies of communication cost and hot spot traffic cost of HexDD. Section 6 presents the simulation results to evaluate the performance of the proposed protocol in comparison with existing protocols. Finally, Section 7 draws the conclusions.

2 Related work

2.1 Mobility patterns and data collection strategies

Sink mobility can be classified as *uncontrollable* or *controllable* in general. The former is obtained by attaching a sink node on a mobile entity such as an animal or a shuttle bus, which already exists in the deployment environment and is out of control of the network. The latter is achieved by intentionally adding a mobile entity e.g., a mobile robot, into the network to carry the sink node. In this case, the mobile entity is an integral part of the network itself and thus can be fully controlled [9].

Different sink mobility patterns provide different data gathering mechanisms ranging from single hop passive communication (i.e., direct-contact data collection), which may require controllable sink mobility, to multi-hop source to sink solutions, which can be achieved by uncontrollable or controllable sink mobility.

Direct-contact data collection has great advantage for energy savings. That is, sinks visit (possibly at slow speed) all data sources one by one and obtain data directly from them. This data collection strategy needs intelligent sink movement computed as the best sink trajectory that covers all data sources and minimizes data collection delay [10]. With this approach, maximum energy efficiency and longest network lifetime is achieved at the expense of long delays. This mobility scheme is feasible for delay tolerant applications.

Rendezvous-based data collection is proposed to achieve a good trade off between energy consumption and time delay. Sensors send their measurement to a subset of sensors called *rendezvous points* (RPs) by multi-hop communication; a sink moves around the network and retrieves data from encountered RPs. The use of RPs enables the sink to collect a large volume of data with an energy cost of multi-hop data communication, and at a time without traveling a long distance. Thus, the use of RPs greatly decreases data collection delay. If the virtual infrastructure of rendezvous-based protocol is well designed, one can achieve scalability and energy efficiency. Rendezvous-based data collection can be used when we have uncontrollable (e.g., random) sink movement in a WSN.

2.2 Data dissemination protocols

Several data dissemination protocols have been proposed for WSNs with mobile sinks. The proposed protocols fall in two major categories: (i) *Flooding-based* and (ii) *Virtual infrastructure-based*. In general, virtual infrastructure-based protocols can be divided into (i) *backbone-based* approaches (e.g., [11]), and (ii) *rendezvous-based* approaches (e.g., [12]) depending on how the virtual infrastructure is formed by the set of potential storing nodes. All protocols discussed in this section assume uncontrolled mobility in the network.

Directed diffusion [5] is a flooding-based approach introducing data-centric routing for sensor networks. In this approach, each sink must periodically flood its location information through the sensor field. This procedure sets up a gradient from sensor node to the sink node, so that each sensor becomes aware of the sink's location for sending future data. Although directed diffusion solves the problem of energy-efficiency by using several heuristics to achieve optimized paths, its flooding-based approach does not scale with the network size and increases the network congestion.

Pursuit-evasion games (PEG) [13] is a sensor network system that detects an uncooperative mobile agent, *evader*, and assists an autonomous mobile robot called the *pursuer* in capturing the evader. The routing mechanism used in PEG, namely *landmark routing*, uses the node at the center of the network as landmark (i.e., only one RP) to route packets from many sources to a few sinks. It constructs a spanning tree having the landmark node as the root of the tree. For a node in the spanning tree to route an event to a pursuer, it first sends the data up to the root, the landmark. The landmark, then, forwards the data to the pursuer. The pursuer periodically informs the network of its position by picking a node in its proximity to route a query to the landmark. Since data dissemination used in PEG is a combination of directed diffusion [5] towards the landmark and central re-dissemination, in order to build the gradients from sensors to landmark node (i.e., spanning tree), it uses flooding-based approach (i.e., each node sends a beacon packet which is further re-broadcasted by all the neighbors of the node) which results in broadcast storm problem increasing the congestion.

As the flat networks and flooding-based protocols do not scale, overlaying a virtual infrastructure over the physical network often has been investigated as an efficient strategy for data dissemination in mobile WSNs [6]. This strategy uses the concept of virtual infrastructure, which acts as a rendezvous area for storing and retrieving the collected measurements. The sensor nodes belonging to the rendezvous area are designated to store the generated measurements during the absence of the sink. After the mobile sink crosses the network, the designated nodes are queried to report the sensory input. The concept of overlaying a virtual infrastructure over the physical network has several advantages. The infrastructure acts as a rendezvous region for the queries and the generated data. Therefore, it enables the gathering of all of the generated data in the network and permits the performing of certain data optimizations (e.g., data aggregation) before sending the data to the destination sink [6]. Second, in WSNs deployed in harsh environments, source nodes can be affected by several environmental conditions (e.g., wildfire, etc.), and therefore, the risk of losing important data is high. To ensure the persistence of the generated data, the source node can disseminate the data towards the rendezvous area instead of storing it locally. Thus, the virtual infrastructure enables data persistence against node failures. Main disadvantage of using a virtual infrastructure is the creation of hot spot regions in the network. However, it is possible to solve this problem by adjusting the size of rendezvous regions. Several protocols that implement a rendezvous-based virtual infrastructure have been proposed in the literature. They vary in the way they construct the virtual infrastructure. In the rest of this section, we summarize these protocols.

The geographic hash table (GHT) [14], which is illustrated in Figure 2a, introduces the concept of data-centric routing and storage. GHT hashes keys into geographic coordinates, and stores a key-value pair at the sensor node geographically nearest the hash of its key. In GHT, the data report type is hashed into geographic coordinates, and the corresponding data reports are stored in the sensor node, called *home-node*, which is the closest to these coordinates. This *home-node* acts as a rendezvous node for storing the generated data reports of a given type. There are as many *home nodes* as data types. The main drawback of this approach is the hot spot problem because all data reports and queries for the same meta-data are concentrated on the

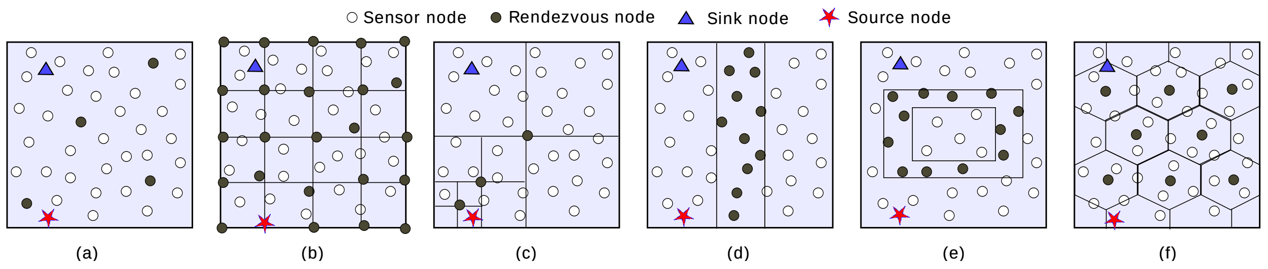


Figure 2 Virtual infrastructure-based data dissemination protocols: (**a**) GHT (hashed location), (**b**) TTDD (grid structure), (**c**) QDD (quad-tree structure), (**d**) LBDD (line-based structure), (**e**) RailRoad (rail structure), (**f**) GCA (hexagonal clustering).

same *home node*. This may restrict the scalability and the network lifetime.

In two-tier data dissemination (TTDD) [15], each source node proactively builds a uniform virtual grid structure throughout the sensor field, as shown in Figure 2b. A sink floods a query within its local grid cell. The query packet then propagates along the grid to reach the source node. While the query is disseminated over the grid, a reverse path is established towards sink and data is sent to the sink via this reverse path. If the stimulus is mobile, number of sources and grids increase. This situation can lead to excessive energy drain, and therefore, limit the network lifetime.

Quadtree-based data dissemination (QDD) [16] protocol defines a common hierarchy of data forwarding nodes created by a quadtree-based partitioning of the physical network into successive quadrants, as shown in Figure 2c. In this approach, when a source node detects a new event, it calculates a set of RPs by successively partitioning the sensor field into four quadrants, and the data reports are sent to the nodes which are closer to the centroid of each successive partition. The mobile sink follows the same strategy for the query packet transmission. The main drawback of this approach is that some of the static nodes that are selected as RPs (e.g., central node in the deployment area) will induce a hot spot problem which may decrease the network lifetime and reliability.

Line-based data dissemination (LBDD) [17], which is proposed for mobility of sink and source nodes, defines a vertical *line* or *strip* that divides the sensor field into two equal sized parts, as shown in Figure 2d. Nodes within the boundaries of this wide *line* are called *inline nodes*. This virtual line acts as a rendezvous area for data storage and look-up. When a sensor detects a new event, it transmits a data report towards the nodes in the virtual line. This data is stored on the first inline node encountered. To collect the generated data reports, the sink sends its query toward the rendezvous area. This query is flooded along the virtual line until it arrives to the inline node that owns the requested data. From there the data report is sent directly to the sink using GF. Using a line as rendezvous area at the middle of the network can results in high latency for the nodes near the boundary of the network.

RailRoad [12] places a virtual *rail* in the middle of the deployment area, as shown in Figure 2e. When the source node generates data, the generated data is stored locally, whereas corresponding meta-data (i.e., event notification) is also forwarded to the nearest node inside the rail. When a sink node wants to collect the generated data, a query message is sent into the rail region. This message travels around the rail. When it reaches the rail node that stores the relevant event notification, the rail node sends a query notification message to the source node. Finally, source node sends data directly to the sink using GF.

Geographical cellular-like architecture (GCA) [11], which is a backbone-based approach, defines a hierarchical hexagonal cluster architecture that basically adopts the concept of home-agent used in cellular networks. Each cluster is composed of a *header* positioned at the center of the hexagonal cell and *member sensors*, as presented in Figure 2f. The mobile sink sends its query to the cell header that sink belongs to. The query packet then is propagated to all cell headers. When the sink moves to another cell, it registers to the new cell's header and also informs its old cell header (home-agent) about its new header's position. The data packets still are propagated towards the home-agent, which further forwards the packet to the sink's new header. In case of sink mobility, GCA results in inefficient (non-optimal) routing path which may increase the data delivery latency.

The hierarchical cluster-based data dissemination protocol (HCDD) [18] defines a hierarchical cluster architecture to maintain the location of mobile sinks and to find paths for the data dissemination from the sensors to the sink. Unlike GCA, HCDD does not require powerful position aware nodes. Each cluster is composed of a cluster head, several gateways, and ordinary sensors. When a mobile sink crosses the network, it

Table 1 Comparisons of virtual infrastructure-based data dissemination protocols for WSNs with mobile sinks

Year	GHT (2002)	TTDD (2002)	QDD (2006)	LBDD (2008)	RailRoad (2005)	GCA (2005)	HCDD (2006)	HexDD (2010)
Position awareness	+	+	+	+	+	+	-	+
Virtual infrastructure	hashed location	grid	quad-tree	line/strip	rail	clusters	clusters	highways
Disseminated information	data	data	data	data	meta-data	data	data	data
Data reportslocation	1 node	1 node	1 out of N nodes	1 out of N nodes	1 out of N nodes	1 node	1 out of N nodes	1 out of N nodes
Routing towards RPs	GF	GF	GF	GF	GF	GF	tree-based routing	honeycomb addressing based routing
Routing hole recovery	-	-	-	-	+	-	-	+
Metric of interest	energy	energy	Energy reliability	energy	energy	energy	energy	Energy reliability latency

registers itself to the nearest cluster head. Then a notification message is propagated to all cluster heads. During this procedure, each cluster head records the sink ID and its sender such that the transmission of future data reports can be performed easily from sources to sink.

Table 1 shows a classification of the existing data dissemination protocols, which support multiple, mobile sinks and how HexDD differs from these existing works. All *rendezvous based* approaches use greedy geographic routing (i.e., GF). Greedy geographic routing is attractive in WSNs due to its efficiency and scalability. However, greedy geographic routing may incur long routing paths, and even fail due to routing holes on random network topologies. Most of the previous studies do not discuss how to maintain the virtual infrastructure if there are holes, a large space without active sensors, which is a common behavior in any real WSN deployment. To recover from the local minima, GPSR [19] and GOAFR [20] route a packet around the faces of a planar subgraph extracted from the original network, while limited flooding is used in [21] to circumvent the routing hole. Unfortunately, the recovery mode inevitably introduces additional overhead and complexity to geographic routing algorithms. The main problem of the *backbone-based* approach is the need to maintain the structure. In addition, the hot spot problem may occur as the traffic is concentrated over a group of cluster headers.

Most of the previous studies do not focus on reliable and real-time data dissemination in mobile sensor networks. To handle dynamic environments efficiently and reliably, we introduce a rendezvous-based data dissemination protocol, namely HexDD, which uses hexagonal cells for geographic routing and provides a fault tolerance mechanism to deal with imperfect conditions of real deployments. To bypass routing holes, we present a simple hole recovery mechanism which avoids to flood any other control message to find new bridge nodes. The hole recovery mechanism tries to find the shortest path to recover holes; therefore, it decreases latency and increases reliability of the data dissemination, as shown in Section 6. In Section 6, it is also shown that in WSNs, where there is no hole, the proposed protocol achieves a high data delivery ratio, low data delivery delay, and low energy consumption and outperforms the existing approaches in these metrics. Moreover, in Section 5 we analyze analytically and show that the communication cost of HexDD is lower than other approaches.

3 Motivating scenario: why mobile sinks?

Sink mobility assumption may be useful for numerous applications. A typical application scenario is *emergency response*. As shown in Figure 1, sensors are randomly deployed by UAVs to monitor the area of interest, e.g., a forest in a fire fighting scenario, and detect dangerous events, e.g., fire in forest. Detection of such events is realized by event-detection algorithms, e.g., [22]. Sensors report an alarm (including data about the current situation of the event) to mobile sinks. Mobile sinks monitor the progression of the event and take the appropriate actions (e.g., sending location of the fire to the mission coordinators via a satellite). Therefore, the sink represents an important component of WSN as it acts as a gateway between the sensor network and the end-users.

The sink mobility assumption can be enforced by the nature of the employed application. For example, in the fire fighting scenario, the mobile entities (e.g., firefighters, firetrucks, UAVs, etc.) of the network have other primary tasks. Firefighters fight cooperatively to eliminate fire in the fire field, while UAVs are responsible for transport load (e.g., water) near the fire field or deploy sensors to inaccessible areas of the network. Their mobility is regulated according to their primary tasks. In the meanwhile, they are informed by the source nodes about the current situation of the event as they carry sink nodes onboard. The firefighters are warned about the dangerous situation around them in time, the spread of the fire, i.e., where it is spreading and how quickly. Therefore, from data collection point of view, the sink mobility is uncontrollable. Sinks move randomly around the network and get data from the sources. Moreover, in emergency response scenarios, the use of mobile objects for data collection makes harder the damage of such component. Indeed, if a static sink is located in the area of interest, it can be damaged by the dangerous event such as fire, thus making the sensors disconnected from the end-users. The mobile sinks enable a more reliable data collection in the dangerous/inaccessible regions.

4 Honeycomb tessellation and HexDD protocol

In this section, we describe how the physical network is partitioned into virtual hexagonal cells by the honeycomb architecture (see Figure 3), and how this architecture is employed by the geographical routing HexDD. Individual sensor nodes in the network are bound to cells of the virtual hexagonal tessellation based on their geographic locations. The architecture also defines three principle diagonal lines–'highways' (or 'border lines')–which divide the sensor field into six parts. The lines, which intersect at the center of the network, constitute the rendezvous region for queries and data.

Division of the sensor field into a regular tessellation is energy efficient compared to other schemes such as Voronoi diagram division [23]. The construction of Voronoi diagram consumes high energy in resource constrained sensor nodes. Instead of square tessellation, which is used in many protocols [15,24], we use a honeycomb tessellation for the homogeneous neighborhood it provides, i.e., all neighbors of a cell share an edge with the cell, no neighboring cells that share only a corner.

Hexagonal cells are used in literature for various applications [11,25,26]. Here, we use hexagonal cells only for the purpose of geographical routing towards a region. Differently from [25], where the hexagonal grid defines the topology of the network, meaning a sensor node in each corner of the grid, we do not assume a regular topology but a random deployment.

Creating of the architecture and our routing protocol require knowledge of location. We assume that sensor nodes are location-aware and also know the network boundaries, as it is also assumed in [11-17]. The location information can be obtained either by GPS-free localization mechanisms [27,28] or by means of a virtual coordinate system [29] during the network initialization phase. Two sensors can communicate when they are within a distance R of each other, called the communicable distance. We assume that the radio range R is the same for all nodes. Through periodic interactions (beacon packets), a sensor node can learn the location and cell of its neighbors. Sensor nodes are mainly static, and there are multiple sinks moving randomly in the sensor field. Sinks are equal from the information point of view; it does not matter to which sink a data packet is sent.

In the following, we introduce the operations of HexDD protocol. The first phase is hexagonal cell-based network partitioning, which establishes the architecture, i.e., honeycomb cells and rendezvous areas are formed. This phase is performed in the network setup. After this setup, the network becomes ready to execute the HexDD protocol.

4.1 Hexagonal cell-based network partitioning

Honeycomb architecture overlays a virtual honeycomb over the sensor field as shown in Figure 4a. In the honeycomb tessellation, each cell has six neighbors covering the surroundings from all directions. For two adjacent cells, every sensor node in one cell can communicate with all the nodes in the other cell. This defines the edge length of the hexagonal cell.

As illustrated in Figure 4b, the longest distance between two adjacent cells is $l_{|AB|} = \sqrt{13r}$, where r is

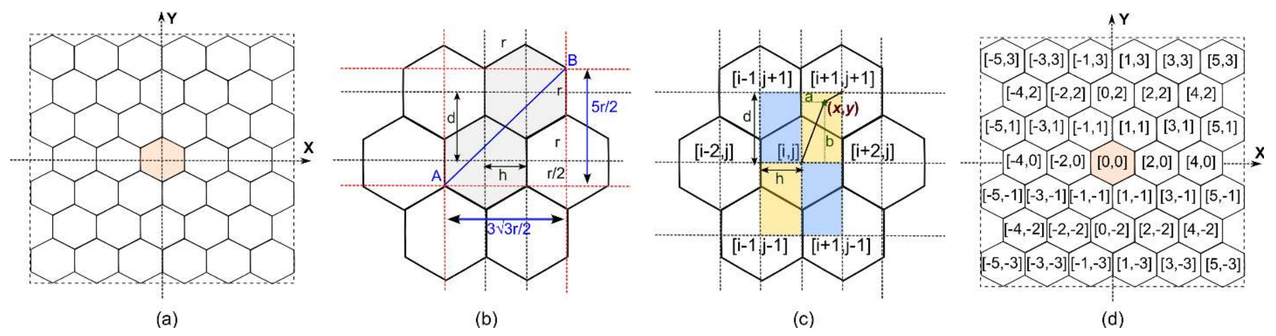


Figure 3 Cell addressing in honeycomb architecture.

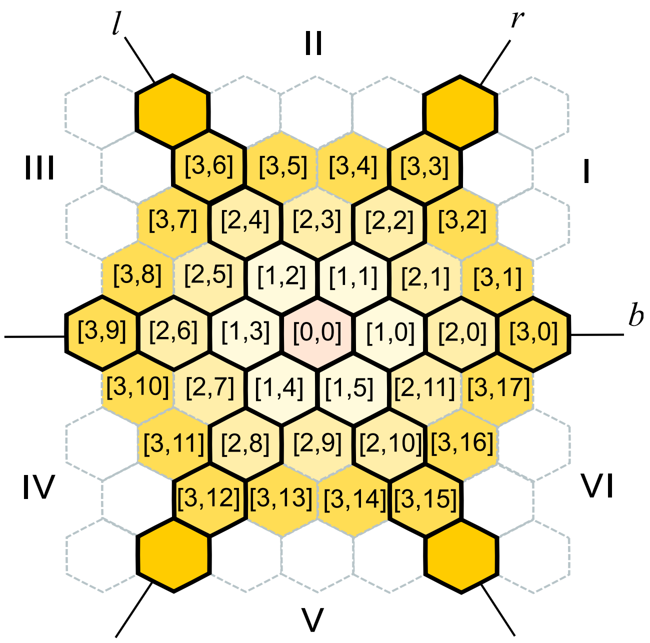


Figure 4 Honeycomb tessellation construction: (**a**) Hexagonal tessellation of the deployment area, (**b**) cell structure, (**c**) node-cell association, and (**d**) cell naming.

the edge length of the hexagon. In order for all nodes in two adjacent cells to be able to communicate with each other, the longest length must satisfy $l_{|AB|} = \sqrt{13}r \leq R$ where R is the transmission range. Therefore, we choose the edge length of the hexagon, $r_{\max} = R/\sqrt{13}$, such that sensors in adjacent cells are within *communicable distance* of each other.

In the honeycomb architecture, a hexagonal cell placement and *node-cell association scheme* needs to be established. In this scheme, hexagonal virtual cells' central points are positioned according to Figure 4c. Apparently, $d = \frac{3}{2}r$ and $h = \frac{\sqrt{3}}{2}r$, where r is the edge size of the hexagonal cell. Each virtual cell center is located at $(i \cdot d, j \cdot h)$ where i and j are integers. A virtual cell centered at $(i \cdot d, j \cdot h)$ is named as the cell $[i,j]$. Figure 4c shows the cell $[i,j]$ and its neighboring cells with their associated names in the XY coordinate system. Figure 4d shows the cell naming in honeycomb architecture.

At the first step, with the given hexagonal edge length, r, each sensor node uses its location information to associate itself with a virtual cell having a name of $[i,j]$. For the *node-cell association* (see our previous study [7] for details), we have used a similar geometrical approach as in [26]. For a node positioned at point (x, y), let $i = \lfloor x/h \rfloor$ and $j = \lfloor y/d \rfloor$. If $i + j$ is even (i.e., the node is in the yellow rectangle in Figure 4c), the node is either in cell $[i,j]$ or in cell $[i + 1,j + 1]$; if $i + j$ is odd (i.e., the node is in the blue rectangle in Figure 4c), the node is either in cell $[i + 1, j]$ or in cell $[i,j + 1]$ depending on which center is closer. Each sensor node uses its coordinates to associate itself with a hexagonal cell. There is no communication overhead since each node executes the algorithm locally.

Next, we transform the cell names of the form $[i, j]$ into special cell addresses of the form $[H,I]$. This addressing is used in the data dissemination. Figure 3 shows the cell addressing in honeycomb architecture. We assign addresses of the form $[H, I]$ to each sensor in the same cell, where H is the shortest cell-count of the node from the origin cell and I denotes the index of the hop-H hexagonal cell. The index starts at the right side of *line b* in Figure 3 and increases in the counter-clockwise direction. Hence, the nodes in the first-hop cells are addressed as [1,0], [1,1],..., [1,5]. Observe that nodes of the form $[H,.]$ are all located on the same hexagonal ring at distance H form the center cell. Since the number of cells on H^{th} hop hexagonal ring is $6 \times H$, the cell addresses range from $[H, 0]$ to $[H,6H\text{-}1]$.

To build $[H,I]$ addresses from $[i,j]$ naming, we use the transformation rules of Table 2. This special addressing has useful properties that allows simple calculations for the packet flow towards the rendezvous regions. In the honeycomb architecture, we classify the sensor nodes into two groups; (i) *border nodes* and (ii) *regular nodes*, according to their positions (cell addresses) on the honeycomb tessellation.

Table 2 Transformation rules from cell name [*i, j*] to cell address [*H,I*]

Hextant number	Condition	Transformation
1	$i > j, j \geq 0$	$[i, j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, j]$
2	$\lvert i\rvert \leq \lvert j\rvert, j > 0$	$[i,j] \Rightarrow [\lvert j\rvert, 2H-(i + j)\lvert 2]$
3	$\lvert i\rvert \geq \lvert j\rvert, j > 0$	$[i,j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, 3H-j]$
4	$\lvert i\rvert > \lvert j\rvert, j \leq 0$	$[i, j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, 3H + \lvert j\rvert]$
5	$\lvert i\rvert \leq \lvert j\rvert, j < 0$	$[i, j] \Rightarrow [\lvert j\rvert, 5H + (i + j)/2]$
6	$i \geq j, j < 0$	$[i,j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, 6H-\backslash j\backslash]$

Definition 1*: All the cells addressed as [H,I] are 'border cells'if $I = q{\cdot}H$, where $q \in \{0, ..., 5\}$. The nodes associated with border cells are called 'border nodes'. All the other notes are called 'regular nodes'.*

In the following we count the border lines using the value of *q*.

The honeycomb architecture defines three principle diagonals covering the cells on the lines labeled *l*, *b*, and *r*, which are passing through the center cell, as illustrated in Figure 3. The cells on these diagonal lines are called *border cells*. Each half line that starts from the center cell is called *border line*. These lines divide the sensor field into six regions, called *hextants.*

Definition 2: *A 'hextant'is made up of cells satisfying the condition $q{*}H \leq I < (q + 1){*}H$ whereqe $q \in \{0,...,5\}$.*

The first border line is a part of the hextant. The *hextant number* of a cell a[*H,I*] is calculated by $q + 1$, where $q = \lfloor I/H \rfloor$. This means that *q* value of all the cells in the same hextant (including the first border line) are the same. Each of six *hextants* is marked with roman numerals in Figure 3. The three diagonal lines act as rendezvous regions for data storage and look-up. Each half line, namely *border line*, is the rendezvous area for the hextant which starts at this border line, assuming a counter-clockwise direction (see Figure 5a).

4.2 Hexagonal cell-based data dissemination

In the proposed data dissemination protocol, we use the concept of central re-dissemination in which the packets flow towards the center cells following previously selected directions. Instead of sending packets directly to the center cell by using a simple geographic routing, we send data through border lines towards the center cell. The aim is to store the generated data reports in the border lines so that the mobile sinks can easily collect them using a query-based data reporting method. However, our approach is purely geographical, which means that we do not use flooding for route setup. The only required information is the node position which is associated with a hexagonal cell in the honeycomb architecture.

Before introducing our HexDD, we first give some important properties of hexagonal tessellation and addressing. Let $k = \lceil I/H \rceil$, thus $k \in \{1,...,6\}$. Inside a hextant, *k* equals the hextant number: $k = q + 1$. In a border line, i.e., cells satisfying $I = q \cdot H$, we have $k = q$. HexDD performs the forwarding of messages (data and query) following border lines and parallel directions to border lines (see Figure 5a). When inside a hextant, the message flows in a direction parallel to the second border line, and once reaching the first border line it continues flowing along that border line. Two neighbor cells a[H,I] and na[H_n,I_n] in the q^{th} border, such that $H = H_n + 1$, satisfy the relation $I = I_n + q = I_n + k$. Two neighbor cells within hextant $q + 1$, such that $H = H_n +1$, satisfy the relation $I = I_n+q+1 = I_n+k$. A flow starting from a cell s[H, I] in hextant q+1 follows the parallel direction with the second border line until it hits the first border line at cell b[H', I'] with $H' = (q+1){*}H\text{-}I = k{*}H\text{-}I$.

The properties of hexagonal tessellation given above are used by the routing algorithm, *Algorithm* 1. With the given virtual infrastructure, the following sections explain the operations of HexDD.

4.2.1 Event data forwarding

Event data forwarding in HexDD is done through border nodes towards center region according to *Algorithm 1-I*. Line 5 of *Algorithm 1* calculates the hextant number *k* of the current cell of the node which has the data packet. Line 6, then, determines the next cell to forward the data packet. To find next hop, *H* of current cell is reduced by one because the packet will be forwarded to the cell which is 1-hop closer to the center and *I* is reduced by *k* since the difference between *I*s of two adjacent cells on the packet forwarding direction of a hextant is equal to *k* for all hextants. As shown in Figure 5a with arrows, sensors route the packets to border cells in the first line segment of the hextant, e.g., line *r* for hextant II, following a direction parallel to the second border line of the hextant, e.g., line *l* for hextant II. When the data reaches one of the diagonal lines, it is forwarded along the border line towards the center cell.

Sensors in the border lines act as RPs for data storage and look-up which means border nodes have a replica of data in their cache. When a sensor on the border line receives a new data packet from a source node, it updates its record with the new data so it keeps the most up-to-date data packet. Another option can be logging all the data in the border nodes from the beginning of the event; however, this requires a lot of memory.

To facilitate the data lookup process, two replication schemes are possible in the border lines: the data can be either stored in all nodes of hexagonal

Algorithm 1 Hexagonal cell-based data dissemination

```
1: Input: $[H, I]$, address of the current cell
2: Input: $[H_s, I_s]$, address of the sink's current cell
3: Output: $[H,I]$, address of next hop cell
```

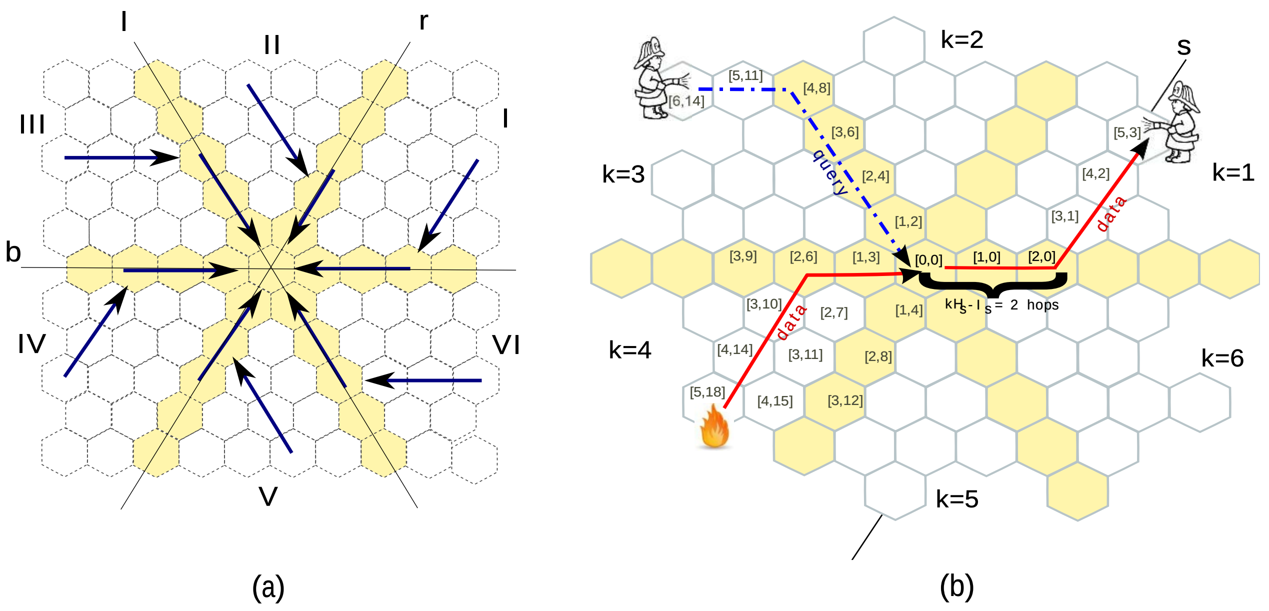


Figure 5 HexDD in honeycomb tessellation: (**a**) Packet forwarding directions, (**b**) data and query dissemination.

```
4: I. Find next hop cell towards center
5:     $k = \lceil I/H \rceil$
6:     $[H,I] \Leftarrow [H-1,I-k]$
7: II. Find next hop cell towards sink
8: $k_s = \lceil I_s/H_s \rceil$
9: $H \Leftarrow H + 1$
10: if $H <= k_s H_s - I_s$ then
11:     $I \Leftarrow I+k_s-1$ //In the border line
12: else
13:     $I \Leftarrow I+k_s$ //within the hextant
14: end if
```

cells or just in the cell-leader of each cell. The first scheme needs a fine-tuning of *border line width*, *w*, to prevent an increase of congestion under high traffic load conditions, while the second one requires a periodic cell-leader election and a replication mechanism. As in [17,30], we disregard the lines' width *w*. We assume that each border line covers only one cell (see Figure 5b).

The HexDD keeps the traffic flow in all regions of the network nearly balanced because honeycomb architecture divides the network space into six partitions and each partition uses a different border line segment for data dissemination; therefore, the traffic is spread among the different border lines.

4.2.2 Querying

In order to retrieve specific data, a sink sends a query towards the center by using *Algorithm 1-I*. The data and query packets are sent towards the center by using the same forwarding directions which are shown in Figure 5a. The first border node which receives the query forwards it towards the center cell. Each node in the border cells checks its cache when it receives a query. If the data requested is in the cache of a border node, it sends data back to the sink. Replicating data on the border cells can decrease the cost of data look-up and the data delivery latency.

4.2.3 Event data delivery to sink

To send data towards the sink, the reverse path of the sink's query forwarding path can be calculated by using the cell address of the sink as given in the *Algorithm 1-II*, or can be stored in the query packet. The forwarding directions of the data packets from center to the sinks are exactly the opposite directions of the arrows shown in Figure 5a. Line 8 of the *Algorithm 1* calculates the hextant number of the sink's current cell. Line 9 increases the *H* by one to get to the next hexagonal ring which is 1-hop closer to the hexagonal ring of the sink's cell. The data first travels in one of the border lines according to hextant number k_s of sink's cell. In line 10, *H* is compared with k_sH_s - I_s to determine the number of hops that the packet should be forwarded along the border line. Thus, the condition in line 10 ensures that the packet does not go further on the border line when it reaches the turning point towards the sink. If the packet is still on the border line, *I* is increased by k_s -1 in line 11. When the packet reaches the cell which is on the same line (i.e., line *s* parallel to line *r* in Figure 5b) where sink's cell is also located, the packet is forwarded towards the inside of the hextant. Within the hextant, *I* of the current cell is increased by k_s in line 13 until the packet reaches the cell of the sink.

Before sending data to a regular node, the algorithm always checks if there is a sink node in the next hop cell. If so, the data is sent to the sink in the next cell. Otherwise, it sends the data packet to a sensor node in the next cell until the packet reaches to a sink.

Figure 5b shows the data and query dissemination in HexDD. If there is no neighbor node to forward the packet (i.e., query or data packet) in the next 2-hop cells calculated by the *Algorithm 1*, the protocol switches to route recovery procedure explained in the following section.

4.3 Handling imperfect conditions of wireless communication

In our hexagonal tessellation construction, we consider a widely used assumption for transmission range. All sensors have the same circular transmission range, R. However, in real sensor deployed environments, radio irregularity (i.e., non-uniform transmission range and/or non-circular transmission range), which obviously affects the network connectivity, can be observed. The effect of the radio irregularity on our hexagonal tessellation based routing is that a sensor node a in cell A may not be able to communicate with some of the sensor nodes in neighboring cells if the transmission range of node a is smaller that R (i.e., $R_a < R$) or the transmission range is non-circular. In case of small difference between R_a and R, the possibility of having some links to neighboring cells (i.e., connected neighbors to node a) is higher. However, the difference between R_a and R may be high in some environments. In this case, node a cannot communicate with some of the neighboring cells or it may be disconnected from the network. Both cases create some routing holes in the network.

An other issue, which can create routing holes, is localization errors in real deployments. The hexagonal tessellation and our geographic forwarding protocol rely on each node being able to estimate its own coordinates. These estimates are highly likely affected by a non-negligible error, which in turn affects the calculated cell addressing $[H, I]$ used for packet forwarding. We use a kind of polar coordinate system to address the cells of the tessellation. This addressing scheme serves as a positioning (coordinate) system that is rougher than the coordinates of the sensor nodes, with a precision appropriate for the transition range. A localization estimate with a reasonable error $err < r$, where r is the edge length of a hexagonal cell, will result in the same cell address $[H, I]$. Therefore, the packet forwarding mechanism will not be affected by the localization errors. If a given node, which is close to the boundary of its hexagonal cell, calculates a wrong cell address due to localization error, the erroneous cell address will be one of the neighbor cells of its real cell. The localization errors may result in some empty cells or some deviations form the regular path of a packet in HexDD.

To handle routing holes and forwarding path deviations created by the imperfect conditions of the wireless environment, in the following section we present a fault tolerance mechanism, which discusses how to determine and bypass the routing holes. This fault tolerance mechanism makes our scheme more feasible in real sensor network deployments. As long as a node, which has a packet to forward, has at least one neighbor in one of the neighboring cells, HexDD combined with fault tolerance mechanism can find an alternative path towards the destination of the packet.

4.3.1 Fault tolerance

Algorithm 1 assumes that there is at least one node which will perform multi-hop routing within each cell. However, this may not be always the case. Sometimes an area of the network can be lost for different reasons, e.g., environmental reasons such as fire. Holes are created where there is a group of cells that do not have any active node inside. Moreover, the imperfect conditions of the wireless communication discussed above may also create holes in the network. In our previous study [8], we discuss possible solutions for fault tolerance. In this article, we propose and present a complete hole detection and bypassing mechanism, which is one of the most important features that shows how we maintain the honeycomb architecture even if parts of the network are lost.

A sensor can easily detect the hole region by checking its neighbor table, which is updated by periodic beacon packets. If the sensor has no neighbor on the next 2-hop cells in its radio range, it concludes that there is a hole at that area of the network. *Algorithm 2*[a] gives the details of HexDD with hole recovery.

Algorithm 2 HexDD with Hole Recovery

```
1: Input: $[H, I]$, address of the current cell
2:    $[H_s, I_s]$, address of the sink's current cell
3:    $N = \{n_1, ..., n_m\}$, list of neighbors
4:    $N_a = \{[H_1, I_1], ..., [H_m, I_m]\}$, list of cell addresses of neighbors
5: Output: $n$, next hop neighbor to forward the packet
6: I. Find next hop neighbor towards center
7: $[H_c, I_c] \Leftarrow$ Find next hop cell towards center (Alg. 1.I)
8: if $[H_c, I_c] = [H_i, I_i] \in N_a$ then
9:    $n \Leftarrow n_i$ {forward data to neighbor in next cell}
10: else {there is a hole, enter route recovery}
11:    $n \Leftarrow n_j$ with $H_j$ the smallest $H$ in $N_a$
12: end if
13: II. Find next hop neighbor towards sink
14: $k = \lceil I_s/Hs \rceil$
15: $p = I_s - (k-1)H_s$
16: if $[H, I]$ in the regular path then
```

```
17: $[H_c, I_c] \Leftarrow$ Find next hop cell towards sink (Alg. 1. II)
18: if $[H_c, I_c] = [H_i, I_i] \in N_a$ then
19:     $n \Leftarrow n_i$ {forward data to neighbor in next cell}
20: else {there is a hole, enter route recovery}
21:     $n \Leftarrow n_j$ with $[H_j, I_j]$ where $|H_s - H_j| + |I_j - (k - 1) H_j - p\backslash$ is the minimum in $N_a$
22: end if
23: else {packet is already in the route recovery}
24:     $n \Leftarrow n_j$ with $[H_j, I_j]$ where $|H_s - H_j| + |I_j - (k - 1)H_j - p|$ is the minimum in $N_a$
25: end if
```

Algorithm 2-I explains route recovery when sending packets towards center. Line 7 of the algorithm calculates the next hop cell and line 8 checks if there is a neighbor in the next cell. If there is no neighbor in the next cell, the algorithm enters route recovery in line 10. To find an alternative path, in line 11, the sensor sending its packet (i.e., data or query) towards center checks its neighbors and chooses the neighbor having the smallest *H*, which shows the shortest cell-count of the node from the origin cell (see node *C* in Figure 6).

Algorithm 2-II explains route recovery when the data is being sent from the center to the sink. In line 15, *p*, the maximum number of hops between the cell of the sink and the first border line, is calculated. That is the number of hops between lines *s* and *b* (i.e., first border line of the hextant) in Figure 6. Line 16 checks if the current node is in the regular path of the packet to know if the packet is already in the route recovery or not. The node is in the regular path if $H <= H_s - p$ and $I = Hk - (H_s - p)$ or $H > H_s - p$ and $I = (H_s - H)k$, otherwise it is off the regular path. If the packet is in the regular path, in line 17, the next hop cell is calculated based on *Algorithm 1-II*. If there is no neighbor in the next hop cell, the packet enters route recovery at line 20. In line 21, the packet is forwarded to the neighbor n_j within cell $[H_j, I_j]$ where H_j is the closest to H_s and I_j is the closest to $p + (k - 1)H_j$ in neighbor list, N_a. This approach achieves to forward the data packet to the cell which is on the hexagonal ring that is the closest to the hexagonal ring of the sink. At the same time, it tries to keep the same distance from the second border line (i.e., line *r*) as sink. In Figure 6, where both the sink and the

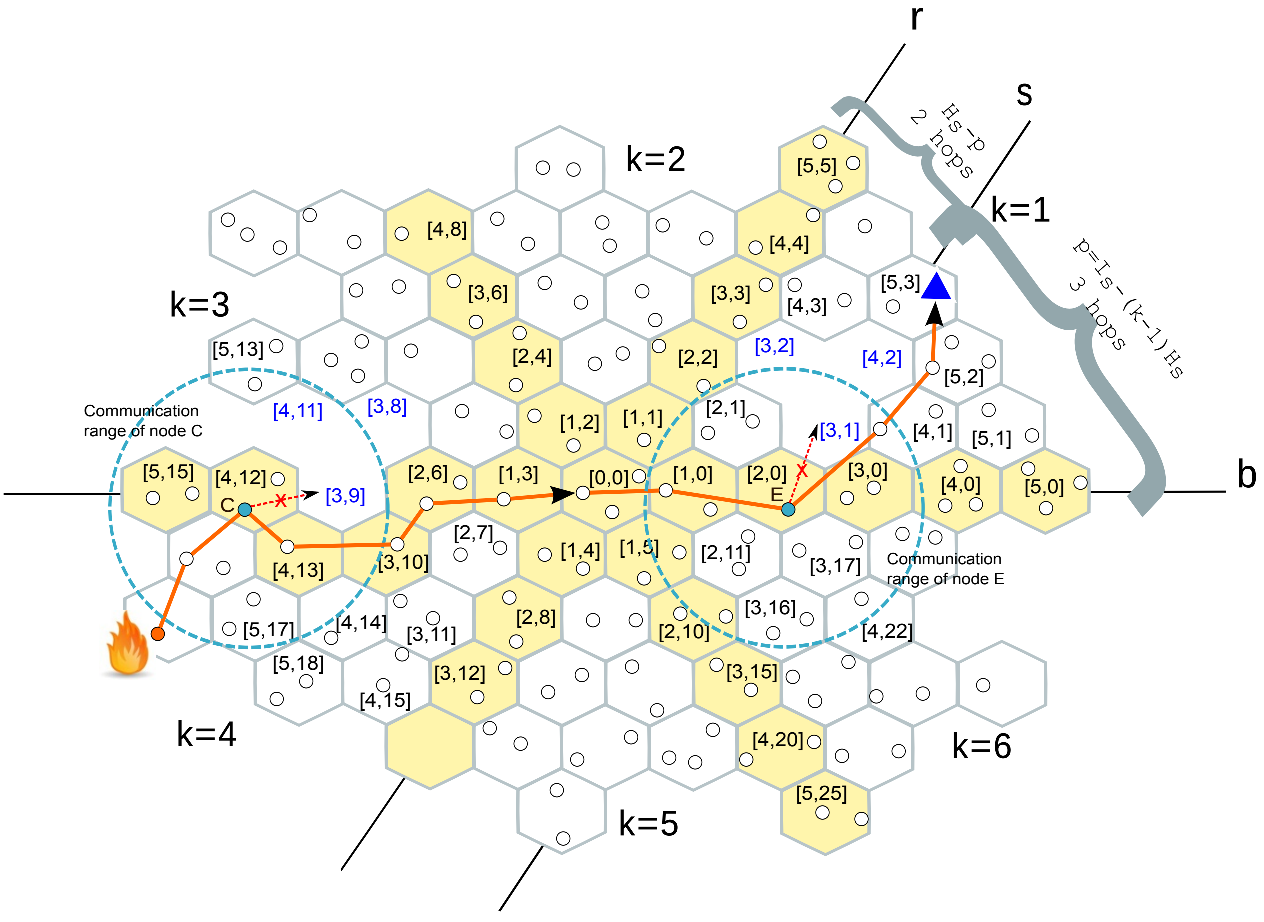


Figure 6 Fault tolerance mechanism in HexDD.

node *E* are located on the line *s*, node *E* in cell [2,0] forwards the packet to the cell [4,1] according to the rule in line 21. If the packet is already in the route recovery, it applies the same rule in line 24.

This mechanism is simple and efficient since it avoids to flood any other control message to inform other nodes about the hole, which is required to find new bridge nodes. This is mainly the advantage of using honeycomb tessellation and the chosen addressing scheme. It is important to point out that in HexDD, if a hole happens at the center of the network, the crossing area of the border lines at the central region should be shifted to a closer location which is not affected by the hole, or the first possible hexagonal ring which excludes the hole can become the central region.

Instead of calculating forwarding path between the center and a sink by *Algorithm 2-II*, the reverse path can be stored in the query. Since the sink sends a new query whenever it changes its cell, saving the path in the query is also efficient. The reverse path in the query recovers the hole at the path back to sink (i.e., assuming communication links are bidirectional) because when the query is being sent towards center, the alternative path is calculated and stored in the query. However, if a new hole is formed on the path back to the sink, the reverse path stored in the query packet will not be valid anymore.

4.4 Mobility support

The mobility of WSN, where most of the sensors are stationary, can be divided into the *source (stimulus) mobility* and *sink mobility*.

The impact of source mobility on the dissemination scheme is very small because when stimulus moves from one cell to another cell, a sensor that captures the stimulus becomes the source node and sends the data towards the center. In our study, the aim is collecting data about the event. For also tracking the event, each data should be augmented with the location of the source node. On the other hand, to support sink mobility, tracking the location of a sink becomes more critical for data delivery. If the sink moves inside its current cell, there is no need for another process since the data will be forwarded to the same neighboring cell until the sink leaves its cell. When the sink moves to another cell, it needs to send a new query message towards the center to inform the center nodes about its new cell. If any border node has the requested data in its cache (see node A in Figure 7), it directly sends data to the new cell of the sink.

Although it is assumed that sensor nodes are stationary in our study, HexDD can also handle mobility of sensor nodes. Sensors can easily recalculate their new hexagonal cells by using node-cell association algorithm [7] while they are moving. However, the uniform deployment of the sensor network should not be affected by the mobility of sensors. Thus, HexDD allows for a limited mobility of sensor nodes, meaning that the percentage of moving nodes should be low, so that the risk of having disconnected network partitions and too many holes in the network will be kept low.

4.5 Resizing rendezvous regions

In HexDD protocol, most of the traffic is concentrated on the center cell. If the number of events and sinks is high, congestion may happen around the center of the network and creates a hot spot region problem. A solution to hot spot region problem is to adjust the size of the central region (i.e., number of center cells and number of sensors in the center cells) according to the size of the network and the network traffic. The HexDD protocol can easily establish the size of the central region according to the number of sink-source pairs in the network. While there is only one sink-source pair in the network, one center cell can be enough to avoid congestion. On the other hand, for larger number of sink-source pairs, the central region consisting of center cell and the cells at the first and/or second hexagonal rings can achieve better performance. For this adaptive mechanism, HexDD simply checks the queue size of the nodes in the central region. If the queue size is above a certain threshold, one more hexagonal ring joins the central region to serve as rendezvous area. The effect of central region resizing on the performance of HexDD protocol is evaluated in Section 6.2.5.

5 Performance analysis

This section provides an analytical study of communication cost and hot spot traffic cost of HexDD and other protocols given in Section 2. The *communication cost* represent the total amount of messages generated in the network during the data dissemination and look up process. It is important to estimate communication cost since it has a direct influence on the network lifetime. The *hot spot traffic cost* is the total energy consumption of one single node located at hot regions. It is also important because it restricts the network scalability and lifetime.

5.1 Analysis model and assumptions

We consider a network with large number of nodes being deployed uniformly and distributed over a unit area. We use the function $H(l)$ as the number of hops on a path between two arbitrary nodes x and y such that $|x,y| = l$ is the euclidean distance between these two nodes. According to [31], given a geographical routing protocol, we have $H(l) = \zeta \frac{1}{r}$ where r is the communication range and $\zeta \geq 1$ is a scaling factor that depends

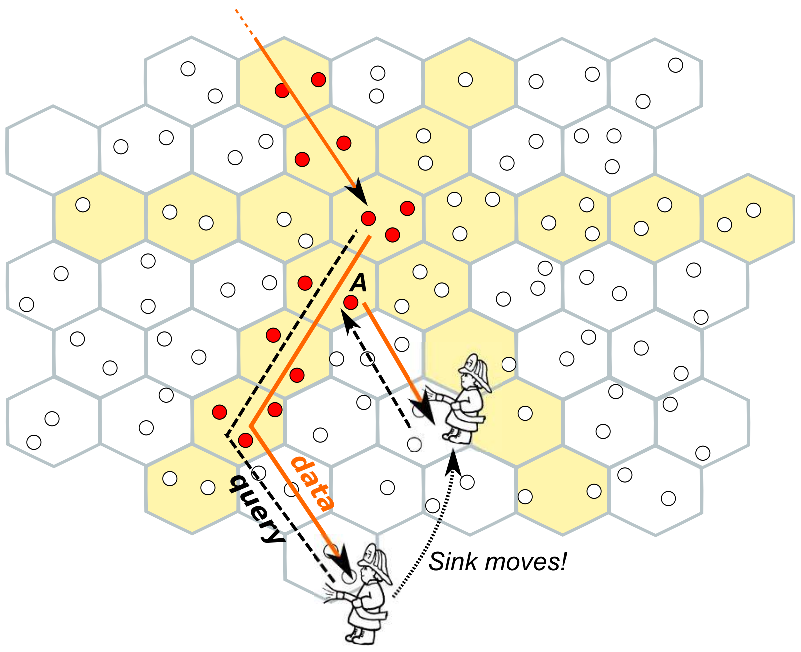


Figure 7 Mobility of sink in HexDD.

on the spatial node density λ. For simplicity in our analytical analysis, we assume that $\zeta = 1$.

For conformity with the analysis in [12], we consider four types of messages: event notification, query, data, and control messages, whose sizes are p_e, p_q, p_d, and p_c respectively. We consider m sinks moving randomly in the sensor field as well as n sources. Each sink generates a number of queries equal to $\bar{q}$ and each source generates a number of events equal to $\bar{e}$. Thus, the total number of queries and events can be written as $m\bar{q}$ and $n\bar{e}$.

5.2 Communication cost

The *total communication cost* is the sum of the communication cost brought by all control messages, event notification messages, queries, and data messages. In other words, it represents the total number of messages generated in the network during the data reporting, data lookup, and data collection processes. The total communication cost is the summation of three components:

(i) C_{DD}: cost of data reporting to the rendezvous region
(ii) C_{DL}: cost of data lookup (query dissemination) to the rendezvous region
(iii) C_{DT} : cost of transferring data from the rendezvous region to a sink

Therefore, the total communication cost of a given protocol is $C_{\text{protocol}} = C_{\text{DD}} + C_{\text{DL}} + C_{\text{DT}}$. We use the following metrics in the calculations: (i) $D_{\text{src,rdv}}$ - the distance between the source node and the rendezvous region, (ii) $D_{\text{sink,rdv}}$ - the distance between sink and the rendezvous region, (iii) $D_{\text{rdv,sink}}$ - the distance between the rendezvous region and sink. In what follows, we compare the HexDD, LBDD, TTDD, GHT, and RailRoad protocols. Figure 8 shows the worst case scenarios for each protocol, which is considered in the calculations.

HexDD

In case of HexDD, upon the detection of a new event, the sensor node sends the sensor reading towards one of the border lines and then to the central region. In the worst case, this message travels the path from source to RP, $D_{\text{src,rdv}} = \frac{1}{2} + \frac{\sqrt{3}}{6} \approx 0.79$, in Figure 8a and meets about $H(0.79)$ nodes. To retrieve data, a mobile sink sends a query message which is forwarded towards one of the border lines and then forwarded to the center. In worst case, the query travels the path from sink to RP, $D_{\text{sink,rdv}} = \frac{1}{2} + \frac{\sqrt{3}}{6} \approx 0.79$, in Figure 8a and meets about $H(0.79)$ nodes. After the query and data meet at the central region, the data packet is transferred from the RP to the sink, $D_{\text{rdv,sink}} = \frac{1}{2} + \frac{\sqrt{3}}{6} \approx 0.79$, and meets in the worst-case $H(0.79)$ nodes. Therefore, the total communication cost of HexDD is

$$C_{\text{HexDD}} = n\bar{e}p_d H(0.79) + m\bar{q}p_q H(0.79) + n\bar{e}p_d H(0.79).$$

LBDD

In the case of LBDD, upon the detection of a new event, the sensor node sends the measured data towards the

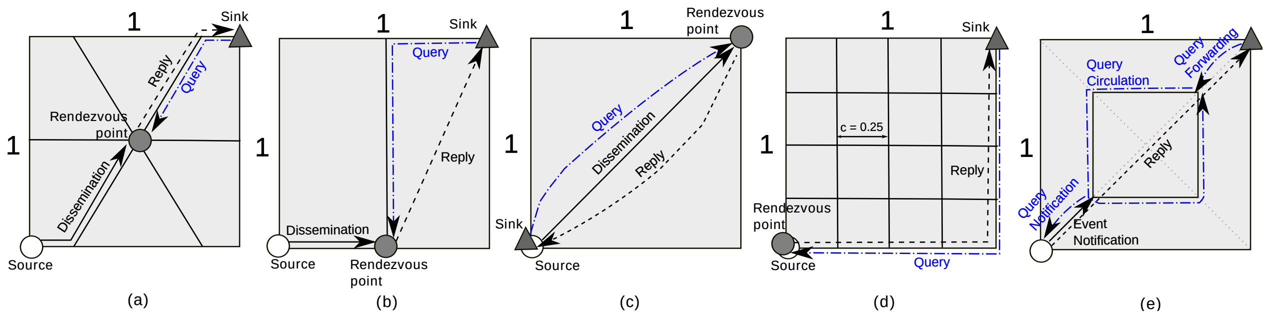


Figure 8 Worst case scenarios for (a) HexDD, (b) LBDD, (c) GHT, (d) TTDD, (e) RailRoad.

line. In the worst case, this message travels $D_{\text{src,rdv}} = 0.5$ in Figure 8b and meets about $H(0.5)$ nodes. To retrieve the data, a mobile sink sends a query message which is forwarded greedily towards the line. This message is then propagated along the line until it is received by the corresponding inline-node. In the worst case, the query travels $D_{\text{sink,rdv}} = 0.5{+}1$ and meets about $H(1.5)$ nodes. Then, the data is transferred from the inline-node to the sink, traveling $D_{\text{rdv,sink}} = \sqrt{5}/2 \approx 1.12$ and meets in the worst-case $H(1.12)$ nodes (diagonal of a half square). To avoid the transfer of duplicated data, it is supposed that a sink receives a response to its query only if the inline-node owns a new data. The total communication cost of LBDD in the worst case is then

$$C_{\text{LBDD}} = n\bar{e}p_d H(0.5) + m\bar{q}p_q H(1.5) + n\bar{e}p_d H(1.12).$$

GHT, TTDD, and RailRoad

The total communication cost of GHT, TTDD, and RailRoad are calculated in a similar way. As shown in Figure 8c, for GHT, $D_{\text{src,rdv}} = D_{\text{sink,rdv}} = D_{\text{rdv,sink}} = \sqrt{2} \approx 1.41$. For TTDD calculation (i.e., the equation of C_{TTDD}), each term indicates the communication costs of grid construction, query forwarding, and data forwarding, respectively. As shown in Figure 8e, in RailRoad, $D_{\text{src,rdv}} = D_{\text{sink,rdv}} = \sqrt{2}/4 \approx 0.35$, and the perimeter of the Rail is 2. Each term of C_{RailRoad} equation indicates the communication cost of event notification, query forwarding, query circulation, query notification, and data dissemination (for further details, refer to [12]).

$$C_{\text{GHT}} = n\bar{e}p_d H(1.41) + m\bar{q}p_q H(1.41) + n\bar{e}p_d H(1.41);$$

$$C_{\text{TTDD}} = n\frac{4\lambda}{H\left(\frac{1}{c}\right)}p_c + m\bar{q}[\lambda c^2 + H(2)]p_q + n\bar{e}[H(2) + H(\sqrt{2}/(2c))]p_d;$$

$$C_{\text{RailRoad}} = n\bar{e}p_e H(0.35) + m\bar{q}p_q H(0.35) + m\bar{q}p_q H(2) + n\bar{e}p_q H(0.35) + n\bar{e}p_q H(\sqrt{2}).$$

Figure 9 shows the comparison of the worst-case communication costs of all approaches for two scenarios. We consider 1000 sensor nodes deployed on a unit square sensor field of size 1 × 1. The sensor coverage area radius is $r = 0.1$ and we suppose that the size of a TTDD cell $c = 0.25$. The first scenario considers a fixed number of queries per sink ($\bar{q} = 50$) with a varying number of data reports per source node. The results for the first scenario are shown in Figure 9a. In the second scenario, we consider a fixed number of data reports per source ($\bar{e} = 50$) for a varying number of queries per sink, where the results are shown in Figure 9b.

It can be seen that TTDD presents a rather high communication cost in both scenarios resulting from its need to build grids and its routing strategy along the grid. RailRoad and LBDD, which implement a large virtual infrastructure, are more suitable for scenarios with a high number of data reports as shown in Figure 9a. The reason is that the infrastructure reduces the communication path; thus, it also reduces the cost between the source and the node having the disseminated data. On the other hand, the protocols GHT and LBDD are more suitable for scenarios with a large number of queries because these protocols propose a low look-up cost as shown in Figure 9b. Finally, HexDD, which combines a large infrastructure with a central re-dissemination strategy reducing the data look-up cost, presents a lower communication cost in both scenarios.

5.3 Hot region traffic cost

In rendezvous-based protocols it is important to estimate how densely messages are concentrated on the rendezvous area. *Hot region traffic cost* is the average energy spent by a hot spot region node. In data-centric storage such as GHT, all messages are directed to several home nodes. To prevent home nodes from being exhausted due to heavy traffic, replicas of home nodes are chosen. This approach, however, increases the total energy consumption and the replication cost of home nodes. In Railroad, every query and event summary is

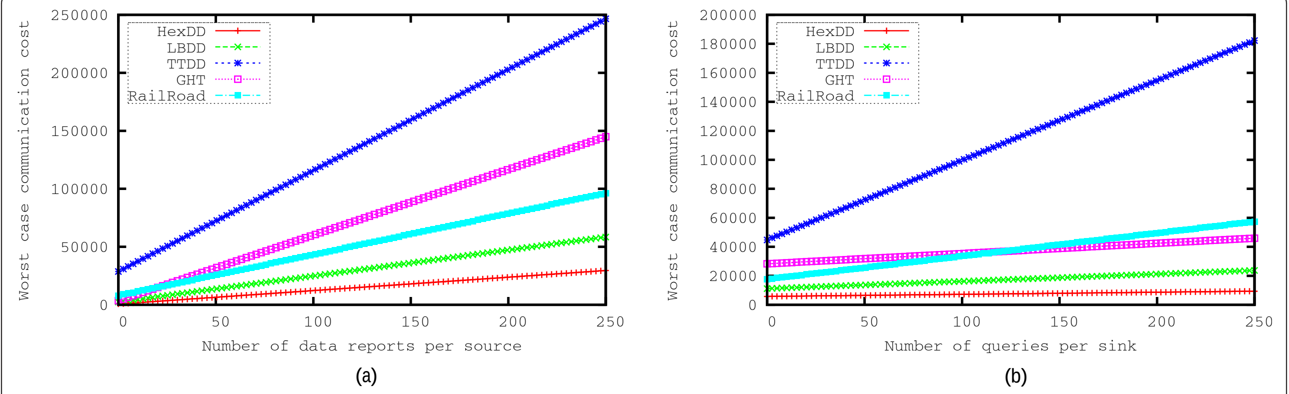


Figure 9 Worst case communication costs ***(m = 5 sinks, n = 10 sources)***: (**a**) number of queries $\bar{q} = 50$, (**b**) number of events $\bar{e} = 50$.

sent to the Rail, which can be the bottleneck that limits the network lifetime. Also, in HexDD queries and data packets are forwarded toward border lines, which are becoming hot regions in the network. In this section, we analyze the hot region traffic cost of GHT, RailRoad, LBDD and our approach HexDD. In the following calculations, T is the amount of energy for a node to transmit a single bit, and R is the energy needed to receive a bit.

In data-centric storage, the home nodes can be hot spots, and the hot spot traffic cost can be written as follows [30]:

$$EH_{\mathrm{GHT}} = R\left[\frac{n\bar{e}}{s}p_d + \frac{1}{\gamma}\frac{m\bar{q}}{s}p_q\right] + T\left[\frac{1}{\gamma}\frac{n\bar{e}}{s}p_d\right]$$

where γ is the number of nodes in a replica set including the home node. It means a home node has γ - 1 replicas. When γ is set to 1, there exists no replica nodes but the home nodes. There are s different event types in the network. We assume that there is only one event type (s = 1) in the network for the calculations. All data and query packets coming from sources and sinks are received by the home node, i.e., $R\left(n\bar{e}p_d + \frac{1}{\gamma}m\bar{q}p_q\right)$. The home node, then, transmits the data packet to the sinks, i.e., $T\left(\frac{1}{\gamma}n\bar{e}p_d\right)$.

The hot spot traffic cost of Railroad can be written as follows [30]:

$$EH_{\mathrm{RR}} = \frac{1}{N_R}\left[\begin{array}{l} Rn\bar{e}p_eN_{ST} + Rm\bar{q}p_qN_{RT} + \\ Tn\bar{e}p_e + Tn\bar{e}p_q + Tm\bar{q}p_qN_{RT} \end{array}\right]$$

where N_R, N_{ST}, and N_{RT} stand for the number of the rail nodes, the number of rail nodes in a station[b], and the number of nodes that a query stays in a single tour around Rail, respectively. In an event notification process, one node out of N_{ST} nodes transmits the event notification packet (i.e., $Tn\bar{e}p_e$) sent by a source node and N_{ST} nodes receive this event notification packet (i.e., $Rn\bar{e}p_eN_{ST}$). For query flooding in the Rail, N_{RT} nodes out of N_R nodes receive a query packet (i.e., $Rm\bar{q}p_qN_{RT}$) sent by a sink and N_{RT} nodes out of N_R nodes transmit the query packet (i.e., $Tm\bar{q}p_qN_{RT}$). Finally, one node out of N_{RT} nodes transmits the query packet to the source node (i.e., $Tn\bar{e}p_q$). The data is directly sent from source to sink with GF.

The hot spot traffic cost of LBDD can be written as follows:

$$EH_{\mathrm{LBDD}} = \frac{1}{N_L}\left[\begin{array}{l} Rn\bar{e}p_dN_{ST} + Rm\bar{q}p_qN_L + \\ Tn\bar{e}p_d + Tm\bar{q}p_qN_L \end{array}\right]$$

where N_L is number of inline nodes and N_{ST} is the number of inline nodes in a station which is a small group of nodes in the virtual line. In the data dissemination process, N_{ST} nodes out of N_L nodes receive a data packet (i.e., $Rn\bar{e}p_dN_{ST}$) sent by a source node. For query flooding in the line/strip, N_L nodes receive the query packet (i.e., $Rm\bar{q}p_qN_L$) sent by a sink and N_L nodes transmit the query packet (i.e., $Tm\bar{q}p_qN_L$). Finally, one node out of N_{ST} nodes sends the data packet to the sink (i.e., $Tn\bar{e}p_d$). GF is used to send data to the sink.

The hot spot traffic cost of HexDD is as follows,

$$EH_{\mathrm{HexDD}} = \frac{1}{3N_{BL}}\left[\begin{array}{l} 2Rn\bar{e}p_d\frac{N_{BL}}{2N_C} + Rm\bar{q}p_q\frac{N_{BL}}{2N_C} + \\ 2Tn\bar{e}p_d\frac{N_{BL}}{2N_C} + Tm\bar{q}p_q\frac{N_{BL}}{2N_C} \end{array}\right]$$

where N_{BL} is the number of border nodes in a diagonal line, and N_C is the average number of nodes in a cell. $N_{BL}/2N_C$ is the number of cells on a border line[c]. Since one node per cell transmit or receive the packets, $N_{BL}/2N_C$ is also the number of nodes having the packets on a border line. In data dissemination and data transfer process, a node in each cell receives and transmits the data packet along the diagonal line (i.e.,

$\bar{q} = 50$). The sink's query travels the border line (i.e., $Rm\bar{q}p_q\frac{N_{BL}}{2N_C} + Tm\bar{q}p_q\frac{N_{BL}}{2N_C}$). The above formula for the hot spot traffic cost of HexDD can be written as:

$$EH_{\text{HexDD}} = \frac{1}{3N_C}\begin{bmatrix} Rn\bar{e}p_d + Rm\bar{q}p_q(0.5)+ \\ Tn\bar{e}p_d + Tm\bar{q}p_q(0.5) \end{bmatrix}$$

For calculation of the hot spot traffic costs of the protocols, the number of sources n and number of sinks m vary between 1 to 18 in the first set of analysis to see the effect of number of sinks and sources on the hot spot regions. In the second set of analysis, we set n to 5 and m to 15 and the number of queries per sink and the number of events per source are varied to see the effect of the network traffic generated by sinks and sources on the hot spot regions. Total number of nodes N in a sensor field of 1000 m × 1000 m is set to 10000. The number of rail nodes N_R is 8% of total nodes and the number of nodes that receive a query in the Rail N_{RT} is 480. In the analysis, we use the same values used in [30] for the number of rail nodes in a station N_{ST}, and R/T which are taken as 16 and 3/8, respectively. Both the width of the Rail and the station are set to 40 m, that is the radio range of the sensor nodes. Based on the values given above, the average number of nodes in a cell, N_C, is 3 in HexDD. We set $p_e = p_c = p_q$ and $p_d = 2 \times p_q$.

In Figures 10, 11, and 12 we show the hotspot traffic cost of HexDD compared with other protocols. In the first graphs of Figures 10a, 11a, and 12a, x axis is the number of sinks (m) and y axis is the number of sources (n). In the second graphs of Figures 10b, 11b, and 12b, x axis is the total number of queries ($m\bar{q}$) and y axis is the total number of events ($n\bar{e}$). The z axis of the graphs shows the ratio between the hot spot traffic cost of HexDD (EH_{HexDD}) and the hot spot traffic cost of another protocol (EH_{protocol}). A border node in HexDD processes less data than a rendezvous node in the other protocol if the ratio $EH_{\text{HexDD}}/EH_{\text{protocol}} < 1.0$. In the first set of graphs, the aim is to see the effect of varying number of sinks and sources on the hot spot traffic costs of the protocols. The second set of graphs shows the hot spot traffic costs in the *event-driven* scenario, where the number of event messages per source ($\bar{e}$) is larger than the number of queries per sink ($\bar{q}$), and in the *query-based* scenario, where the number of queries per sink is larger than the number of event messages per source.

Figure 10a shows the hot region traffic cost of HexDD compared with the data-centric storage GHT with varying number of sinks and sources. The result shows that a home node in a data-centric storage has to process much more requests than a border node in HexDD protocol since $EH_{\text{HexDD}}/EH_{\text{GHT}} < 1.0$ for all the given values of number of sinks and number of sources. This is more remarkable as the number of sources increases and the number of sinks decreases. In Figure 10b, where we vary the number of queries per sink and the number of data reports per source, the same behavior is observed as the total number of events increases and the total number of queries decreases. Both graphs show that the hot spot traffic cost of HexDD is much less than that of a data-centric storage.

In Figure 11 we compare the hot region traffic costs of HexDD and RailRoad. The results in Figure 11a show that when we have many event sources but a couple of sinks in the network (i.e., see n = 15, m = 3, and $EH_{\text{HexDD}}/EH_{\text{RR}}$ = 1.8 in the figure), a border node in HexDD processes much more requests than a rail node in RailRoad. This is due to the fact that RailRoad does not process/forward data reports in the Rail region; on the other hand, in HexDD diagonal lines are also used for data forwarding to cache data on the border nodes for sink queries. This is an expected results because HexDD is designed for networks where the difference between the number of sinks and sources is not very high. For instance, when n = 15 and m = 6, the ratio $EH_{\text{HexDD}}/EH_{\text{RR}}$ = 0.98 so HexDD is still better than RailRoad. As observed in the figure, when the number of sinks is greater than or equal to the number of sources, the hot spot traffic cost of HexDD is much less than that of RailRoad. Figure 11b presents the results of a scenario having 15 sinks and 5 sources in the network. Apparently, HexDD becomes advantageous over RailRoad in terms of hot spot traffic cost in the *query-driven* scenarios, where the query generation rate is higher than the event generation rate. Also, when the total number of queries is close to the total number of events, HexDD still processes less requests on the rendezvous lines than RailRoad.

Figure 12 compares the hot region traffic costs of HexDD and LBDD. It has a similar behavior with the previous graphs for RailRoad comparison because the ratio $EH_{RR}/EH_{LBDD} \simeq 0.6$ for the given network specifications. This means that an inline node of LBDD already processes more requests in the line-based rendezvous region than a rail node in RailRoad. Also, as shown in Figure 12a,b, an inline node of LBDD processes much more requests than a border node of HexDD in most of the cases. The same observations previously discussed for RailRoad comparison are also valid for LBDD comparison. Only the ratio $EH_{\text{HexDD}}/EH_{\text{LBDD}}$ is smaller than the ratio $EH_{\text{HexDD}}/EH_{\text{RR}}$ for the same inputs. For instance, when n = 15 and m = 6, the ratio $EH_{\text{HexDD}}/EH_{\text{LBDD}}$ = 0.59.

In this section, we analyzed the influence of the rendezvous region placement on the number of packets (i.

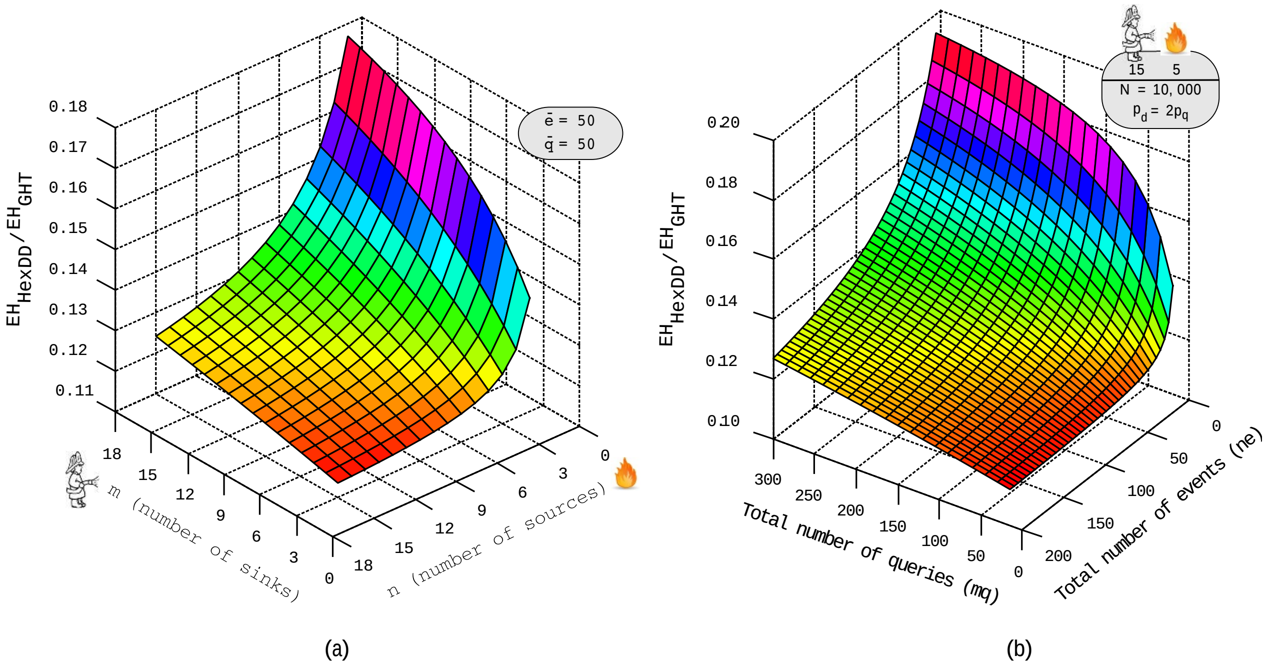


Figure 10 Analysis of hot spot traffic cost of HexDD compared with GHT: (**a**) number of sinks vs. number of sources, (**b**) total number of queries vs. total number of events.

e., query and data) processed (i.e., received/transmitted) in each protocol during the data dissemination. We also analyzed the effect of network traffic generated by sinks and sources on the load created in rendezvous regions of different protocols. In these analytical analyses, networking issues, such as packet drops, retransmissions, congestion near hot spot regions, were not taken into account. In the following section, we evaluate the

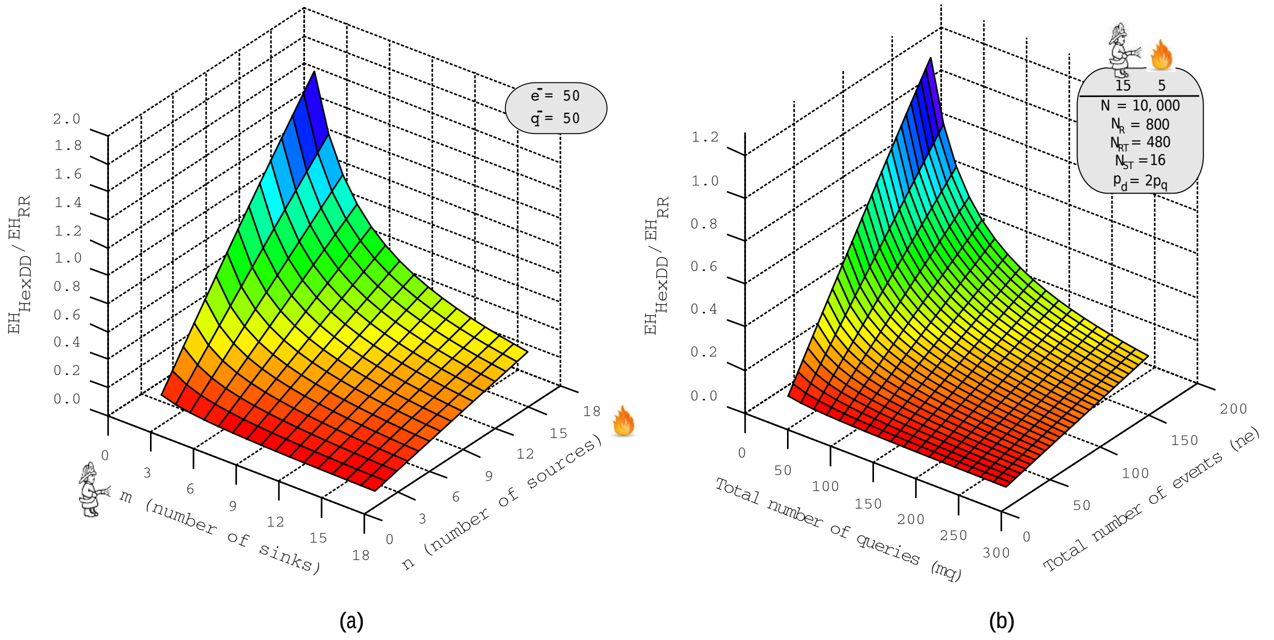


Figure 11 Analysis of hot spot traffic cost of HexDD compared with RailRoad: (**a**) number of sinks vs. number of sources, (**b**) total number of queries vs. total number of events.

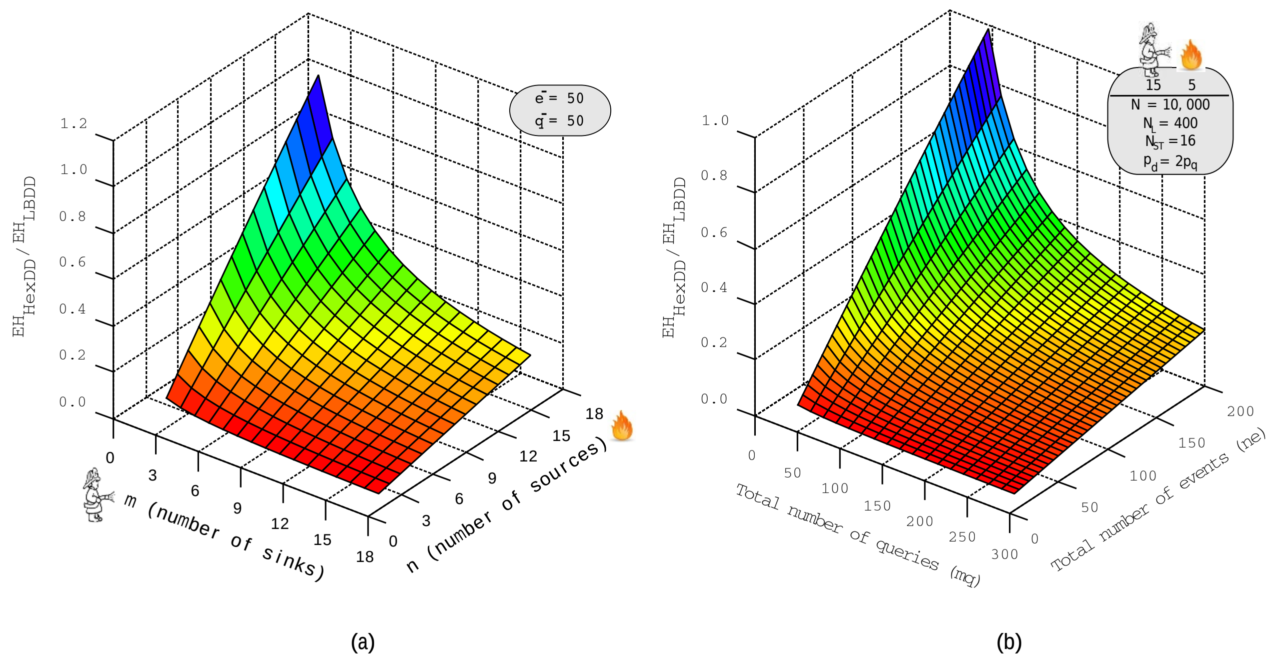


Figure 12 Analysis of hot spot traffic cost of HexDD compared with LBDD: (**a**) number of sinks vs. number of sources, (**b**) total number of queries vs. total number of events.

performance of the protocols by simulating realistic scenarios and networking conditions.

6 Performance evaluation

For the purpose of performance evaluation, we compare the proposed protocol, HexDD, with two other rendezvous-based approaches, LBDD, and TTDD. We choose TTDD and LBDD for the comparison since we would like to investigate the effect of using hexagonal tessellation instead of rectangular grids and using three diagonal lines acting as rendezvous area instead of only one line-based region. The simulations were carried out to evaluate routing performance and the fault-tolerance performance of the protocols. We also investigate the effect of central region size on HexDD protocol.

For this purpose, first, we analyze the protocols with varying number of sink-source pairs. Second, we explore the impact of sink mobility (i.e., sink's maximum speed) on the performance of these protocols. We then analyze the fault-tolerance performance and vary the total number of holes and the size of holes in the network. We also analyze the protocols' energy distribution maps which are important to see hot spot regions created by each protocol in the network. Indeed, since all data reports and queries are concentrated over the central region in HexDD protocol, the hot spot problem can arise, limiting the network lifetime and the scalability. To prevent the central cell from being a bottleneck, it is possible to design a larger central region, including the first hexagonal ring, to better distribute the load among its nodes. Therefore, we, finally, test the routing performance of HexDD with different size of central region (i.e., center with one cell or center with first hexagonal ring) to see if we can reduce energy consumption per node at the center with a larger central rendezvous region.

6.1 Simulation environment

HexDD protocol is implemented and tested in NS2 [32]. To guarantee a fair comparison between TTDD and HexDD, we set simulation parameters comparable to those used in [15]. This includes simulation of IEEE 802.11 DCF as the underlying MAC and an energy model in which a sensor's transmitting, receiving and idling power consumptions are set to 0.66W, 0.395W, and 0.035W, respectively. Although IEEE 802.15.4 is a standard developed to meet the needs for low-power and low-cost wireless communication, we prefer to use IEEE 802.11 standard in our simulations to be comparable with previous studies. A comparative performance study of IEEE 802.15.4 and IEEE 802.11 can be found in [33]. The cell size in TTDD is set to 600 m. In LBDD, width of the virtual line is set to 250 m. Each node has a transmission range of 250 m. For routing performance simulations, 250 sensor nodes are randomly distributed on a 2000×2000 m^2 field. For fault-tolerance performance simulations, 210 nodes are randomly distributed on a 1500×1500 m^2 field. Each simulation run lasts for

200 s. Results are averaged over six random network topologies. A source generates one data packet per second, so there are in total 200 data packets/source sent. Sinks' mobility follows the standard *linear mobility* model. Mobile sinks could attain a maximum speed up to 14 m/s with 5 s pause time. The stimuli remain static during the simulation time. For different sets of simulations, speed, and pause times of sink are varied.

We use the following metrics to evaluate the performance of the protocols: (i) *Data delivery ratio:* defined as the ratio between the total number of data packets received by the sinks and the total number of event data generated by the sources; (ii) *Data delivery delay:* defined as the total time elapsed between the data generation by a source and its reception by a sink, also averaged over all sink-source pairs; and (iii) *Energy consumption:* defined as the communication (transmitting and receiving) energy the network consumes; the idle energy is not counted since it depends largely on the data generation interval and does not indicate the efficiency of data delivery.

6.2 Simulation results

For each metric, we have evaluated the performance in four different scenarios:

- *Impact of the number of sink-source pairs:* In this scenario, the number of sink-source pairs is varied. Mobile sinks could attain a maximum speed up to 10 m/s with 5 s pause time.
- *Impact of sink mobility:* We test the performance of the protocols under both low mobility (i.e., 4-5 km/h for walking humans) and high mobility (e.g., 50-60 km/h for UAVs) scenarios. The sinks' speeds are set to 0, 5, 10, 15, and 20 m/s (0 to 72 km/h) with a pause time of 5 s, where the speed of 0 m/s allows for static sinks in the network. The speed 20 m/s means that a sink crosses the border of a cell approximately every 7 s (i.e., $2r = 138.5$ m which is the longest distance in a hexagon, $138.5/20 \cong 7$ s) in HexDD. There are 6 sink-source pairs in this scenario.
- *Impact of the number of holes in the network:* To see the performance of the fault tolerance mechanism, we create holes in the network for this scenarios. Each different shaped and middle sized hole covering 5 cells in the network is randomly generated and positioned on a hextant or over the border lines. The number of sources is equal to the number of holes in the network. Each source is placed on a location where it is affected by at least one hole. Three destination sinks are chosen randomly in the network.
- *Impact of the size of holes in the network:* In this scenario, we test the performance of the fault tolerance mechanism in terms of size of holes. In this set of simulations, there are one sink-source pair and one hole in the network. The size and the shape of the hole is changed for different runs. The size of the hole is represented as the number of cells that the hole covers.

6.2.1 Data delivery ratio

In the first three scenarios showing impact of sink-source pairs, sink mobility, and number of holes, we observe that HexDD outperforms others, as shown in Figure 13a-c. In these scenarios data delivery ratio for LBDD and TTDD is in the range 86%-98%, whereas for HexDD the range is 94%-100%. The use of six border lines in HexDD allows to better distribute the load among the nodes inside the rendezvous area. This assures high delivery ratio even for large number of sink-source pairs (see Figure 13a). Even though there is no explicit mobility tracking scheme in HexDD, it functions well under high mobility (see Figure 13b). The hole recovery mechanism in HexDD achieves a more reliable packet delivery than LBDD and TTDD (see Figure 13c).

Figure 13d shows the data delivery ratio vs. increasing hole size. The hole recovery mechanism in HexDD works well also for varying size of holes, as a result the data delivery ratio is high. TTDD performs better in such a scenario. Its grid structure allows the data packets to find other paths to bypass a hole even when we have a large hole in the network, thus the success ratio of TTDD is the highest. LBDD has the lowest performance because for larger holes the GF has more difficulties to find a path along the boundary of the hole and most of the time fails to route packets across the big holes.

6.2.2 Data delivery delay

As shown in Figure 14a-d, in all scenarios TTDD has the highest delay. In Figure 14a, the increase in the delay of TTDD is due to the time spent for the creation and the propagation of the grid for each source. The reason of the high delay of TTDD in Figure 14b is that sink mobility causes the reconstruction of a new path between the sink and the dissemination point on the grid every time the sink changes its local cell. TTDD has the highest delay in Figure 14c,d since the alternative paths along the grids to bypass holes are in most of the cases longer than the possible shortest path between a source and a sink. The delay of LBDD and HexDD are close to each other in all scenarios. The delay of HexDD is smaller than LBDD in scenarios showing the impacts of sink-source pairs, and sink mobility in Figure 14a,b, respectively. This is because a sink can access data

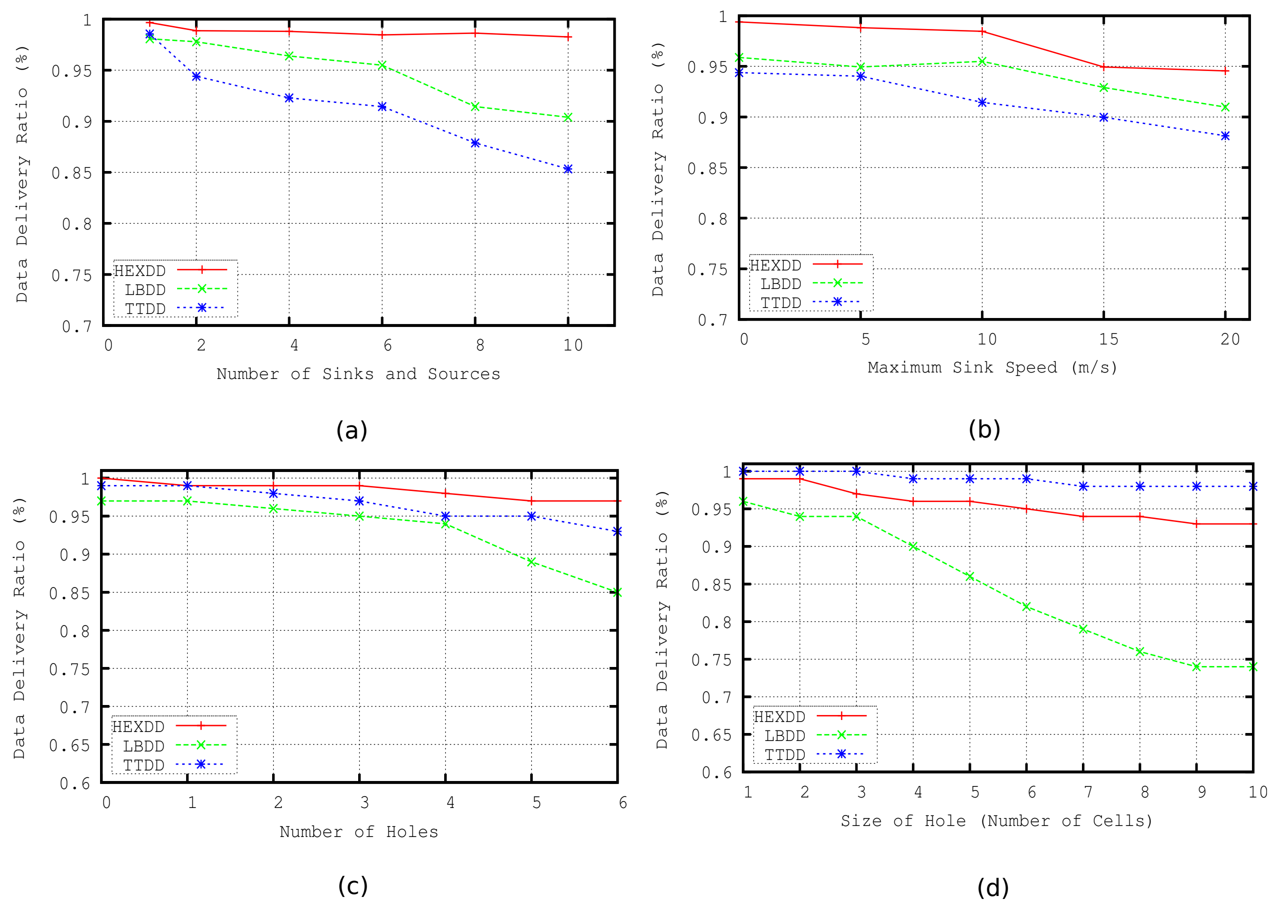


Figure 13 Data delivery ratio of the protocols against varying (a) number of sink-source pairs, (b) sink speed, (c) number of holes, and (d) size of holes.

stored in the border lines in a short time in HexDD even while it is moving around the network. In routing hole scenarios, the delay of LBDD is lower that HexDD as shown in Figure 14c,d. The reason of this is that in LBDD more packets get stuck in the holes and the lost packets are not included in the delay results. HexDD tries to forward packets along the boundary of a hole via the shortest possible path so it also has a low delay.

6.2.3 Energy consumption

In all scenarios shown in Figure 15, TTDD presents a rather higher energy consumption than HexDD and LBDD. In varying sink-source pairs and sink mobility scenarios (see Figure 15a,b, respectively), the energy consumption of TTDD is the highest since there is no global virtual infrastructure in TTDD, instead for each source node a separate grid structure is constructed. In routing hole scenarios (see Figure 15c,d), the energy consumption of TTDD is the highest due to long data forwarding paths on the grid between sources and sinks to evade holes. Thus, in TTDD there is a trade-off between the energy spent for recovering holes and the data delivery ratio. TTDD achieves a high data delivery ratio in hole recovery scenarios (see Figure 13d), but it is very costly.

Energy consumption of HexDD is the lowest in all scenarios as shown in Figure 15. The global virtual hexagonal infrastructure of HexDD results in lower energy consumptions since data packets are required to send only towards the rendezvous nodes (i.e., nodes in border lines). In routing hole scenario shown in Figure 15c, the energy consumption of HexDD first increases until a certain point (see between 0 and 3 holes in the figure) since it achieves to recover holes and find paths between sources and sinks. After that point, the energy consumption decreases slightly because the data delivery ratio of HexDD also decreases slightly (see Figure 13c). Less data forwarding results in less energy consumption. Energy consumption of HexDD in Figure 15d has also a similar behavior. The energy consumption of LBDD is higher than HexDD, but it stays more or less same when we increase the number or size of holes in the network (see also Figure 13c,d, respectively). This is

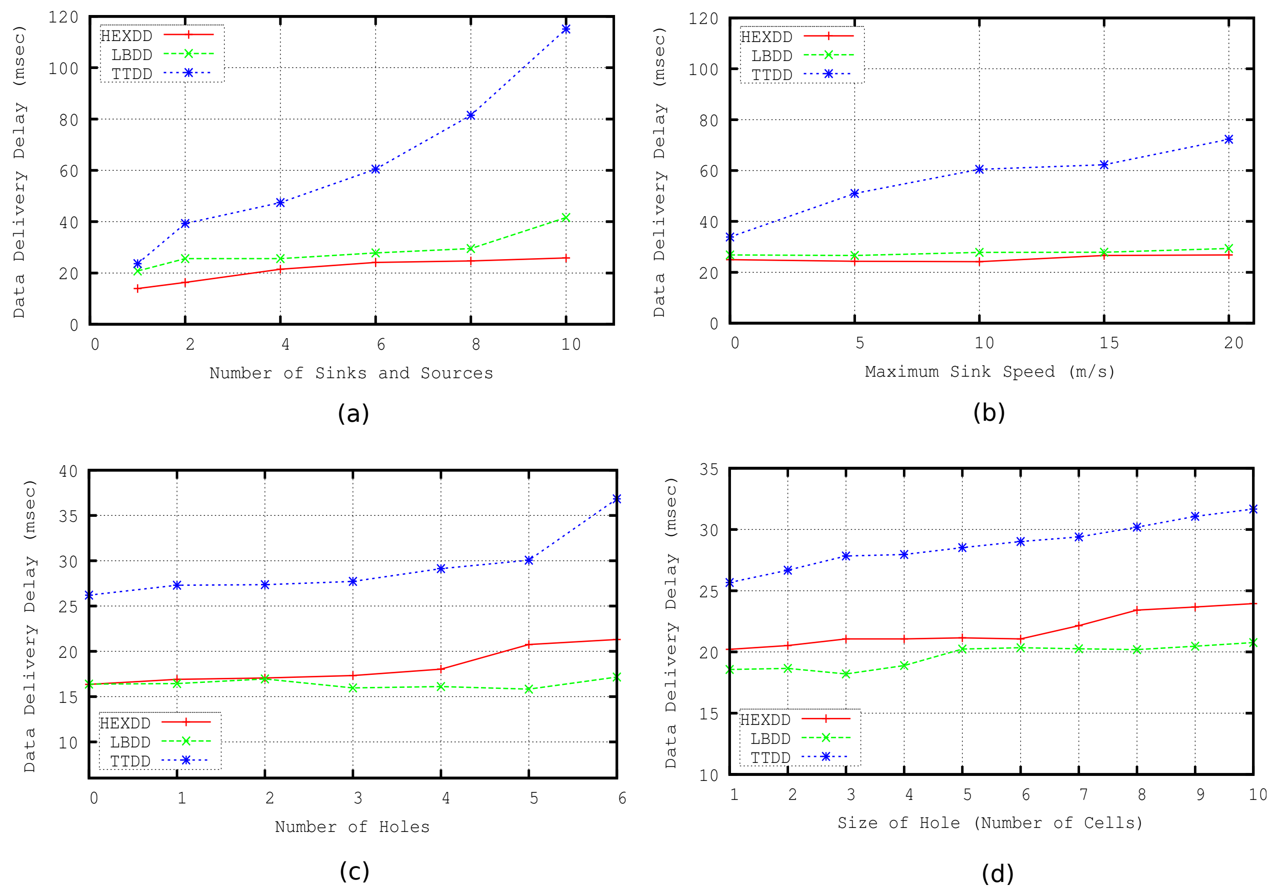


Figure 14 Data delivery delay of the protocols against varying (a) number of sink-source pairs, (b) sink speed, (c) number of holes, and (d) size of holes.

because the LBDD protocol always tries to find a path on the border of a hole and it forwards data packet until it gets stuck at some point. Therefore, LBDD still consumes energy on the same level although it has a lower data delivery ratio (see Figure 13c,d) when we increase the number or size of holes in the network.

6.2.4 Hotspot regions

The use of a virtual infrastructure for the data dissemination can lead to the hot spot problem. Indeed, as all data reports and queries are concentrated over the rendezvous area, the hot spot problem can arise, limiting thus the network lifetime and the scalability. In this set of simulations, we analyze hot spot region in a network having six sources and one sink, which is located at different locations for three different scenarios. The simulation run lasts for 600 s.

Figure 16 shows the distribution map of energy consumption for the protocols. Energy consumptions in hello packet transmissions for neighbor discovery and idle mode are not shown in the maps since it depends largely on the data generation interval and does not indicate the efficiency of the protocol. Although energy consumption is highly variable and depends on the current location of the sink and source, an important observation about our approach is that nodes in the border lines experience a higher energy consumption, which shows that energy consumption is distributed among the nodes in the rendezvous region. On the other hand, the nodes close to the center of the network consume the highest energy, as expected. The maximum energy expenditure of a center node is around 1200W (see Table 3). It is, therefore, observed that in HexDD the network lifetime is defined by a few nodes that are at the center of the network (see Figure 16c). Concerning LBDD, we notice that energy consumption is also distributed among the nodes in the rendezvous region, which is the central strip in the network. However, LBDD also has the highest energy consumption (around 900W) at the center of the network (see Figure 16b). The sensor node, which consumes the highest energy in the network, spends around 1000W in TTDD. Since a separate grid structure is constructed by each individual source in TTDD, the energy consumption is equally high throughout the network (see Figure 16a). This

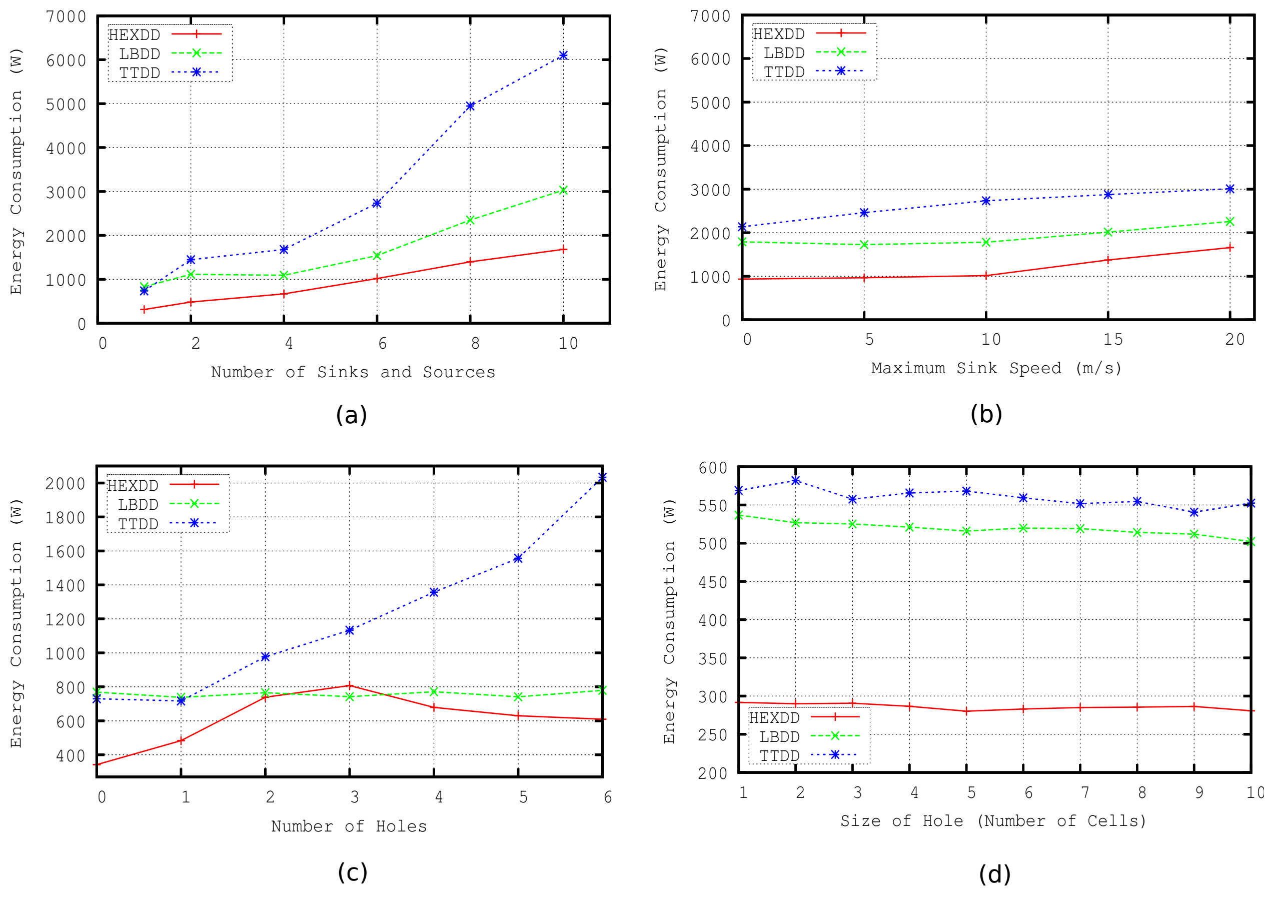


Figure 15 Energy consumption of the protocols against varying (a) number of sink-source pairs, (b) sink speed, (c) number of holes, and (d) size of holes.

increases the probability to exhaust the battery energy of the majority of nodes, leading to network partitioning and reduced network lifetime.

The overall energy consumptions of the protocols are shown in Table 3. The energy consumption of TTDD is higher than HexDD and LBDD, since as the sink moves, it tends to reconstruct a new path to a dissemination node on the grid by local query flooding and agent updates. Also, LBDD floods the query of the sink in the inline region for its location updates. HexDD has the smallest overall energy consumption. These simulation results about energy consumptions of the protocols agree with our theoretical analysis on communication costs of the protocols in Section 5.2. Although HexDD achieves a lower overall energy consumption, the nodes in the central cells in HexDD are more heavily loaded then other protocols. In the following section, we investigate the effect of rendezvous region resizing on the central nodes and on the overall network performance.

6.2.5 Impact of central rendezvous region size adjustment

In this section, we investigate the effect of central region size adjustment on the energy distribution in the network and network lifetime. We change the size of the central rendezvous region. Instead of having one cell at the center, we have extended the center with the first hexagonal ring as shown in Figure 17a. Yellow cells are (roughly) the rendezvous regions in the virtual infrastructure, and the pink cells are the central cells.

Energy map of central region with one hexagonal ring
Figure 16d shows the energy distribution of HexDD with one ring approach. In this approach, the load of the central cell in HexDD is distributed among all the nodes in the first hexagonal ring as shown in the figure. The energy consumptions of the nodes in the first hexagonal ring cells (see Figure 16d and Table 3), which is around 600W, is less than the energy consumptions of the nodes in the central cell (see Figure 16c and Table 3), which is around 1200W. As shown in Table 3, HexDD with one central cell approach consumes an overall energy of 1611W; on the other hand, the overall energy consumption of HexDD with one hexagonal ring approach, which is 1728W, is slightly higher than that of HexDD with one central cell, but the hot spot problem is lessened.

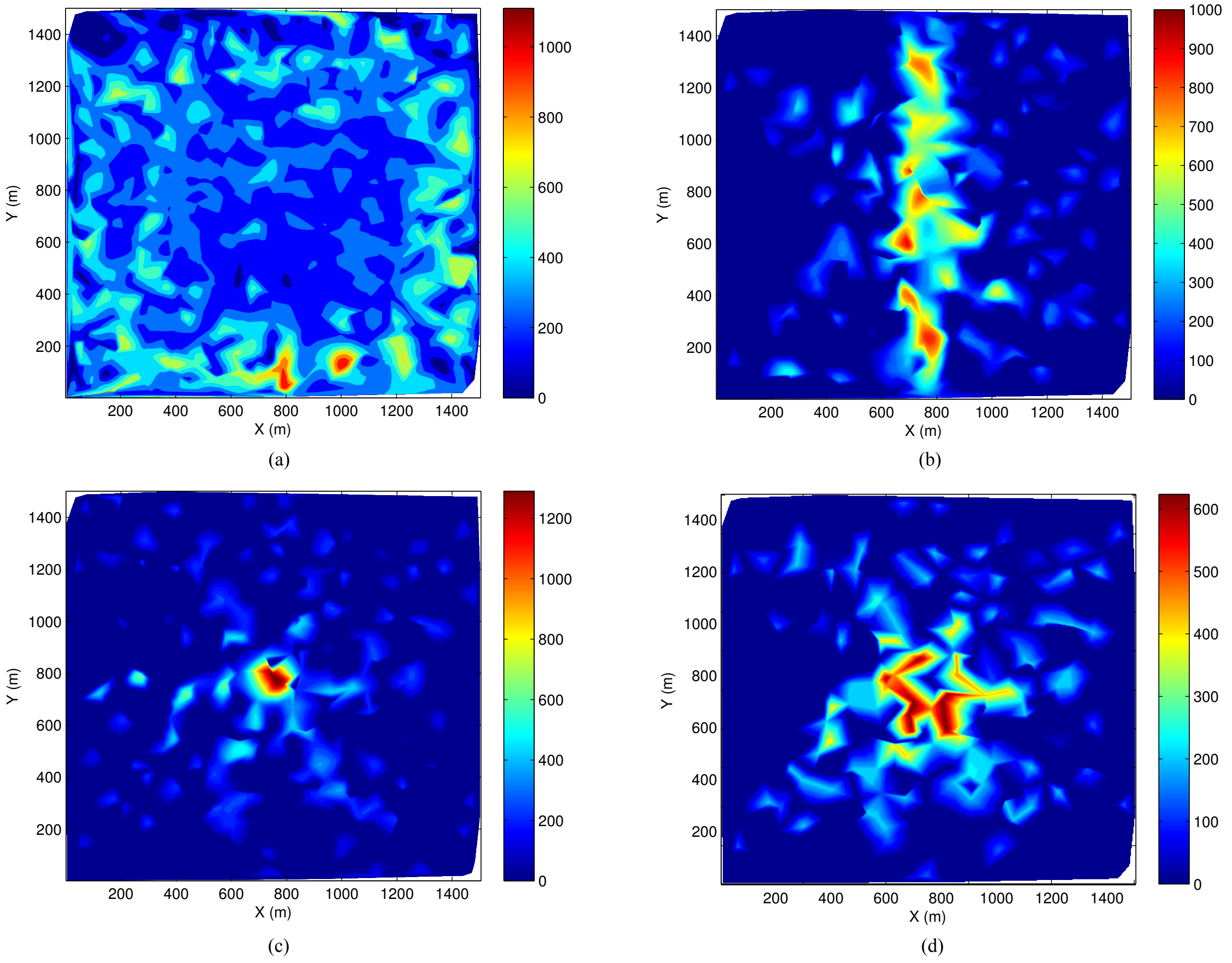


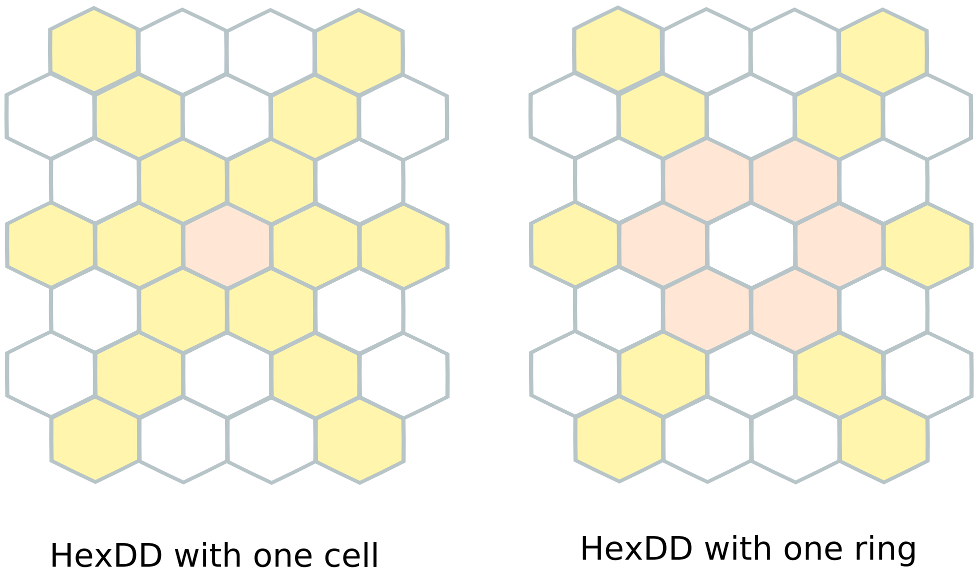
Figure 16 Impact of rendezvous-based data dissemination protocols on the energy consumption distribution: energy map of (**a**) TTDD, (**b**) LBDD, (**c**) HexDD, and (**d**) HexDD with one ring.

Network lifetime Several definitions of network lifetime can be found in literature. In this study, we define the network lifetime from an application point of view as *the time the application stops being operational, which in our case is the time corresponding to the last report received by a sink* [17]. In other words, when a sink is no longer able to receive reports from the sensors, the sink is said to be disconnected from the sensors, and the network is not functional anymore. To analyze the network lifetime, in Figure 17b we plot the *average application success ratio*. This ratio is defined as the ratio between the total number of data reports received by the sinks and the total number of reports generated by the sources since the start of the simulation.

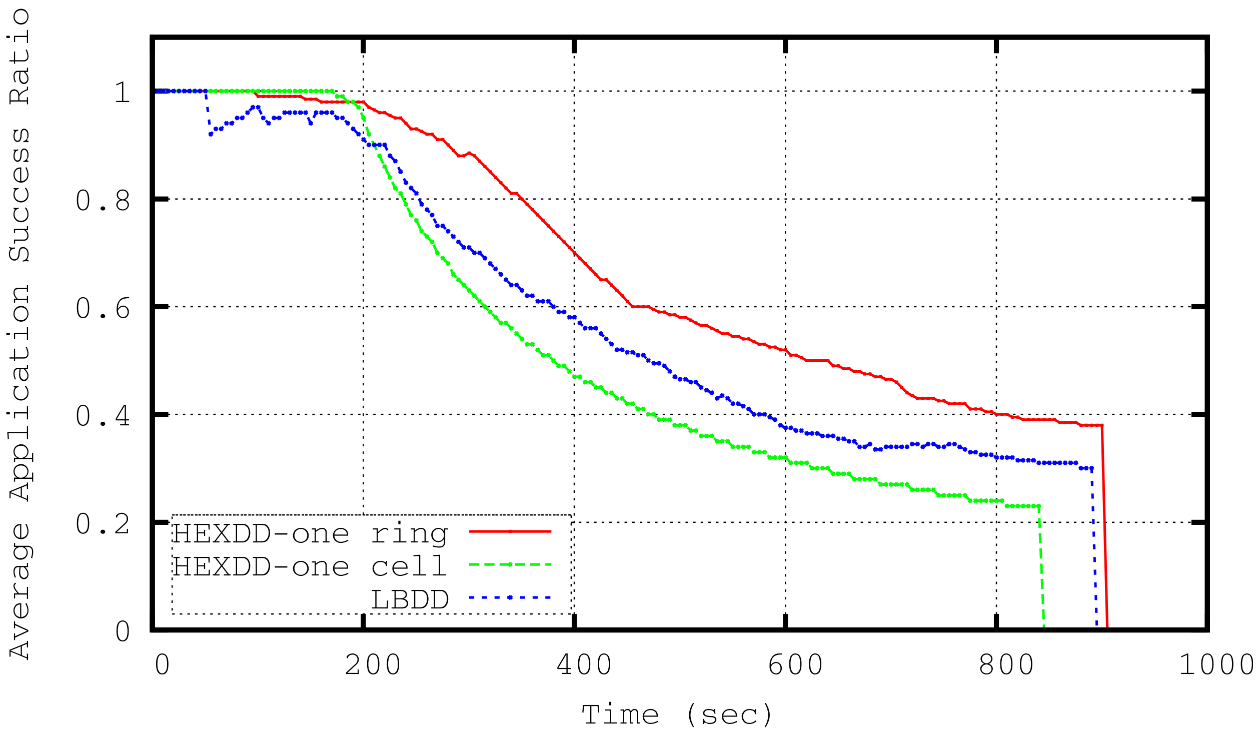
The first phase (i.e., up to 200 s in the figure) represents the normal behavior of the network when all the sensor nodes are active. HexDD with one cell at the center presents a higher average success ratio compared to LBDD and HexDD with one ring at the center during this phase. Because LBDD and HexDD with one ring at the center have to flood the sink queries within the virtual infrastructure (i.e., line strip in LBDD and hexagonal ring in HexDD) to reach the node storing the requested data, the probability of collision is thus higher and the application is less reliable. This is why their obtained average success ratio are slightly lower than HexDD with one cell at the center. In the second phase, which occurs at an instant around 200 s and the nodes start to die, LBDD and HexDD with one ring present a higher average success ratio compared to HexDD with

Table 3 Energy consumptions (W) of the three protocols

	HexDD (with one cell)	HexDD (with one ring)	LBDD	TTDD
Overall	1611	1728	2872	9468
Maximum/node	1200	600	900	1000



(a)



(b)

Figure 17 Average application success ratio: (**a**) central rendezvous region size adjustment, (**b**) average application success ratio.

one cell. Since they have a larger virtual infrastructure, the energy consumption of LBDD and HexDD with one ring is distributed over the entire rendezvous area, avoiding thus the hot spot problem and the existence of critical nodes such as the central nodes of HexDD with one cell at the center. Larger infrastructure introduces more redundancy between nodes, increasing the protocol robustness. This directly impacts the application success ratio which remains higher with LBDD and HexDD with one ring than with HexDD with one central cell.

Another way for better distributing the load or the energy consumption over the network in HexDD protocol is shifting the central cells to another location in the network (for both one-cell and one-ring approaches) and rotating the border lines over time. The central cell shifting can be easily done by taking the new position of the center as the reference point in the node-cell association process. For rotating border lines over the center cell, the whole honeycomb tessellation should be rotated to keep the same addressing scheme for routing. This requires some additional steps in the node-cell association algorithm.

7 Conclusions and future work

In this article, our goal was designing a routing protocol that supports mobility of sink and source by keeping the data delivery ratio as high as possible and data delivery delay as low as possible, while taking care of energy consumption by keeping it low. We proposed a virtual infrastructure called *honeycomb architecture* and an efficient protocol for geographical routing of event messages, *HexDD*, based on this infrastructure. The HexDD uses the concept of rendezvous region for events and queries. The border lines used as rendezvous areas, which lie on three main direction of the network, make it faster for sinks to access data. Honeycomb tessellation offers advantages in terms of algorithmic simplicity in routing and fault tolerance against node failures. HexDD makes the system resistant to node failures in the virtual infrastructure and helps in quick routing hole recovery in the network. We evaluated analytically the communication cost and hot region traffic cost of HexDD and saw that HexDD is a very suitable protocol for applications, where we have many mobile sinks and a reasonable number of sources in the network, such as emergency situation scenarios. Also, the simulation results show that our architecture helps to minimize overall energy consumption and keeps the data delivery ratio high even when routing holes exist in the network. To avoid the hot region problem, which may be observed in the border lines and the central cells, one solution is to adjust the size of the border lines and shape of the central region according to the size of the network and the network traffic. In the simulations, we show the energy distribution over the network when we have different central regions (i.e., one central cell or one central ring). As recent studies have been exploiting heterogeneity in the WSNs, deployment of higher energy and communication capacity nodes can be used at the center of the network to leverage the overall system capability of HexDD. Deploying more nodes to these regions is also another solution for the hot spot problem.

In the future, we plan to investigate the in-network data aggregation. Since the forwarding paths along the diagonals of sensor field are shared among all source-sink pairs, it provides an opportunity for similar data to meet at some common border nodes. Data from multiple sources can be aggregated and replaced by a single data packet and forwarded towards the destined sink. Our proposed scheme can achieve further performance gain by in-network data aggregation. Another issue to be considered is the effect of underlying MAC protocol on the performance of the data dissemination. It should be further investigated what impacts different MAC protocols have on the system. Designing a MAC protocol which can cooperate with networking layer can reduce the impact of MAC on the data dissemination. Since our protocol is location-based, the impact of localization errors on the performance of the protocol should also be further investigated with additional scenarios although the fault tolerance mechanism handles positioning errors. The final plan is to implement HexDD in real nodes and see the effectiveness of the protocol in the real world scenarios.

Endnotes

[a]For simplicity in Algorithm 2 we show neighbor checking for only next 1-hop cell.[b]In RailRoad and LBDD, the rendezvous region is divided into smaller subregions called *station*. All the nodes in a station are informed about the data but one of them forwards data towards sink.[c] Border line is the half of a diagonal line.

Acknowledgements
This study had been partially supported by the RECONSURVE project funded by ITEA2 and Agentschap NL.

Competing interests
The authors declare that they have no competing interests.

Received: 5 April 2011 Accepted: 16 January 2012
Published: 16 January 2012

References
1. M Marin-Perianu, C Lombriser, O Amft, P Havinga, G Troster, Distributed activity recognition with fuzzy-enabled wireless sensor networks, in *Proceedings of 4th IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '08, Santorini Springer Verlag, Greece, pp. 296–313 (2008)
2. I Krontiris, Z Benenson, T Giannetsos, F Freiling, T Dimitriou, cooperative intrusion detection in wireless sensor networks. Wirel Sensor Netw Lecture Notes in Computer Science. **5432**, 263–278 (2009). doi:10.1007/978-3-642-00224-3_17
3. S Kim, S Pakzad, D Culler, J Demmel, G Fenves, S Glaser, M Turon, Wireless sensor networks for structural health monitoring, in *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, Boulder, Colorado, USA, pp. 427–428 (2006)
4. A Erman-Tuysuz, L van Hoesel, P Havinga, J Wu, Enabling mobility in heterogeneous wireless sensor networks cooperating with UAVs for mission-critical management. IEEE Wirel Commun. **15**(6), 38–46 (2008)
5. C Intanagonwiwat, R Govindan, D Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in *Proceedings of the*

6th annual international conference on Mobile computing and networking, MobiCom'00, Boston, MA, USA, pp. 56–67 (2000)
6. E Hamida, G Chelius, Strategies for data dissemination to mobile sinks in wireless sensor networks. IEEE Wirel Commun Mag. **15**(6), 31–37 (2008)
7. A Tuysuz-Erman, P Havinga, Data dissemination of emergency messages in mobile multi-sink wireless sensor networks, in *Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, Med-Hoc-Net'10, Juan-Les-Pins, France, pp. 1–8 (2010)
8. A Tuysuz-Erman, A Dilo, P Havinga, A fault-tolerant data dissemination based on honeycomb architecture for mobile multi-sink wireless sensor networks, in *Proceedings of the 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, ISSNIP'10, Brisbane, Australia, pp. 97–102 (2010)
9. X Li, A Nayak, I Stojmenovic, Chapter 6: sink mobility in wireless sensor networks, in *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, Wiley, (2010)
10. H Sabbineni, K Chakrabarty, Datacollection in event-driven wireless sensor networks with mobile sinks. Int J Distrib Sensor Netw. **2010** (2010). Article ID 402680
11. X Chen, M Xu, A Geographical cellular-like architecture for wireless sensor networks, in *Proceedings of the 1st International Conference on Mobile Ad-hoc and Sensor Networks*, MSN'05, Wuhan, China, pp. 249–258 (2005)
12. JH Shin, D Park, A virtual infrastructure for large-scale wireless sensor networks. Comput Commun. **30**(14-15), 2853–2866 (2007). doi:10.1016/j.comcom.2007.05.020
13. C Sharp, S Schaffert, A Woo, N Sastry, C Karlof, S Sastry, D Culler, Design and implementation of a sensor network system for vehicle tracking and autonomous interception, in *Proceedings of the 2nd European Workshop on Wireless Sensor Networks, EWSN'05*, Istanbul, Turkey, pp. 93–107 (2005)
14. S Ratnasamy, B Karp, D Estrin, GHT: a geographic hash table for data-centric storage, in *Proceedings of 1st ACM International Workshop of Wireless Sensor Networks and Applications, WSNA'02*, ACM, Atlanta, GA, USA, pp. 78–87 (2002)
15. F Ye, H Luo, J Cheng, S Lu, L Zhang, A two-tier data dissemination model for large-scale wireless sensor networks, in *Proceedings of the 8th annual international conference on Mobile computing and networking, MobiCom'02*, ACM, Atlanta, GA, USA, pp. 148–159 (2002)
16. ZH Mir, YB Ko, A quadtree-based data dissemination protocol for wireless sensor networks with mobile sinks, in *Proceedings of 11th International Conference Personal Wireless Communications, PWC'06*, Albacete, Spain, pp. 447–458 (2006)
17. EB Hamida, G Chelius, A line-based data dissemination protocol for wireless sensor networks with mobile sink, in *Proceedings of the IEEE International Conference on Communications, ICC'08*, IEEE, Beijing, China, pp. 2201–2205 (2008)
18. CJ Lin, PL Chou, CF Chou, HCDD: hierarchical cluster-based data dissemination in wireless sensor networks with mobile sink, in *Proceedings of International conference on Wireless communications and mobile computing, IWCMC'06*, Vancouver, BC, Canada, pp. 1189–1194 (2006)
19. B Karp, HT Kung, GPSR: greedy perimeter stateless routing for wireless networks, in *Proceedings of the 6th annual international conference on Mobile computing and networking, MobiCom'00*, Boston, MA, USA, pp. 243–254 (2000)
20. F Kuhn, R Wattenhofer, A Zollinger, Worst-case optimal and average-case efficient geometric ad-hoc routing, in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc'03*, Annapolis, MD, USA, pp. 267–278 (2003)
21. I Stojmenovic, X Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. IEEE Trans Parallel Distrib Syst. **12**, 1023–1032 (2001). doi:10.1109/71.963415
22. M Bahrepour, N Meratnia, M Poel, Z Taghikhaki, P Havinga, Distributed event detection in wireless sensor networks for disaster management, in *Proceedings of International Conference on Intelligent Networking and Collaborative Systems, INCoS'10*, IEEE Computer Society, Thessaloniki, Greece, pp. 507–512 (2010)
23. A Okabe, B Boots, K Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, (John Wiley & Sons, New York, USA, 1992)
24. Y Xu, J Heidemann, D Estrin, Geography-informed energy conservation for Ad Hoc routing, in *Proceedings of the 7th annual international conference on Mobile computing and networking, MobiCom'01*, ACM, Rome, Italy, pp. 70–84 (2001)
25. A Sharieh, Q Mohammad, W Almobaideen, A Sliet, Hex-Cell: modeling, topological properties and routing algorithm. Eur J Sci Res. **22**(2), 457–468 (2008)
26. RP Liu, G Rogers, S Zhou, Honeycomb architecture for energy conservation in wireless sensor networks, in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM'06*, San Francisco, CA, USA, pp. 1–5 (2006)
27. X Cheng, A Thaeler, G Xue, D Chen, TPS: a time-based positioning scheme for outdoor wireless sensor networks, in *Proceedings of the 23nd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'04*, vol. 4. Hong Kong, pp. 2685–2696 (2004)
28. B Dil, P Havinga, COM-LOC++: a distributed range-free localization algorithm in wireless networks, in *Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP'10*, Brisbane, Australia, pp. 10–11 (2010)
29. A Rao, S Ratnasamy, C Papadimitriou, S Shenker, I Stoica, Geographic routing without location information, in *Proceedings of the 9th annual international conference on Mobile computing and networking, MobiCom'03*, ACM, San Diego, CA, USA, pp. 96–108 (2003)
30. JH Shin, J Kim, K Park, D Park, Railroad: virtual infrastructure for data dissemination in wireless sensor networks, in *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, PE-WASUN'05*, ACM, Montreal, QC, Canada, pp. 168–174 (2005)
31. S De, A Caruso, T Chaira, S Chessa, Bounds on hop distance in greedy routing approach in wireless ad hoc networks. Int J Wirel Mob Comput. **1**(2), 131–140 (2006). doi:10.1504/IJWMC.2006.012472
32. Network Simulator ns-2 http://www.isi.edu/nsnam/ns Accessed on 01 April 2011
33. J Zheng, M Lee, Chapter 4: a comprehensive performance study of IEEE 802.15.4, in *Sensor Network Operations*, IEEE Press, Wiley Interscience, pp. 218–237 (2006)

doi:10.1186/1687-1499-2012-17
Cite this article as: Erman *et al.*: A virtual infrastructure based on honeycomb tessellation for data dissemination in multi-sink mobile wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:17.