Chapter 19

Distributed Data Storage and Retrieval Schemes in RPL/IPv6-Based Networks

Pietro Gonizzi and Gianluigi Ferrari
University of Parma
Jérémie Leguay and Paolo Medagliani
THALES Communications and Security

Contents

19.1 Introduction and Motivations

In contrast with conventional network data storage, storing data in wireless sensor networks (WSNs) represents a challenge because of the limited power, memory, and communication bandwidth of WSNs. Nowadays, sensors have reached higher capabilities, in terms of processing speed and local storage, than in the past years [1]. This makes them more attractive for in-network storage deployments.

This chapter deals with distributed data storage and retrieval techniques in WSNs. The focus is on WSN systems for the Internet of Things (IoT), a new vision of the Internet aiming at pushing IP connectivity into smart objects. The range of objects involved in the IoT also encompasses radiofrequency identification (RFID) and machine-to-machine (M2M) systems, to cite a few. These technologies will also be described in detail.

When WSNs are deployed for IoT observation systems, they usually consist of, on one hand, unattended nodes that sense the surrounding environment and, on the other hand, a sink node that is in charge of collecting data measurements and relaying them to a management entity. There are many reasons by which a sensor node may not be able to transmit data to the sink right after acquisition. First, the sink may not always be reachable from the sensor nodes. For instance, a mobile node can be used to periodically pull out all the collected data. Second, when applications do not require real-time collection, storing data units and sending aggregate data bursts could contribute to a reduction in the number of radio transmissions, thereby increasing the lifetime operation of the WSN. Illustrative applications include habitat monitoring, such as tracking animal migrations in remote areas [2], studying weather conditions in national parks [3], etc. Such scenarios require the collection and storage of as much data as possible between two consecutive data retrievals performed by an external agent. However, storing data on the sensor node leads to local memory overflow if data retrieval is not performed in a timely manner by the sink. To avoid data dropping or overwriting, sensor nodes can cooperate with each other by sharing acquired data.

Node failure is also a critical issue in WSNs. Periodic inactivity (e.g., for energy saving purposes), physical destruction, and (software) bugs are likely to appear in WSNs, leading to data loss. Thus, redundancy through data replication (i.e., by storing copies of the same data onto various nodes) contributes to increasing the resilience of the WSN. However, distributed data storage across nodes, with or without redundancy, is a challenge because it requires properly selected donor nodes (i.e., nodes available to store data from other nodes) and entails communication overhead to transmit data to the selected nodes. Therefore, network storage capacity and resilience have an energy cost and this limits the network lifetime.

Although distributed data storage and replication have been studied in the literature [4,5], there are still a number of challenges to tackle. First, to the best of our knowledge, most previous studies did not encompass both the distribution and the replication aspects. Second, the proposed solutions are usually not tested experimentally at a large scale, although we stress how real deployment is of primary importance when developing WSN applications.

In this work, we propose a low-complexity distributed data replication mechanism to increase the resilience and storage capacity of a WSN-based observation system against node failure and local memory shortage. We extended our previous work [6], demonstrating how this mechanism can be applied with standard low-power IP sensor networks. We evaluate our approach through experimental tests conducted on the SensLAB real platform [7] and inside the Cooja [8] environment.

The chapter is organized as follows. We first review the state of the art on distributed data storage techniques and current standard IoT protocols in Section 19.2. We then present our low-complexity greedy (LG) distributed data storage scheme in Section 19.3. We apply this approach to standard IoT networks based on RPL routing protocol in Section 19.4, and treat the data retrieval issue in Section 19.5. Finally, we provide a few conclusions and perspectives in Section 19.6.

19.2 State of the Art

This section proposes to first review ongoing activities in the IoT area, before presenting more specifically related work on distributed storage.

19.2.1 Related Work on the IoT

The IoT is a recent concept relying on the integration within the Internet of a great number of communicating objects (e.g., sensors, actuators, RFID, etc.). Even if not natively connected, the objects are provided with an interface, which lets them communicate with other interconnected objects. The interconnection between devices is mainly carried out using the IPv6 protocol [9,10] and its extensions, which we will review later in this section. On top of this, M2M architectures, such as the one developed by the European Telecommunications Standards Institute (ETSI) M2M group [11], define a set of useful Representational State Transfer (REST)-based standards for the representation of devices and their data format on the one hand and for the definition of the common interfaces on the other hand. This allowed suppliers to easily create and integrate new devices, and also allowed developers to create applications and solutions based on the abovementioned devices, abstracting from their specific characteristics and implementations. Given the continuously increasing dissemination of these devices, the perspectives, in terms of both economics and of utilization, are remarkable.

Overall, the IoT provides an information system architecture in which a large number of very heterogeneous and eventually constrained devices could interconnect and interoperate without requiring a significant effort for their adaptation [12].

19.2.2 Standard Communication Protocols

One of the greatest key enablers of the IoT are the RFID devices, especially in the version described in the ISO/IEC 18000-6 standard [13]. There are two different devices according to this standard: (i) the reader and (ii) the tags. The tags are devices used for item identification and, in this case, storage of data. The reader, on the other hand, is a device in charge of interrogating the tags, which upon request, transmit their identifier. Depending on the type of battery a device is equipped with, an RFID tag may have limited transmission range and storage capabilities. Anyway, in each case, the addressing scheme used by the RFIDs is different from the scheme adopted by the IPv6, thus a device acting as a gateway between the RFID network and the IP network is required in both IoT and M2M infrastructures.

Although RFID technology originally pushed the birth of the IoT, nowadays, we are witnessing the increasing diffusion of smarter devices, with processing and storing capabilities that could be addressed through the IPv6 protocol. The ZigBee consortium has standardized these IP-communicating devices, creating a contact point between the IoT architectures [14]. This standard is suited for a family of low-rate wireless personal area networks (LR-WPANs), allowing network creation, management, and data transmission over a wireless channel with the highest

possible energy savings. This standard is based, at the first two layers of the ISO-OSI stack, on the IEEE 802.15.4 standard [15], which employs either a nonpersistent carrier sense multiple access with collision avoidance (CSMA/CA) or a slotted and beaconed medium access control (MAC) protocol. In both cases, the MAC protocol is targeted to be energy efficient, aiming to minimize collisions, and thus avoid useless packet retransmissions. Additionally, the 802.15.4 introduces the duty cycle of the RF interfaces of a transmitting device to avoid having to listen to the channel when not necessary. The ZigBee standard defines some profiles for energy efficiency and home monitoring that are totally compliant with M2M systems.

The Internet Engineering Task Force (IETF) has set the 6LowPAN, RoLL, and CoRE working groups to elaborate standard IPv6 extensions for low-power and lossy networks (LLNs) connected to the Internet. A LLN can be, for instance, a network of low-power sensors or actuators in a home or industrial automation application.

6LowPAN provides a standard adaptation layer (IETF RFC 2464, 5072, 5121) for the transport of IPv6 packets across low-power sensor networks using the IEEE 802.15.4 MAC layer for instance. The RPL (IPv6 routing protocol for LLN) [16,17] was developed to have a really limited control traffic, fit harsh and constrained environments, with limited data rate, and potentially elevated error rate. The main characteristics of this protocol will be explained in Section 19.4.1.

To efficiently collect information in LLNs, the IETF CoRE (Constrained RESTful Environments) working group has defined the constrained application protocol (CoAP) standard, supporting REST-like applications for communication between entities [18]. CoAP is a web protocol similar to HTTP, specifically developed for exploiting the functionalities of resource-constrained devices (seen as web resources). In addition, CoAP defines a mechanism for the discovery of the resources available inside a network, a multicast communication scheme, and a publish/subscribe mechanism, in which the role of initiating the communication is given to the CoAP server, which transmits the information to the subscribed CoAP clients without requiring, each time, an explicit message request by them. At the transport layer, CoAP is based on the UDP protocol, due to the reduced number of control messages introduced. To meet possible QoS requirements, because UDP is natively unreliable, CoAP has introduced the use of confirmation messages.

Especially used inside telecommunication architectures, the term M2M refers to embedded systems that are able to communicate with other systems, such as databases, applications servers, and smartphones, without requiring external human intervention. The ETSI has set a work group on this subject. At the end of 2011, the group presented the first version (release 1.0) of the ETSI M2M architecture [19]. The structure of the M2M infrastructure was split into five parts: (i) devices, (ii) area networks, (iii) gateways, (iv) core network, and (v) applications. A *device* is a node in charge of both collecting data autonomously and sending data to an application upon request. The *area network* is the element that connects the devices with a *gateway*, which in turn acts as a proxy between the *area network* and the *core network*. ETSI M2M R1 defines REST documents for representing the entities (i.e., devices, gateways, and applications), the data, and a notification mechanism. It relies on the HTTP and CoAP protocols for communications between the M2M entities.

19.2.3 Distributed Storage

Various schemes to efficiently store and process sensed data in WSNs have been proposed in the past few years [20].

In distributed WSN storage schemes, nodes cooperate to efficiently distribute data across other nodes. Previous studies focused on data-centric storage approaches [21–23], wherein data collected in some WSN regions are stored at supernodes that are responsible for these regions or for

a particular type of data. Hash values are used to distinguish data types and the corresponding storage locations. Even if this approach is based on node cooperation, it is not fully distributed because supernodes store all the contents generated by the others.

In a fully distributed data storage approach, all nodes participate in sensing and storing in the same way. First, all nodes store their sensor readings locally and, once their local memories have filled up, they delegate storage to other nodes. A first significant contribution in this direction is given by Data Farms [24]. The authors propose a fully data-distributed storage mechanism with periodic data retrieval. They derive a cost model to measure energy consumption and show how a careful selection of nodes offering storage, called *donor nodes*, optimizes the system's capacity at the price of slightly higher transmission costs. They assume that the network is built on a tree topology and each sensor node knows the return path to a sink node, which periodically retrieves data. Another study was proposed in EnviroStore [25]. The authors focus on data redistribution, when the remaining storage of a sensor node exceeds a given threshold, through load balancing. They use a proactive mechanism in which each node maintains a local memory table containing the status of the memory of its neighbors. Furthermore, mobile nodes (called *mules*) are used to carry data from an overloaded area to an offloaded one and to take data to a sink node. Takahashi et al. [26] solved the data preservation problem in an isolated sensor network using graph theory, by transferring data items from low-energy nodes to high-energy ones. However, only low-energy nodes generate content. In a study by Tseng et al. [27], each data unit is assigned a priority. High-priority traffic is distributed to nodes closer to the sink. The major shortcoming of the above studies is the absence of large-scale experiments to evaluate the system's performance, which is a key contribution of our work.

Data replication consists of adding redundancy to the system by copying data at several donor nodes (within the WSN) to mitigate the risk of node failure. It has been widely studied for WSNs and several works are available in the literature [4,28]. A scoring function for choosing a suitable replicator node was proposed in a study by Neumann et al. [4]. The function is influenced by critical parameters such as the number of desired replicas, the remaining energy of a replicator node, and the energy of the neighbors of the replicator node. In TinyDSM [5], a reactive replication approach is discussed. Data bursts are broadcast by a source node and then handled independently by each neighbor, which decides whether to store a copy of the burst or not. Maia et al. [29] suggested ProFlex, a distributed data storage protocol for replicating data measurements from constrained nodes to more powerful nodes. Unlike the above approaches, we propose a replication-based distributed data storage mechanism with lower complexity because the responsibility of finding donor nodes is not centralized at the source node but is handed over to consecutive donor nodes in a recursive manner.

Overall, with respect to the related studies, our work goes further. First, we encompass, with a fully distributed mechanism, both data replication and distributed storage. Second, we conduct large-scale experiments on the SensLAB real testbed to evaluate our solution. Third, we extend our approach on top of the RPL routing protocol to be fitted within an IoT system.

19.3 Redundant Distributed Storage

19.3.1 Introduction

In the current and next sections, we propose two redundant distributed data storage mechanisms to increase the resilience and storage capacity of a WSN against node failure and local

memory shortage. We evaluate our approaches through simulations and experimental tests conducted on the SensLAB real platform [7] and inside the Cooja [8] environment. Namely, the two mechanisms are LG and RPL greedy (RG). They mainly differ in the following aspects:

- The RG mechanism is built on top of the RPL routing protocol [16,17], and uses routing information to efficiently distribute and propagate data items through the WSN. On the other hand, LG does not rely on routing.
- In RG, replicas of a data item are preferably stored close to the sink, to be easily retrieved by a periodic agent. On the contrary, LG selects a neighbor node with the most available space, without considering its physical position.
- The two implementations are different. LG has been implemented in TinyOS 2.1.0 operating system [30], whereas RG has been developed in Contiki [31], on top of IPv6 and UDP. Therefore, they use different protocol stacks.

This section briefly describes the LG mechanism and its experimental validation performed on the SensLAB testbed. The reader can refer to the study by Gonizzi et al. [6] for a more accurate analysis. The RG mechanism will be presented in the next section.

19.3.2 LG Mechanism

We assume that a WSN is deployed to measure environmental data and store it until a sink node performs a periodic retrieval of the entire data stored in the WSN. Between two consecutive data retrievals, the objective is to distribute multiple copies of every data unit among nodes to avoid data losses caused by local memory shortages or by node failures. Data distribution relies on the proactive announcement, by every node, of its memory status. Each node periodically broadcasts a memory advertisement message containing its current available memory space. It also maintains an updated memory table containing the memory statuses of all its detected one-hop neighbors. Upon reception of a memory advertisement, a node updates its local memory table with the new information received. The memory table contains an entry for each neighbor. Each entry contains the address of the neighbor, the last received value of its available memory space, and the time at which this value was received.

When sending a replica of acquired data, a node looks up in its table the neighbor node with the largest available memory and most recent information. Such a neighbor is called the *donor node*. If no donor node can be found and there is no available space locally, then the acquired data is dropped. In particular, if a donor can be selected, the node sends to it a copy of the data unit, specifying how many other replicas are still to be distributed in the WSN. The number of required replicas is set to either $R - 1$ (if the node can store the original data locally) or R (if the node's local memory is full), where R denotes the maximum number of replicas. The replication process continues recursively in a hop-by-hop manner until either the last (Rth) replica is stored or stops when one donor node cannot find any suitable next donor node. In the latter case, the final number of replicas actually stored in the WSN, for that specific data item, is smaller than R.

In Figure 19.1, we show an illustrative example, where $R = 3$ copies, for two different cases. In Figure 19.1a, copies of a data unit generated by node 1 are stored (respectively, in nodes 1, 2, and 4). In Figure 19.1b, the replication process stops at node 2 and no suitable donor node can further be found. In this case, the last replica is not stored.

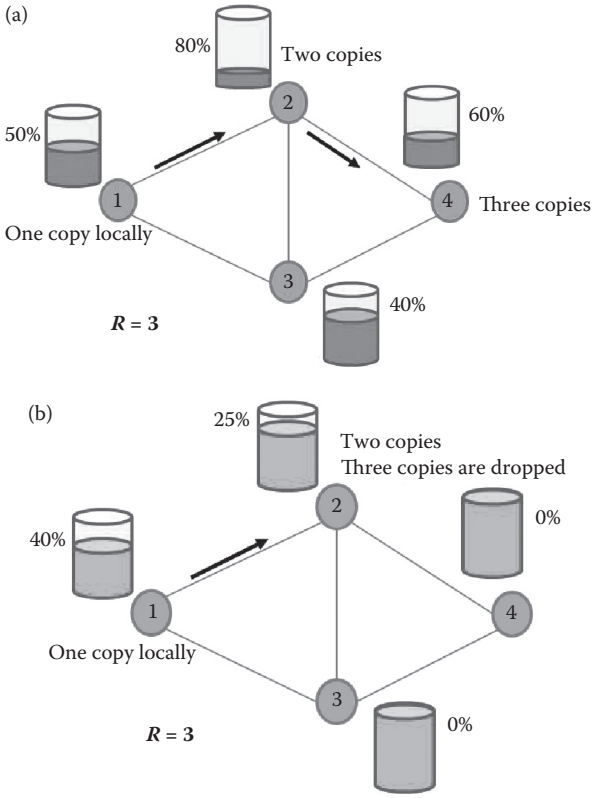

Figure 19.1 Hop-by-hop replication in the case of $R = 3$ desired replicas, for several scenarios: (a) replicas propagate up to two hops from the source node; (b) the replication process stops at an intermediate donor node and the last (Rth) copy is dropped.

At the end of the sensing period, a sink node is meant to gather all data present in the WSN by sending requests to all nodes. Upon sending the data to the sink, a node clears its local buffer and resumes sensing the environment.*

19.3.3 Performance Evaluation

The LG mechanism has been evaluated experimentally in the SensLAB testbed [7]. The SensLAB platform offers more than 1000 sensors at four sites in France (Grenoble, Strasbourg, Lille, and Rennes) where researchers can deploy their codes and run experiments. At each site, nodes are installed on an almost regular grid, as depicted in Figure 19.2. Each node's platform embeds a TI MSP430 microcontroller and operates in various frequency bands depending on the radio chip (either CC1100 or CC2420).

The LG mechanism is significantly influenced by the network topology. For instance, in a dense network, where nodes have several neighbors, data can be distributed more efficiently than in a sparse network with only a few direct neighbors per node. The network topology depends on

* We remark that data retrieval, with LG distributed storage, has not been implemented, as the protocol stack being considered does not rely on any specific multihop routing protocol.

Figure 19.2 Views from the SensLAB sites.

the transmission power of the nodes. We have run four experiments setting the transmit power P_t to 8.9, 0, –15, and –20 dBm, respectively, deploying $N = 78$ nodes. Each node has a finite local memory with size equal to 250 data units, and a sensing period chosen randomly in the interval 0.1 to 5.1 s. We have computed the number of neighbors detected in each case by counting the number of memory advertisements received by each node. We assume node x is a neighbor of node y if node y detects at least one memory advertisement from node x, within the overall experiment duration. Note that the larger the number of received memory advertisements, the higher the radio link quality.

It is of interest to evaluate the time required by the system to reach the network storage capacity, which denotes the maximum amount of data that can be stored in the WSN, in the different cases. According to a study by Gonizzi et al. [6], given the number of nodes N, the buffer sizes $\{B_i\}$, and the sensing rates $\{r_i\}$, the network storage capacity C (dimension: data units), in the absence of replications, is simply given by

$$C = \sum_{i=1}^{N} B_i \tag{19.1}$$

whereas in the case of data replication (i.e., with $R > 1$), the system has a resulting storage capacity, denoted by C_r, which is upper-bounded by C. In this case, the capacity is reduced by the maximum number R of replicas and can be lower-bounded as follows:

$$C_r \geq \frac{C}{R} \tag{19.2}$$

Obviously, $C_r = C$ when $R = 1$, that is, only the original copy is kept and no replication is performed at all.

In the case of absence of replicas, given the fixed memory size and the number of nodes equal to 250 and 78, respectively, the capacity is $C = 250 \times 78 = 19{,}500$ data units. In Figure 19.3a, the amount of data stored is shown as a function of time for the four experimental cases being considered.

As one can see, capacity is reached later when a lower transmission power is used—for instance at –20 dBm—because fewer neighbors are detected. Consequently, data cannot be distributed efficiently through the network. On the other hand, with a transmit power equal to 8.9 dBm, nodes have a larger view of the network and the capacity is reached earlier. For instance, at $t = 400$ s, the stored data at 8.9 and 0 dBm are 17,000 data units, approximately 90% of the capacity. The stored data at –15 and –20 dBm, at the same time instances, are approximately 74% and 68% of the total capacity, respectively. Moreover, two theoretical cases are considered for comparison. In the case with local storage [Local storage (anal) curve], nodes fill their own local buffer autonomously, that is, the fill-up time for the i node is $t_i = B_i/r_i$. In this case, the time interval to reach

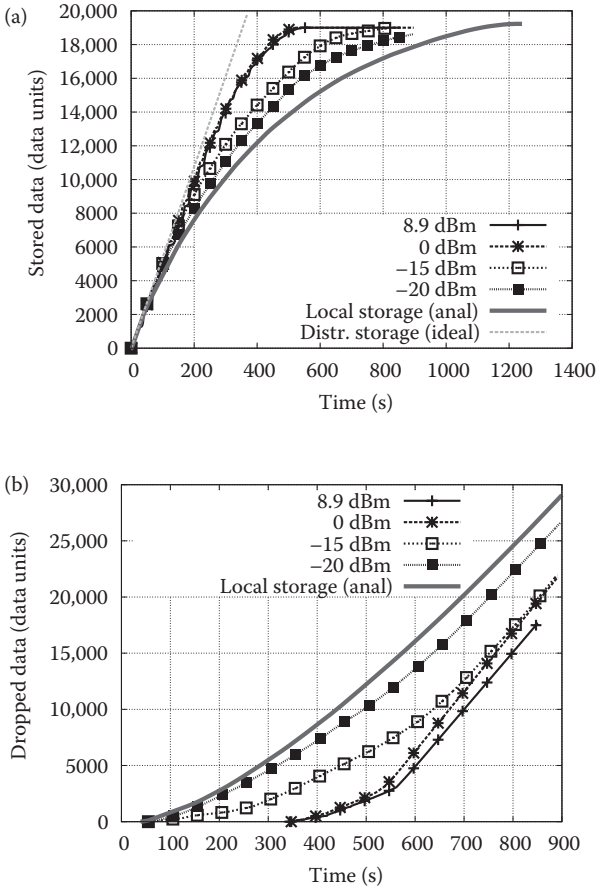

Figure 19.3 (a) Stored data and (b) dropped data in the system for various values of the transmit power, as a function of time. In all cases, we consider: no replication ($R = 1$), memory size equal to 250 data units, $N = 78$ nodes, and a memory advertisement period of 25 s.

the storage capacity corresponds to the longest storage filling time across all nodes. As expected, the analytical curve relative to local storage lower bounds the experimental curves. In the case with ideal distributed storage ["Distr. storage (ideal)" curve], a performance benchmark can be obtained considering an ideal WSN in which nodes can communicate with any other node, considering instantaneous transmission. In this case, the WSN is equivalent to a single supernode with memory size equal to the total capacity C and sensing rate equal to the sum of the individual sensing rates of all nodes.

In Figure 19.3b, the amount of dropped data is shown, as a function of time, in the four cases considered in Figure 19.3a. Nodes drop newly acquired data once their local memory has filled up and no neighbors are available to donate extra storage space. At t = 400 s, dropped data at 8.9 and 0 dBm equal 600 data units, approximately 3% of the stored data. In the cases with lower transmit power, for example, at –15 and –20 dBm, the amount of dropped data is significantly higher. As expected, data dropping starts in advance with lower transmission power because each node has a lower number of surrounding donor nodes to which to transfer data. In the same figure, the amount of dropped data, in the case with local storage, is also shown for comparison.

As for redundancy, replication is introduced by setting the number of copies (referred to as replicas) to a value $R > 1$. Replicas of a sensed data unit follow a hop-by-hop replication from the generator node to subsequent donor nodes. We have computed the average hop distance reached by the replicas from the generator node, for various values of R. The results show that replicas do not propagate, on average, beyond two hops from the generator node (Figure 19.4). This is in agreement with LG because donor nodes are selected on the basis of their available memory space but not their physical position.

In conclusion, LG adopts a low-complexity data distribution mechanism that performs well in a dense network with high connectivity. However, replicas of a data item are kept in the proximity of the generator node, on average within two hops: this may lead to the complete loss of some data in the case of a failure that damages all the nodes within a certain region. To reduce this risk, in the next section, an enhanced data distribution mechanism based on RPL routing protocol is presented.

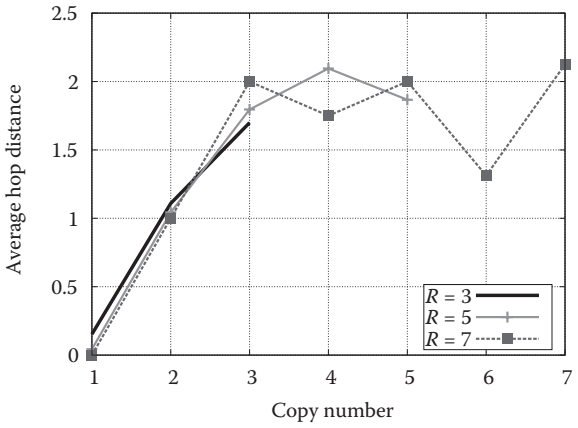

Figure 19.4 Average hop distance reached by the kth replica versus k. k varies between 1 and 3 (R = 3), 1 and 5 (R = 5), and 1 and 7 (R = 7). The average is computed on all the unique data present in the system with full redundancy, that is, with exactly R copies.

19.4 Redundant Distributed Storage with RPL

In the previous section, we have presented LG, a LG mechanism for distributed data storage. A drawback of LG is that replicas of a given data item do not spread throughout the WSN. In this section, an alternative approach, called RG, is described. RG follows a similar approach to LG, but includes information on the position of the node within the WSN. The position of a node is suggested by the RPL rank, a value provided by the RPL routing protocol, which indicates the position of a node within a routing tree, according to a routing function.

This section is organized as follows. First, an overview of RPL is given in Section 19.4.1. A detailed description of RG highlighting the main differences with LG is provided in Section 19.4.2. In Section 19.4.3, we evaluate the protocol through extensive simulations conducted in Cooja, the WSN simulator in Contiki. And finally, Section 19.4.4 concludes the discussion about data storage and introduces the data retrieval problem, which is the topic of the last part of this chapter.

19.4.1 Overview of RPL

RPL is a distance-vector protocol that creates a routing tree, referred to as destination-oriented acyclic-directed graph (DODAG), in which the cost of each path is evaluated according to the metrics defined in an objective function (OF). The goal of this protocol is the creation of a collection tree protocol (CTP), and a point-to-multipoint network from the root of the network to the devices inside the LLN, as well as a point-to-point network between any pair of devices.

To build the tree and to keep the status of the network updated, the root of the RPL tree periodically sends DODAG information object (DIO) messages. The receiving nodes may relay these messages or just consume them, if configured as leaves of the tree. The mechanism of RPL is quite simple: each node has a rank that places it in the hierarchy of the RPL tree and lets it define which nodes are its parents. When a DIO message is received for the first time, a node, before setting its rank, listens through the network discovery mechanism for the possible parents to which it can join. At the end of this discovery phase, according to the outcomes of the OF, a node sets its rank to be highest among the ranks of its possible fathers. To avoid loops or network misconfiguration, two nodes in the same network are not allowed to have the same rank; therefore, as soon as such a situation is detected, one of the conflicting nodes updates its status. According to the outcome of the OF, the node also selects the best path that it can use to transfer the data to the root of the tree. On the other side, if a DIO message has already been received at least once, the node evaluates the incoming DIO message to check whether its position in the DODAG tree can remain the same or must be updated. In the former case, the node discards the packet, whereas in the latter, it computes its new rank and it discards the list of parents to avoid creating loops due to its new position in the DODAG tree. In Figure 19.5, we show the operations that a node carries out to establish its role in the DODAG tree when it receives a DIO packet.

Because devices in LLNs are typically resource constrained, the RPL protocol also introduces a trickle mechanism to reduce the transmission frequency of DIO messages according to the stability of the network. If the network status is stable, the frequency of transmission of DIO messages decreases. As soon as an anomaly or an inconsistency within the network is detected, the frequency is kept back to the default value and a procedure of recovery is started. If the chosen procedure is a local repair, the node simply selects a new best parent. Otherwise, in the case of a global repair, the root sends a new DIO message to restart the construction of the tree.

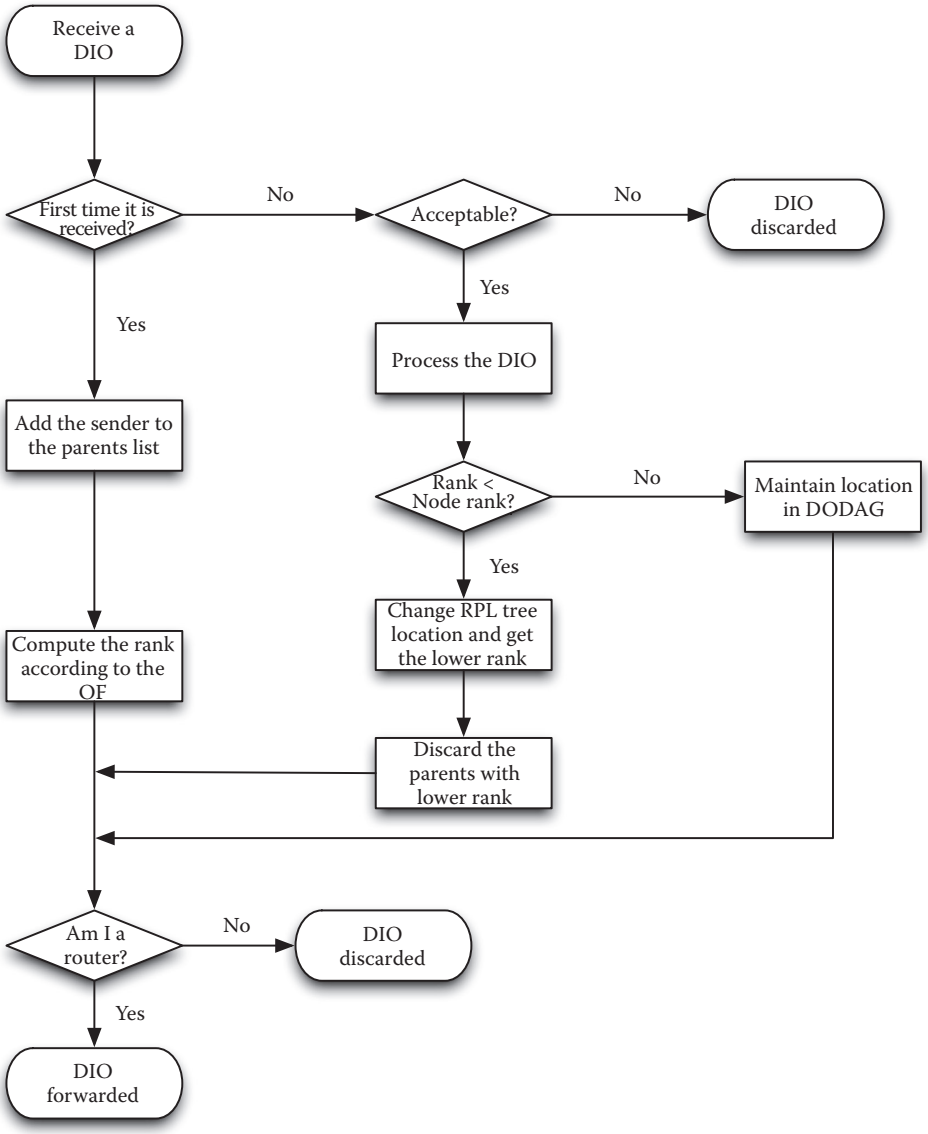

Figure 19.5 Diagram of the creation of a RPL tree in a node belonging to a DODAG tree.

Because information may also flow in the other direction, that is, from the root to the leaves, and because the communication links are asymmetric, RPL has also defined a strategy for the construction of downward routes. In this case, the remote nodes send destination advertisement object (DAO) messages to the root. While traveling back to the root, the intermediate nodes' addresses are stored in the packet, so that a complete route from the root to the node is created. There are two ways of operating while creating a downward route: (i) *storing* and (ii) *nonstoring* mode. In the former case, a message is sent from the remote node to the root and each intermediate parent node stores the addresses of the children from which it has received a DAO. In the latter, intermediate nodes do not store the addresses of their children but they are only limited to

inserting their address into the DAO instead, transferring it to the root, which will be the only node with a complete topology to transfer information downward.

The control messages exchanged by RPL can also be used to convey additional information, depending on the specific application requirements. This technique, referred to as *piggybacking*, allows a significantly reduced amount of exchanged messages. For instance, it can be used to transfer information about the status of a node.

19.4.2 RG Mechanism

The RG algorithm is based on the same principles of LG: nodes of the WSN, upon joining a RPL DODAG, keep on collecting data (acquired with a given sensing rate). Data retrieval is performed by an external agent that periodically connects to the DODAG root and gathers all the data from the WSN.*

To prevent data losses, data is stored in several nodes (possibly including the generating node). This consists of copying and distributing replicas of the same data to other nodes with some available memory. As in LG, information about memory availability is periodically broadcast, by each node, to all direct neighbors.

The main parameters are listed in Table 19.1. Without loss of generality, we consider a WSN with N fixed RPL nodes deployed over an area whose surface is A (dimension: m^2). Nodes only interact with one-hop neighbors, that is, with nodes within the radio transmission range d (dimension: m). Note that an additional node acts as the RPL DODAG root, even if it does not participate in sensing and storage. Therefore, the final number of nodes is $N + 1$. The ith node has a finite local buffer of size B_i (dimension: data units) and sensing rate r_i (dimension: data units/s).

Each node broadcasts without acknowledgement and every T_{adv} (dimension: s), its memory status to all nodes within direct transmission range (i.e., one-hop neighbors). Each memory advertisement consists of six fields relative to the sending node: the RPL rank of the node, value of sensing rate, up-to-date available memory space, an aggregate that indicates the status of node memories in the down direction in the DODAG, an analogous value for the up direction, and a sequence number. Each node maintains a table that records the latest memory status received from neighbor nodes. Upon reception of a memory advertisement from a neighbor, a node updates its memory table, using the sequence number field to discard multiple receptions or out-of-date advertisements. The aggregate of the status of node memories in the down direction in the DODAG is given the minimum hop distance at which a node with some available space can be found. This distance is computed as follows: if a node detects that at least one of its children (i.e., neighbors with higher RPL rank) has some space locally, it sets this distance to 1. Otherwise, a parent increments by 1 the value of the minimum distance given by its children. Once the distance reaches a maximum value, a node assumes that there is no available memory in the down direction of the DODAG. The maximum value of the flag is given by the *MaxHopDown* parameter, listed in Table 19.1. Similarly, the status of the node memories in the up direction is computed in the same way, but in the inverse direction of the DODAG; in this case, the maximum hop distance that can be announced in a memory advertisement is given by the *MaxHopUp* parameter.

The first message exchange, depicted in Figure 19.6a, is between node 3 and node 1. Node 3 transfers a memory advertisement saying that it has no available space but one of its one-hop children does have available space, as stated by the minimum hop distance downward (mhd_{down})

* RPLC-based data retrieval is the focus of the next section.

Table 19.1 Main System Parameters

Symbol	*Description*	*Unit*
N	Number of RPL nodes	scalar
A	Surface of deployment area	m^2
d	Node transmission range	m
B_i	Node i's buffer size, $i \in \{1,\ldots,N\}$	scalar
r_i	Node i's sensing rate, $i \in \{1,\ldots,N\}$	s^{-1}
MaxHopDown	Maximum distance (in the down direction), that can be announced in a memory advertisement, at which some available space is present	scalar
MaxHopUp	Maximum distance (in the up direction), that can be announced in a memory advertisement, at which some available space is present	scalar
T_{adv}	Period of memory advertisement (from each node)	s
R	Maximum number of replicas per sensing data unit	scalar
T	Period of data retrieval (from the sink)	s

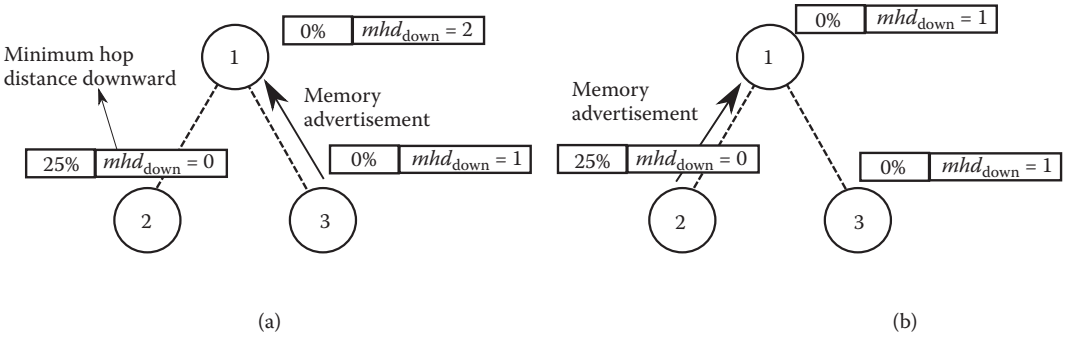

Figure 19.6 Messages exchanged for memory advertisements. In (a), node 3 transfers a memory advertisement saying that it has no available space but one of its one-hop children does have available space, as stated by the minimum hop distance downward (mhd_{down}) parameter. Node 1, which has no available memory, then sets mhd_{down} to 2, because it has received information that the closest node with available memory is at a two-hop distance. In (b), node 2 sends its memory advertisement saying that it has available space. Consequently, node 1 becomes aware that there is a closer node with available space, so it updates its mhd_{down} to 1.

parameter. Node 1, which has no available memory, then sets mhd_{down} to 2, because it has received information that the closest node with available memory is at a two-hop distance. Then, as shown in Figure 19.6b, node 2 sends its memory advertisement saying that it has available space. Consequently, node 1 becomes aware that there is a closer node with available space, so it updates its mhd_{down} to 1. An illustrative scenario is shown in Figure 19.6.

Like LG, the RG mechanism is fully decentralized, in the sense that all nodes play the same role. It consists of creating at most R copies of each data unit generated by a node and distributes them across the network, storing at most one copy per node, possibly closer to the DODAG root to later reduce the energy consumption of the retrieval phase. Each copy is referred to as a *replica*. Let us focus on node $i \in \{1,\ldots, N\}$. At time t, the node generates (upon sensing) a data unit. The memory table of node i contains one entry per direct neighbor. Node i selects from its memory table the neighbor node, called a *donor*, with the largest available memory space and the most

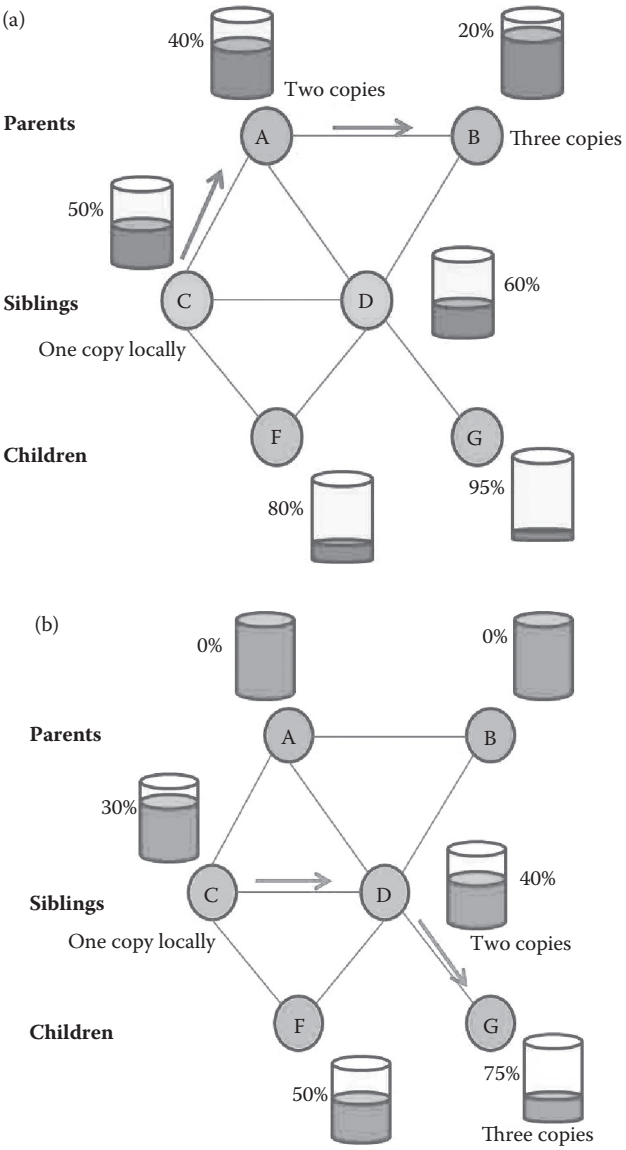

Figure 19.7 Hop-by-hop replication in the case of $R = 3$ desired replicas, for several scenarios: (a) replicas propagate to nodes closer to the RPL root first; (b) data is distributed to nodes in the down direction of the tree as the memories of the parents are full.

recent information. Moreover, priority is given to those donors that are parents of node *i* in the tree, that is, nodes with lower rank. If no parents can be selected, node *i* looks for a sibling in the tree, that is, a node with the same parent, providing that such a node has some available space. If no sibling can be chosen, node *i* searches for a child. In case all neighbors have no space locally, then node *i* checks if a neighbor at least knows about some available space beyond one hop, that is, in the up direction or in the down direction of the DODAG. In this case, again, priority is given to nodes in the up direction. If there is no suitable neighbor in the memory table, there is no possibility to distribute replicas of the data unit across the network. In this case, only one copy can be stored in the local memory of node *i*, if *i* has some space locally.

If a donor node can be selected, node *i* sends it a copy of the data unit, specifying how many other copies are still to be distributed in the WSN. As in LG, the number of required copies is set to either $R - 1$ (if node *i* can store the original data locally) or R (if node *i*'s local memory is full). Upon reception of the copy, the donor node stores the copy in its memory, if it has some space locally, and selects the next donor node among its neighbors, discarding the sending node and the source node from the candidate nodes. The next donor is chosen such that its RPL rank diverges from that of the generator node. This causes replicas to spread well throughout the RPL tree. At this point, the donor sends the copy to the next chosen donor node, decrementing the number of required copies by 1. The replication process continues recursively until either the last (Rth) copy is stored or stops when one donor node cannot find any suitable next donor node. In the latter case, the final number of copies actually stored in the WSN is smaller than R. Note that, if a donor cannot find a next neighbor with a suitable RPL rank, the replica may follow a different reversed path along the DODAG, and retake the original direction later.

In Figure 19.7, we show an illustrative example with $R = 3$ desired replicas. In Figure 19.7a, nodes always try to distribute replicas to parents in the up direction of the RPL tree, for example, nodes with a lower rank. As the network is saturated, data is distributed toward nodes with the same rank or with a higher rank, that is, in the down direction of the DODAG, as shown in Figure 19.7b.

19.4.3 Performance Evaluation

RG has been implemented in Contiki 2.5 and evaluated in Cooja, a Java-based WSN simulator [8]. The scenario is depicted in Figure 19.8. A rectangular grid is composed of 60 storing nodes, shown in gray. Each node inside the grid communicates with four direct neighbors. Moreover, to simulate real conditions, the node interferes with some extra nodes: collision occurs if a node and at least one among its direct neighbors or its interfering nodes transmit a packet at the same time. For example, referring to Figure 19.8, the neighbors of node 36 are nodes 50, 51, 55, and 43. The interfering nodes, shown between the two circles, are nodes 29, 54, 56, and 57. Collisions may also be caused by the hidden terminal problem [32]. A 100% packet delivery ratio (PDR) is assumed, that is, packets are always delivered, providing that no collision has occurred. Note that nodes along the borders have fewer neighbors than the others. Finally, node 1, in the upper left part of the grid, acts as the RPL root, creates the routing topology, and performs periodic data retrieval. In Figure 19.9, a detailed view of the scenario is depicted. Nodes at the same hop distance from the RPL root are drawn with the same color. The maximum number of hops is equivalent to 13. It has been observed that RPL sometimes modifies the structure of the tree; therefore, a node may change its RPL rank.

The interval between two consecutive data retrievals is $T = 10$ min. The sensing period of the nodes is an integer number chosen randomly and independently in the range of 31 to 33 s.

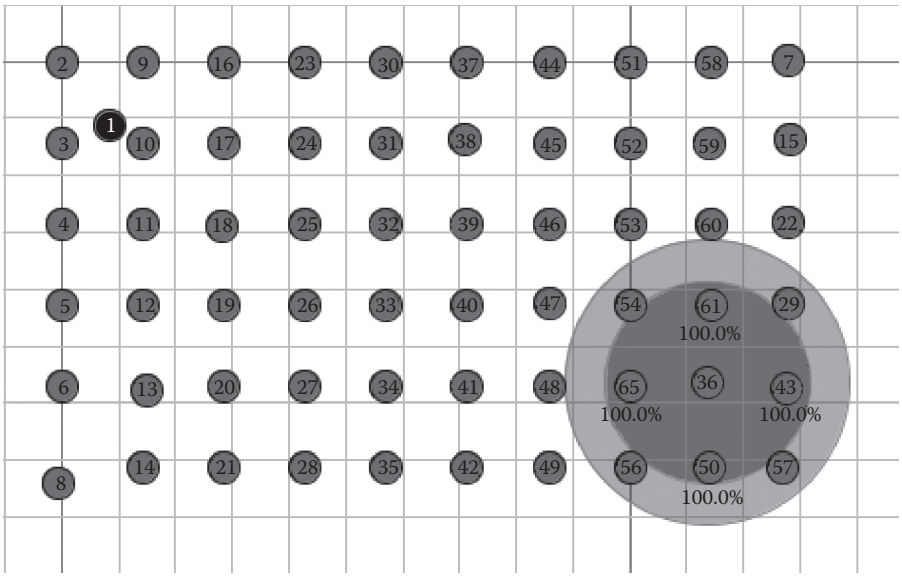

Figure 19.8 Scenario evaluated in Cooja. Sixty storage nodes (shown in gray) are deployed in a regular grid. Each node has four direct neighbors. The RPL root is node 1 of the figure (shown in black).

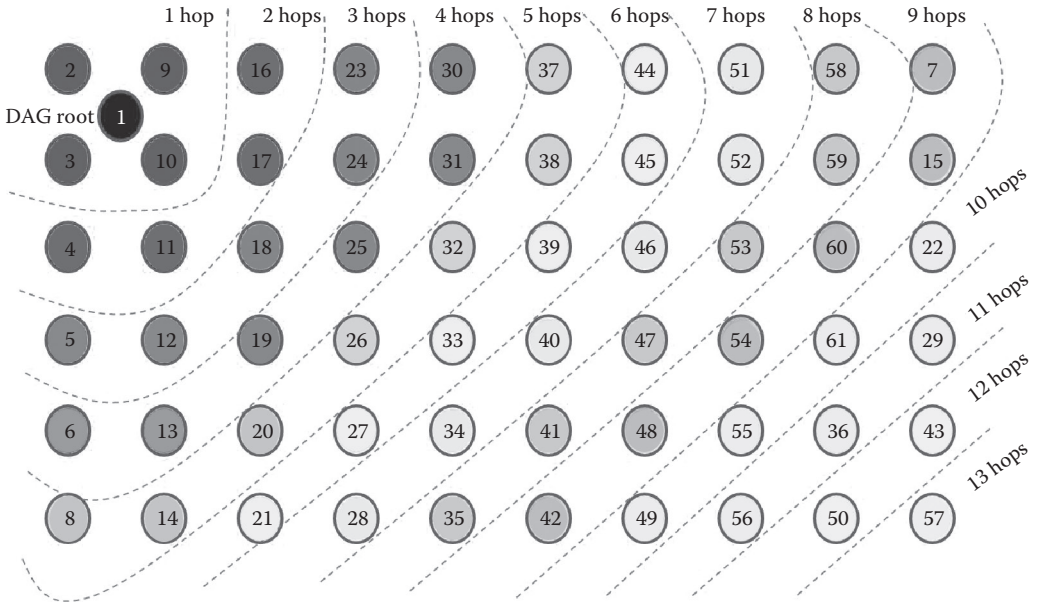

Figure 19.9 Scenario evaluated in Cooja. Sixty storage nodes are deployed in a regular grid. Each node has four direct neighbors. The RPL root is node 1 of the figure. The network has a total size equal to 13 hops.

All nodes have the same buffer size, which equivalent to $B = 100$ data units. The memory advertisement period T_{adv} is set to 30 s. We adopt the expected transmission count (ETX) as an RPL metric. According to this metric, the routing function tends to minimize the number of expected transmissions toward the RPL root. In the presence of perfectly reliable links, it is equivalent to minimizing the number of hops.

19.4.3.1 Effect of the Number of Replicas

We have run four simulations, setting the number of replicas R to 1, 3, 5, and 7, respectively. Note that the case with $R = 1$ is without replication, that is, only the original copy is stored.

The time required by the system to reach the network storage capacity has been computed in the different cases. With the setting described above, the network storage capacity C, which denotes the maximum amount of data that can be stored in the WSN, is equivalent to $C = N \times B = 60 \times 100 = 6000$ data units. In Figure 19.10a, the amount of total stored data is shown, as a function of time, for the four considered simulated cases. It can be observed that the capacity is reached later when fewer replicas are used—for instance when $R = 1$—because less data is generated. On the other hand, for higher values of R, capacity is reached earlier. However, the slope of the curve, in the case with $R = 3$, tends to approximate the ones obtained with $R = 5$ and $R = 7$. Because a higher storing rate quickly saturates the memories of the nodes at the beginning of the simulation and leads to a less efficient data distribution later.

In Figure 19.10b, the amount of dropped data is shown as a function of time. Nodes drop newly acquired data once their local memory has filled up and no neighbors are available for donating extra storage space. Remember that data is marked as dropped if no replicas at all can be stored. This explains why related curves with $R = 3$ and $R = 5$ are close to each other. This suggests that it is inconvenient to store exactly R replicas for each sensed data unit because the first replicas that are stored would quickly congest the memories and impede the next ones to be stored.

In Figure 19.10c, the amount of unique stored data is shown as a function of time. In fact, in the presence of redundancy ($R > 1$) several copies of the same data unit are distributed in the WSN, so that the amount of original node data is smaller than the total amount of actually stored data. The analysis of the unique stored data is expedient to evaluate the efficiency of data retrieval, as will be shown in Section 19.5.

At this point, it is of interest to evaluate the data placement throughout the WSN over time. As discussed previously, RG distributes replicas prioritizing donor nodes closer to the root. Figure 19.11 shows the average saturation level of the memories of the nodes at different hop distances from the root, for several observation instances. It can be noticed that, according to RG, the portion of the RPL tree closer to the root fills the memories faster. This can be of help in the data retrieval phase because data follows a shorter path to reach the sink.

As for the distribution of the replicas, Figure 19.12 shows the average hop distance reached by the redundant copies, from the owner of the original one for different cases with $R > 1$. Results show that replicas keep on moving away from each other, passing through nodes with a higher RPL rank in the tree. By comparing the results in Figure 19.12 with those, relative to the LG mechanism, shown in Figure 19.4, it can be observed that with the RG mechanism, replicas tend to spread farther from the originating node than with the LG mechanism. Therefore, the RG mechanism leads to a more resilient data preservation in the presence of a failure involving a source node and several of its neighboring nodes.

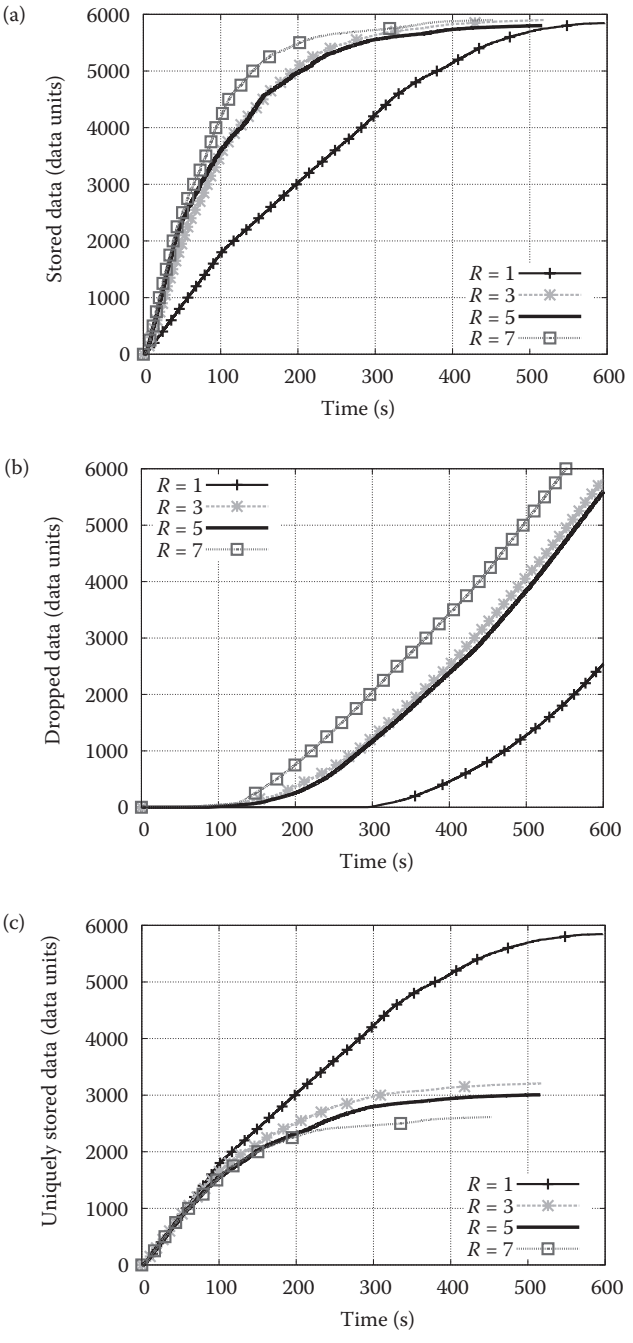

Figure 19.10 (a) Stored data, (b) dropped data, and (c) unique stored data in the system for various values of replicas *R*, as a function of time. In all cases, we consider: memory size equal to 100 data units, *N* = 60 storage nodes, and a memory advertisement period of 30 s.

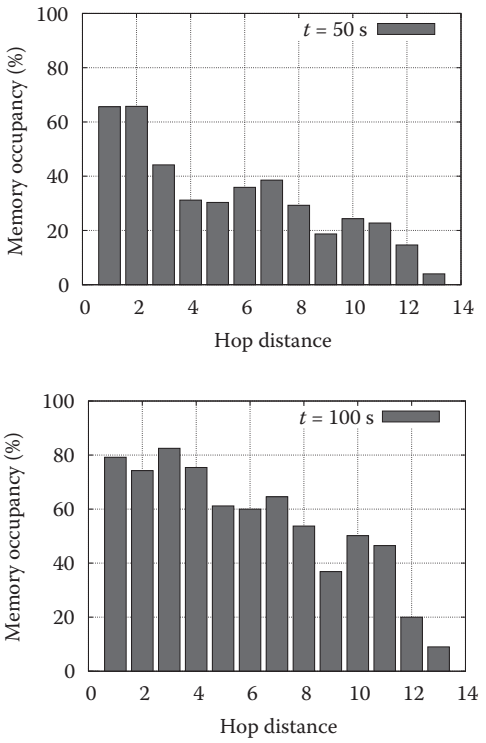

Figure 19.11 Percentage of memory occupancy varying the hop distance from the RPL root, at different time instances. Memories of nodes closer to the RPL root saturate faster. The number of replicas *R* is set to 7.

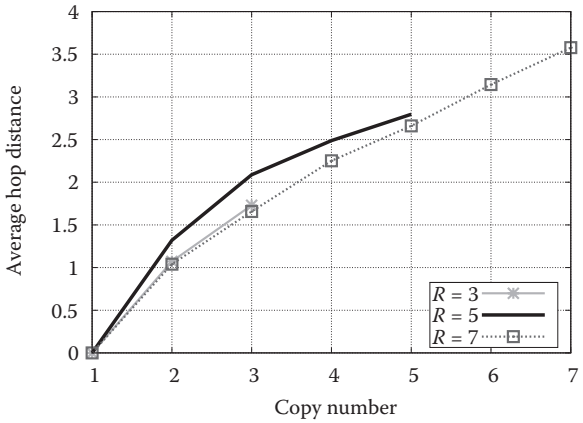

Figure 19.12 Average hop distance reached by the *k*th replica versus *k*. *k* varies between 1 and 3 (*R* = 3), 1 and 5 (*R* = 5), 1 and 7 (*R* = 7). The distance is calculated from the position of the first replica.

19.4.4 How to Retrieve the Stored Data?

In Section 19.3 and in the current section, two redundant data distribution mechanisms have been presented: the LG mechanism increases the network storage capacity with a slight signaling overhead; the RG mechanism, owing to a more accurate view of the network status (thanks to routing information), allows replicas to spread better across the WSN. Although the two proposed mechanisms deal with distributed storage, data retrieval is fundamental to collecting (to a sink) the stored data.

Data retrieval consists of a periodic collection of the whole sensed data, stored in the WSN, at a single node in the network, usually denoted as a sink. The sink then passes all the collected data to a monitoring center for further processing. During the retrieval phase, the major challenges are the following:

- Sending lot of data depletes the battery of the nodes, thus the lifetime of the WSN.
- Multihop communication is required for those nodes which are far away from the sink. Therefore, battery depletion is even more critical for intermediate nodes that have to forward the data of the others.
- The amount of stored data in the WSN can be extremely high, if the retrieval is not performed in a timely manner. Sending all the data instantaneously to the sink can cause many collisions, thus losing a lot of information. A solution could be letting the nodes to wait some time between two consecutive data transmissions. However, this may excessively prolong the duration of the retrieval.

Although the LG mechanism does not lead directly to a data retrieval phase, the RG does, as it builds on the RPL routing algorithm. The next section is dedicated to data retrieval with RPL and all the aspects outlined above will be covered in depth.

19.5 Data Retrieval with RPL

Data retrieval consists of forwarding the collected sensing data of the WSN to a central base station for further processing. In the literature, it often appears with various terms, for example, *data collection* and *data gathering*. This section briefly reviews some existing works in this field and presents the data retrieval mechanism that has been implemented and evaluated in RG.

19.5.1 Related Work

As stated in an article by Wang and Liu [34], data retrieval in WSNs is still in its early stage. Two main approaches can be identified: mobile data retrieval and fixed data retrieval.

Mobile data retrieval has been studied in the literature. The idea is to have a mobile entity that travels across the WSN and gathers data from every sensor node within the communication range. This should save battery at the sensors and increase the lifetime of the network. Usually, no specific routing topology is required, as the communication between the mobile sink and the sensor node is single hop. In a study by Gao and Zhang [35], the authors investigated the optimal path selection for a mobile sink in a path-constrained scenario with delay requirements. In a study by Jain et al. [36], a rich analytical framework to measure data retrieval rate, latency, and consumed power is given. Several protocols to optimize data transfer at the minimum energy cost have been

proposed [37]. And in a study by Somasundara et al. [38], the problem of data collection scheduling to avoid node's buffer overflow is investigated. For a detailed survey on mobile data retrieval, please refer to the article by Francesco et al. [33]. In our work, we consider periodic data retrieval performed by a fixed sink node.

As for fixed data retrieval, a first approach, discussed in a study by Tseng et al. [27], consists of a sink node that periodically stops at a fixed point in the network. Once this point is reached, the sink advertises its presence by sending queries. Each sensor node builds its own return path to the sink and consequently sends the collected data. In such a way, a routing tree toward the sink is formed. The CTP is probably the routing mechanism most frequently used for multihop fixed data retrieval in sensor networks [39]. The strengths of CTP are its ability to quickly discover and repair path inconsistencies and its adaptive beaconing, which reduces protocol overhead and allows for low radio duty cycles. Extended versions of CTP have been proposed to deal with the nodes' mobility [40]. Dozer [41] is a data retrieval protocol aimed at achieving extremely low energy consumption. It builds a tree structure used to convey data to the sink, enriched with a time division multiple access (TDMA) scheme at the MAC layer to synchronize the nodes. In Koala [42], a data-gathering system is proposed. In contrast with Dozer, routes are built at the sink and no persistent routing state is maintained on the motes. Coupled with a low-power probing (LPP) technique at the MAC layer, Koala can achieve very low duty cycles.

With respect to the related works, we consider periodic data retrieval in an isolated RPL-based network. Instead of a cross-design among MAC layer, topology control, routing, and scheduling, we let the RPL protocol handle the routing structure autonomously. Moreover, we focus more on data availability at the sink and on latency of the retrieval than on energy efficiency.

19.5.2 Data Retrieval Mechanism with RG

We describe here the data retrieval mechanism that has been designed and implemented to work together with the RG distributed data storage mechanism proposed in Section 19.4.

Recall from Section 19.4.2 that the RG mechanism replicates and distributes copies of sensed data units toward nodes closer to the sink. The sink is supposed to periodically send retrieval requests through the RPL root. In particular, the retrieval request is broadcast by each node of the tree only once. Subsequent receptions of the same retrieval request are ignored. Moreover, a sequence number is used by the RPL root to identify each periodic retrieval request.

Data collection takes place from each sensor node toward the RPL root. Such many-to-one traffic patterns, if not carefully handled, can cause (i) many collisions and (ii) highly unbalanced and inefficient energy consumption in the whole network. To reduce these risks, the RG mechanism has been enriched with the following: A replica of a given data item is sent to the sink only if it is the closest to the RPL root among all the stored replicas of that data item. To let a node be informed that it holds a replica closer to the root, donor nodes include their IPv6 address and RPL rank within a data packet during the distribution phase. Because RG follows a hop-by-hop greedy replication scheme, the next donor node along the chain checks the rank of the closest donor node that has stored the replica before, and compares it with its own rank. In case its own rank is lower than the value announced in the data packet, that is, the node is closer to the RPL root than its ancestor donor, and providing that the node stores the replica in its local memory, then the node informs the ancestor donor about the newly closer position of the replica. In the retrieval phase, the ancestor donor will not send such a replica to the sink, as it knows that it is not the closest one. An illustrative example is depicted in Figure 19.13. In Figure 19.13a, replicas

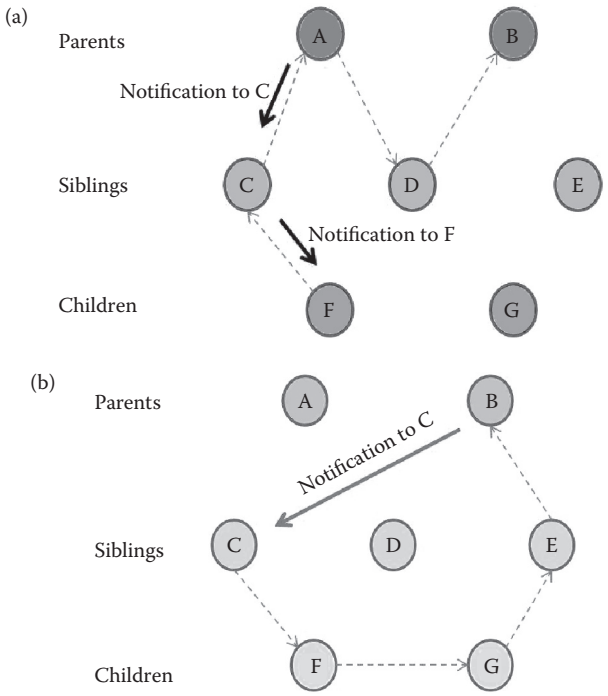

Figure 19.13 During the data retrieval phase, only the closest replica is sent to the sink. A donor node with lower rank informs the prior best donor about the new position of the replica. In (a), replicas of a data item, generated at node *F*, are stored, progressively, at nodes *C*, *A*, *D*, and *B*. Node *C* informs node *F* of its better position in the tree (*C* has rank *k*, whereas *F* has rank *k* + 1, with *k* > 0). Similarly, node *A* has to inform node *C* for the same reason. Note that *D* and *B* do not send notification to *A*, as they are not closer to the RPL root. In (b), the notification is exchanged only from *B* to *C*.

of a data item generated at node F are stored, progressively, at nodes C, A, D, and B. Node C informs node F of its better position in the tree (C has rank k, whereas F has rank $k + 1$, with $k > 0$). Similarly, node A has to inform node C for the same reason. Note that D and B do not send notification to A, as they are not closer to the RPL root. In Figure 19.13b, notification is delivered only from B to C.

19.5.3 Performance Evaluation

The data retrieval scheme developed in RG has been evaluated with the same settings of Section 19.4.3. Periodic retrieval occurs every $T = 600$ s. At this time instance, as shown in Figure 19.10a, the system has already reached the network storage capacity C, equal to 6000 data units. Therefore, the amount of data stored in the WSN to be delivered to the RPL root is considerable; if all nodes would send 100 data units contemporaneously, collisions would affect the amount of data successfully delivered to the sink. To investigate the data availability at the sink, that is, the amount of data effectively retrieved by the sink, data units have been injected in the network, from each node toward the RPL root, one at a time. The interval between the injection of two consecutive data units is denoted as I. Intuitively, higher injection rates may speed up the retrieval process but could

also increase the collisions' probability, thus the percentage of data loss at the sink. On the other hand, a longer transmission interval I may increase data availability, penalizing the latency. This is critical in scenarios in which the sink cannot prolong the duration of the retrieval.

Collisions are also influenced by the amount of data that is to be retrieved. With redundancy ($R > 1$), less data is to be delivered to the sink because RG only sends one replica of a data unit, for example, only the closest replica is sent. It has already been shown in Figure 19.10c that the amount of unique stored data in the system with $R > 1$ is much less than in the case with no redundancy, that is, $R = 1$. However, a comparable volume of unique data is present in the cases with $R = 3$, $R = 5$, and $R = 7$. In Figure 19.14a, the amount of retrieved data is shown, as a function of time, for the various values of replicas R. The interval I is set to 2 s in this case. It can be observed that the amount of retrieved data decreases with R because there is more unique data in the system when no redundancy is required. With $R = 1$, all 6000 data units are sent to the sink, as there is no redundancy, but only a small fraction, that is, approximately 50% of the capacity, is successfully retrieved; the rest is lost because of collisions. For higher values of R, there are more replicas of

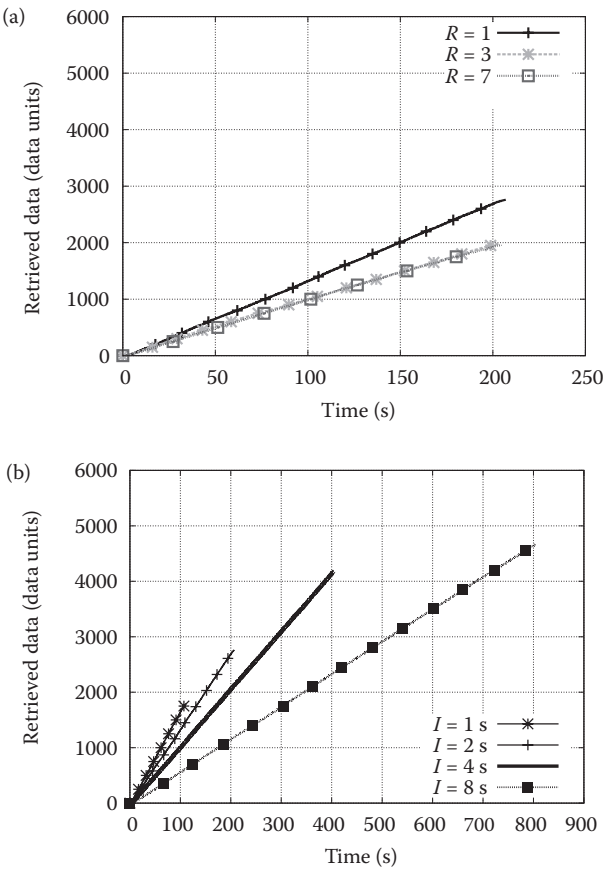

Figure 19.14 Retrieved data considering memory size equal to 100 data units, $N = 60$ storage nodes, and a memory advertisement period of 30 s. In (a), the amount of retrieved data is shown, as a function of time, for the various values of replicas R. In (b), the amount of retrieved data is shown, as a function of time, for the various values of the transmission interval I.

the same data unit in the WSN; therefore, according to the mechanism, only the closest replica is collected. However, because the amount of unique stored data is similar, the volume of retrieved data is the same in the cases with $R = 3$ and $R = 7$. The approximately 2000 retrieved data units correspond to approximately 80% of the capacity.

To reduce the effect of collisions, several simulations have been run with different values of the transmission interval I. The results depicted in Figure 19.14b for $R = 1$ show that the amount of retrieved data significantly increases for higher values of I. To summarize, the percentage of retrieved data among the total unique stored data in the system is shown in Table 19.2 for various combinations of R and I.

Table 19.2 Percentage of Retrieved Data

	I = 1 s	*I = 2 s*	*I = 4 s*	*I = 8 s*
R = 1	33%	50%	68%	82%
R = 3	49%	74%	95%	100%
R = 5	50%	74%	95%	100%
R = 7	55%	80%	100%	100%

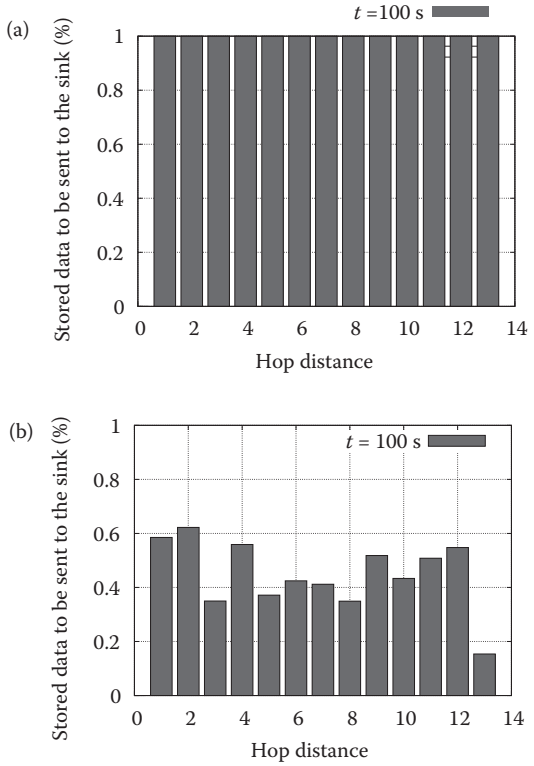

Figure 19.15 Data that is sent to the sink in the retrieval phase, for various values of *R*: (a) *R* = 1 (no redundancy) all stored data is to be sent. (b) *R* = 7 nodes closer to the sink are going to send more data than the others. Data retrieval period is set to 100 s.

Finally, Figure 19.15 shows the average percentage of data sent to the sink from several hop distances from the RPL root. In this case, the data retrieval period is set to $T = 100$ s, similar to the beginning of the simulation. Figure 19.15a corresponds to the case with $R = 1$: All nodes in the WSN send all their data to the sink. In Figure 19.15b, $R = 7$ replicas are stored in the system. In this case, nodes closer to the RPL root deliver more data than the others.

19.6 Conclusion

This chapter has addressed the problem of redundant data distribution and retrieval for IoT-based observation systems. Two redundant distributed data storage mechanisms have been proposed to increase the resilience and storage capacity of a WSN against node failure and local memory shortage. The performance of the two distributed storage mechanisms, denoted as LG and RG, has been evaluated extensively through simulations and real experiments. The RG mechanism, being built on top of RPL, lends directly to the implementation a complimentary data retrieval mechanism, whose performance has been evaluated as well. Our results clearly show a trade-off between storage redundancy (which depletes the total available storage memory) and robustness against possible node(s) failure.

Future research activities will include other parameters in the replication strategies such as the energy consumption of the nodes or the reachability of nodes, especially if they operate a low-power MAC layer with duty cycles (e.g., Contiki MAC or X-MAC). We also envision the study of dynamic reconfigurations of node behaviors (e.g., sampling) and communication layers (e.g., transmitting power and duty cycle) to meet replication demands with minimum energy cost. Finally, we would like to evaluate the performance of the RG mechanism on a real testbed such as SensLAB.

Acknowledgment

This work is funded by the European Community's Seventh Framework Programme, area "Internetconnected Objects," under grant no. 288879, CALIPSO project—Connect All IP-based Smart Objects. The work reflects only the authors' views; the European Community is not liable for any use that may be made of the information contained herein.

References

1. G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN'06)*, pages 374–381, Nashville, TN, USA, 2006.
2. P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107, San Jose, CA, USA, October 2002.
3. A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 68–77, Atlanta, GA, USA, September 2002.

4. J. Neumann, C. Reinke, N. Hoeller, and V. Linnemann. Adaptive quality-aware replication in wireless sensor networks. In *International Workshop on Wireless Ad Hoc, Mesh and Sensor Networks (WAMSNET'09)*, pages 413–420, Jeju Island, Korea, December 2009.
5. K. Piotrowski, P. Langendoerfer, and S. Peter. TinyDSM: A highly reliable cooperative data storage for wireless sensor networks. In *International Symposium on Collaborative Technologies and Systems (CTS'09)*, pages 225–232, Baltimore, MD, USA, May 2009.
6. P. Gonizzi, G. Ferrari, V. Gay, and J. Leguay. Redundant distributed data storage: Experimentation with the SensLab testbed. In *Proceedings of the 1st International Conference On Sensor Networks (SENSORNETS)*, pages 15–23, Rome, Italy, February 2012.
7. SensLAB. A very large scale open wireless sensor network testbed, 2008. Web site: http://www.senslab.info/.
8. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Proceedings of the 31st IEEE Conference on Local Computer Networks* (LCN), pages 641–648, Tampa, FL, USA, November 2006.
9. J.W. Hui and D.E. Culler. IP is dead, long live IP for wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys'08)*, pages 15–28, Raleigh, NC, USA, November 2008.
10. J.W. Hui and D.E. Culler. IPv6 in low-power wireless networks. *Proceedings of the IEEE*, Volume 98, Issue 11, pages 1865–1878, November 2010.
11. Machine to machine (M2M) standardization-ETSI, 2012.
12. H. Wang. M2M communications. In *Proceedings of the IET International Conference on Communication Technology and Application (ICCTA 2011)*, page 1009, Beijing, China, October 2011.
13. ISO/IEC 18000-6. Information technology—Radio frequency identification for item management—Part 6: Parameters for air interface communications at 860 MHz to 960 MHz. *International Organization for Standardization (ISO)*, pages 1–134, 2004.
14. ZigBee Specification. ZigBee Standards Organization Std. r17. October 2007.
15. IEEE 802.15.4 Std. Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), pages 1–679, October 2003.
16. O. Gaddour and A. Koubâa. RPL in a nutshell: A survey. *Computer Networks*, Volume 56, Issue 14, pages 3163–3178, July 2012.
17. T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550 (Proposed Standard), March 2012.
18. Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained application protocol (CoAP), draft-ietf-core-coap-11, July 2012.
19. ETSI M2M web site. http://www.etsi.org/website/technologies/m2m.aspx.
20. F. Hongping and F. Kangling. Overview of data dissemination strategy in wireless sensor networks. In *International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT)*, pages 260–263, Shenzhen, China, April 2010.
21. A. Awad, R. Germany, and F. Dressler. Data-centric cooperative storage in wireless sensor network. In *2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, pages 1–6, Bratislava, Slovak Republic, November 2009.
22. S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions On Database Systems*, Volume 30, Issue 1, pages 122–173, March 2005.
23. S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, and L. Yin. Data-centric storage in sensornets with GHT, a geographic hash table. *ACM Mobile Networks and Applications*, Volume 8 Issue 4, pages 427–442, August 2003.
24. A. Omotayo, M.A. Hammad, and K. Barker. A cost model for storing and retrieving data in wireless sensor networks. In *The IEEE 23rd International Conference on Data Engineering Workshop (ICDE)*, pages 29–38, Istanbul, Turkey, April 2007.
25. L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *The 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 1802–1810, Anchorage, AK, USA, May 2007.

26. M. Takahashi, Bin Tang, and N. Jaggi. Energy-efficient data preservation in intermittently connected sensor networks. In *The IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 590–595, Shanghai, China, April 2011.
27. Y.-C. Tseng, W.-T. Lai, C.-F. Huang, and F.-J. Wu. Using mobile mules for collecting data from an isolated wireless sensor network. In *Proceedings of the 39th International Conference on Parallel Processing (ICPP'10)*, San Diego, CA, USA, September 2010.
28. N.H. Azimi, H. Gupta, X. Hou, and J. Gao. Data preservation under spatial failures in sensor networks. In *Proceedings of the 11th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'10)*, pages 171–180, Chicago, IL, USA, September 2010.
29. G. Maia, D.L. Guidoni, A.C. Viana, A.L.L. Aquino, R.A.F. Mini, and A.A.F. Loureiro. Proflex: A probabilistic and flexible data storage protocol for heterogeneous wireless sensor networks. Research report, INRIA, July 2011.
30. TinyOS web site. http://www.tinyos.net/.
31. Contiki OS web site. http://www.contiki-os.org/.
32. A. Tsertou and D.I. Laurenson. Revisiting the hidden terminal problem in a CSMA/CA wireless network. *IEEE Transactions on Mobile Computing*, Volume 7, Issue 7, pages 817–831, July 2008.
33. M.D. Francesco, S.K. Das, and G. Anastasi. Data collection in wireless sensor networks with mobile elements: A survey. *ACM Transactions on Sensor Networks*, Volume 8, Issue 1, pages 7:1–7:31, August 2011.
34. F. Wang and J. Liu. Networked wireless sensor data collection: Issues, challenges, and approaches. *IEEE Communications Surveys Tutorials*, Volume 13, Issue 4, pages 673–687, 4th quarter 2011.
35. S. Gao and H. Zhang. Energy efficient path-constrained sink navigation in delay-guaranteed wireless sensor networks. *Journal of Networks*, Volume 5, Issue 6, pages 658–665, June 2010.
36. S. Jain, R.C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Journal of Mobile Networks and Applications*, Volume 11, Issue 3, pages 327–339, June 2006.
37. G. Anastasi, M. Conti, E. Monaldi, and A. Passarella. An adaptive data-transfer protocol for sensor networks with data mules. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Helsinki, Finland, June 2007.
38. A.A. Somasundara, A. Ramamoorthy, and M.B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium, Lisbon, Portugal*, December 2004.
39. O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, pages 1–14, Berkeley, CA, USA, November 2009.
40. N. Hassanzadeh, O. Landsiedel, F. Hermans, O. Rensfelt, and T. Voigt. Efficient mobile data collection with mobile collect. In *Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 25–32, Hangzhou, China, May 2012.
41. N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *6th International Symposium on Information Processing in Sensor Networks (IPSN 2007)*, pages 450–459, Cambridge, MA, USA, April 2007.
42. M.-E. Razvan, C.-J.M. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks, (IPSN'08)*, St. Louis, MO, USA, April 2008.