Information Fusion 22 (2015) 16–25

Contents lists available at SciVerse ScienceDirect

Information Fusion

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

Data dissemination scheme for distributed storage for IoT observation systems at large scale

Pietro Gonizzi [a,*], Gianluigi Ferrari [a], Vincent Gay [b], Jérémie Leguay [b]

[a] Wireless Ad-hoc and Sensor Networks (WASN) Laboratory, Department of Information Engineering, University of Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy
[b] Advanced Studies Department, Thales Communications & Security, 4 rue des Louvresses, 92230 Gennevilliers, France

ARTICLE INFO

Article history:
Received 18 May 2012
Received in revised form 1 March 2013
Accepted 3 April 2013
Available online 25 April 2013

Keywords:
Wireless sensor networks
Data distribution
Data replication
SensLAB
RPL

ABSTRACT

In the emerging field of the Internet of Things (IoT), Wireless Sensor Networks (WSNs) have a key role to play in sensing and collecting measures on the surrounding environment. In the deployment of large scale observation systems in remote areas, when there is not a permanent connection with the Internet, WSNs are calling for replication and distributed storage techniques that increase the amount of data stored within the WSN and reduce the probability of data loss. Unlike conventional network data storage, WSN-based distributed storage is constrained by the limited resources of the sensors. In this paper, we propose a low-complexity distributed data replication mechanism to increase the resilience of WSN-based distributed storage at large scale. In particular, we propose a simple, yet accurate, analytical modeling framework and an extensive simulation campaign, which complement experimental results on the SensLab testbed. The impact of several key parameters on the system performance is investigated.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

The concept of Internet of Things (IoT) has emerged over the last decade and relies on the vision of everyday objects having an end-to-end IP connection with the Internet. The range of objects involved in IoT encompasses several related systems, such as Radio Frequency Identification (RFID), Machine-to-Machine (M2M), and Wireless Sensor Networks (WSNs). Many business applications based on the IoT have already rolled out, notably smart grid or smart infrastructures. However, there are still many challenges to be addressed, especially regarding the specification of a standard architecture and IPv6-compliant communication protocols.

WSNs for IoT observation systems are typically composed by unattended nodes, that sense the surrounding environment, and a sink node, which acts as data collector and gateway towards the Internet. Communications between sensor nodes and the sink are typically not instantaneous, especially in isolated WSNs where the sink node is not always present. Moreover, when applications do not require real-time data collection, storing data units and sending aggregate data bursts can contribute to reduce the amount of radio transmissions, thus increasing the lifetime operation of the WSN. Efficient data retrieval can be carried out periodically through the use of a mobile node which wanders across the WSN and collects the data stored at the nodes. However, rare data retrieval can cause local memory overflow and, therefore, data loss. In order to avoid this, nodes can cooperate by storing the sensed data in a distributed way.

As WSNs tend to be left unattended for long periods, distributed data storage has to be robust also against node failures. In order to achieve this goal, an attractive approach consists in combining distributed storage with data replication, i.e., by distributing and storing multiple copies of the same data across the WSN. This redundancy exacerbates the communication overhead required to find proper "donor nodes" (i.e., nodes available to store data of other nodes), as well as the storage capacity of the system. Therefore, the design of efficient distributed storage algorithms with data replication requires to deal with trade-offs between storage capacity, system robustness, energy consumption (network lifetime), communication efficiency.

In this paper, we propose a low complexity ("greedy") distributed data replication mechanism to increase the resilience and storage capacity of a IoT-based surveillance system against node failure and local memory shortage. In [1], we have presented preliminary experimental results on distributed data storage in SensLab [2]. In the current paper, we extend this experimental approach and complement it with a simple, yet accurate, analytical performance evaluation framework, and with an extensive simulation campaign performed with the Cooja network simulator [3]. In particular, the proposed framework allows to evaluate: the network storage capacity; the time required to reach this capacity and, consequently, to start dropping data; the system robustness,

[*] Corresponding author.
E-mail addresses: pietro.gonizzi@studenti.unipr.it (P. Gonizzi), gianluigi.ferrari@unipr.it (G. Ferrari), vincent.gay@thalesgroup.com (V. Gay), jeremie.leguay@thalesgroup.com (J. Leguay).

1566-2535/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.inffus.2013.04.003

in terms of retrievable data in case of a failure of several neighboring nodes ("bomb-like scenario"). The validity of the proposed framework is confirmed by simulations and realistic experimental data obtained with SensLab, studying the impact of several key parameters. To the best of our knowledge, this is the first work that integrates experimental results of SensLab with analytical and simulative counterparts.

The paper is organized as follows. Section 2 is dedicated to related works. In Section 3, we motivate and detail the design of our greedy mechanism for distributed data replication. In Section 4, an analytical performance evaluation framework is proposed. Section 5 is dedicated to the presentation of performance results, based on the proposed analytical framework, the simulation campaign, and the real experiments conducted in the SensLab testbed. At last, Section 6 concludes the paper, with an outlook on the extension of our distributed mechanism with RPL, the standard routing protocol for the IoT [4].

2. Related work

In the past years, various schemes to efficiently distribute and replicate data in WSNs have been proposed [5].

In WSN distributed storage schemes, nodes cooperate to efficiently distribute data across the WSN. There are two main approaches: *data-centric storage* and *fully distributed data storage*. The data-centric storage approach is described in [6–8]. Here, some distinguished storage nodes, e.g., determined by a hash function, are responsible for collecting a certain type of data. A load-balanced distributed storage approach is proposed in [9], according to which data are preferably stored in densely populated areas of the sensing field to minimize data loss. In [10], data are stored according to their spatial and temporal similarity, in order to reduce the overhead as well as the latency of a query request. Even if data-centric storage approaches are based on node cooperation, they are not fully distributed since specific nodes store all the contents generated by the others. A detailed survey on data-centric storage schemes is presented in [11].

In a fully distributed data storage approach, all nodes contribute equally to sensing and storing. All nodes try, first, to store the sensor readings locally and, then, delegate other nodes in the WSN to store newly collected data as soon as their local memories are full. A first significant contribution in this direction is Data Farms [12]. 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, denoted as "donor nodes," optimizes the system capacity at the price of slightly higher transmission costs. They assume a network tree topology, where each sensor node knows the return path to a sink node, which periodically retrieves data. The energy consumption problem is studied also in [13], where data preservation in an isolated WSN is considered. In this context, an energy-efficient data distribution scheme is proposed, based on the dissemination of data from low energy nodes to high energy ones. However, only low energy nodes generate contents.

An interesting approach for load balancing is proposed in EnviroStore [14]. The authors focus on in-network data redistribution when the remaining storage space of a sensor node exceeds a given threshold. They use a proactive mechanism, where each node maintains a local memory table containing the statuses of the memories of its neighbors. Furthermore, mobile nodes (called *mules*) are used to carry data from an overloaded area to an offloaded one, as well as to send the collected data to a sink node. The deployment of mobile mules is also addressed in [15], where an efficient distributed data storage mechanism for an isolated WSN with limited storage space is proposed. Data is opportunistically offloaded to mobile mules when the latter are in proximity. Moreover, data are assigned different priorities: high prioritized data are stored closer to areas where the mules will pass more frequently. The main limitation of the above studies consists of a lack of a comprehensive performance evaluation framework, which encompasses analysis, simulations, and experiments. This holistic approach is a key contribution of our work.

Data replication strategies have been proposed in the literature, mainly to overcome the problem of node failures. The goal of replication is to copy data at other nodes within the WSN to increase resilience. Authors in [16] propose ProFlex, a distributed data storage protocol for replicating data measurements from constrained nodes to more powerful nodes. The protocol benefits from the higher communication range of such nodes and uses the long link to improve data distribution and replication against the risk of node failures. In [17], a replicator node is selected according to some critical parameters such as connectivity, available storage and remaining energy of the node. However, a model for such selection is not given. In TinyDSM [18], a reactive replication approach is presented. Replicas are randomly distributed within a predefined replication range influenced by the specific replica number and density.

Overall, with respect to related works, our paper goes beyond. First, we encompass, with a fully distributed mechanism, both data replication and distributed storage. Second, we provide a simple, yet accurate, analytical modeling framework to measure several key parameters, such as: replication robustness (by means of the percentage of data stored at a given distance from the generator node), network storage capacity, time to reach capacity, and data drop for local memory shortage. Third, we conduct extensive simulations and real experiments to evaluate our approach and validate the proposed framework.

3. Replication-based distributed data storage

We assume that the nodes of the WSN keep on collecting data (acquired with a given sensing rate). Periodically, data is retrieved from a sink and cancelled from their memories. This periodic retrieval is instrumental to allow the use of limited onboard memories. Data retrieval consists in forwarding the collected sensed data of the WSN to a central base station for further processing. In this paper, we do not focus on data retrieval, which is the subject of our current research activity.

In order to prevent data losses due to nodes' failures or memory shortages, nodes cooperate in the following way. A data acquired by a node is stored in several nodes (possibly including the generating node). This consists in copying and distributing replicas of the same data to other nodes with some available memory.

Information about memory availability is periodically broadcasted, by each node, to all its neighbors. Conversely, each node keeps on updating a local memory table relative to the memory statuses of all detected neighbors. Upon reception of a memory status from a neighbor, a node updates the corresponding entry of its local memory table with the new information received.

The proposed replication-based distributed data storage is *greedy*: in order to create a replica of a stored data, a node selects, according to its neighbors' memory table, the "best" neighbor—the selection criterion will be specified in the following. The selected neighbor becomes a *donor node*. If no donor node can be chosen and there is no available space in the local memory, then the acquired data is dropped. In the remainder of this section, we formalize this greedy algorithm.

Table 1 lists the main parameters of the system. A WSN with N fixed nodes is deployed over a region with area A (dimension: [m^2]). The radio transmission range of the nodes is denoted as d

Table 1
Main system parameters.

Symbol	Description	Unit
N	Number of nodes	scalar
A	Surface of deployment area	m^2
d	Node's transmission range	m
$V_1^{(i)}$	Number of 1-hop neighbors	scalar
B_i	Node i's buffer size, $i \in \{1,\ldots,N\}$	scalar
$T_{\text{sens},i}$	Node i's sensing interval, $i \in \{1,\ldots,N\}$	s
$r_{\text{sens},i}$	Node i's sensing rate, $i \in \{1,\ldots,N\}$	s^{-1}
P_{t}	Common node transmit power	mW
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

(dimension: [m]). The number of direct (1-hop) neighbors of node i ($i \in \{1,\ldots,N\}$), i.e., nodes within the transmission range, is denoted as $V_1^{(i)}$. The ith node has a finite local buffer of size B_i (dimension: [data units]) and sensing interval $T_{\text{sens},i}$ (dimension: [s]), whose corresponding sensing rate is $r_{\text{sens},i} = 1/T_{\text{sens},i}$ (dimension: [data units/s]). Each node broadcasts, without acknowledgment and every T_{adv} (dimension: [s]), its memory status to all 1-hop neighbors. Each memory status message contains the following values, relative to the sending node: (i) node ID; (ii) current available memory space; (iii) sensing rate; and (iv) a sequence number identifying the message. Each node maintains a local memory table which records the latest memory status received from neighbor nodes. The local memory table contains one entry per neighbor, with indication of its most recent available memory space and the corresponding notification time. 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 memory table has a fixed size. In case of complete depletion of the memory table, as it may occur in dense networks (with a large number of 1-hop neighbors), a node stores only the "best" neighbors.

The greedy distributed storage mechanism consists in creating *at most* R copies of each data unit generated by a node and distribute them across the network, storing at most one copy per node. Each copy is referred to as *replica*. Let us focus on node $i \in \{1,\ldots,N\}$. At time t, node i generates (upon sensing) a data unit. If node i has some available space in its memory, it stores a copy of the data unit locally, setting the number of remaining copies to $R-1$. Otherwise, if the local memory of node i is full, or multiple copies are to be stored, node i selects, from the memory table, a neighbor node to store a copy of the data unit. In particular, node i selects the neighbor node, called *donor*, with the largest available memory space and the most recent information. Denoting the neighbors of node i as $\{1,\ldots,V_1^{(i)}\}$, the donor at time t, indicated as $D^{(i)}(t)$, is chosen according to the following heuristic rule:

$$D^{(i)}(t) = \arg \max_{j \in \{1,\ldots,V_1^{(i)}\}} \frac{B_j(t_j)}{t - t_j}, \qquad (1)$$

where $t_j < t$ denotes the time at which the available memory space $B_j(t_j)$ of node j was received by node i, with $B_j(t_j) \leqslant B_j$. If there is no suitable neighbor in the memory table (i.e., $B_j(t_j) = 0,\ \forall j \in \{1,\ldots,V_1^{(i)}\}$), there is no possibility to distribute replicas of the data unit across the network. In this case, only the original data unit can be stored in the local memory of node i, provided that there is some available space at node i.

Upon reception of the copy, the donor node D: (*i*) stores the copy in its memory and (*ii*) selects the next donor node among its neighbors, according to a modified version of (1). In particular, in order to avoid loops, previously selected donors (which already have a copy of the received data) are not enlisted among the candidate nodes. Therefore, the selection criterion for the choice of a donor for the rth replica ($r \in \{1,\ldots,R\}$) is

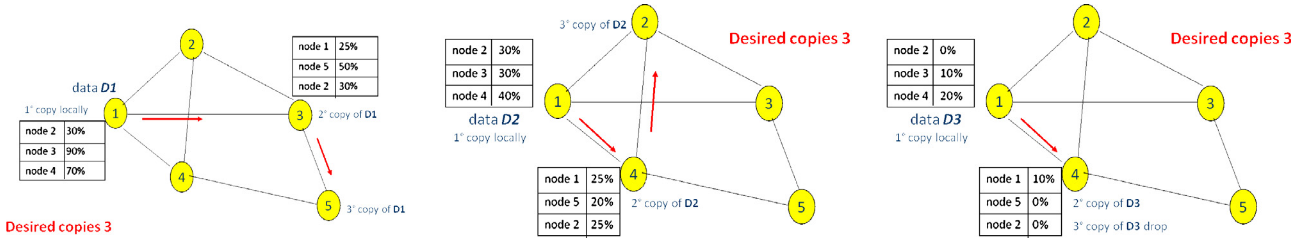
$$D_r^{(i)}(t) = \arg \max_{j \in \{1,\ldots,V_1^{(i)}\} \setminus S^{(r-1)}} \frac{B_j(t_j)}{t - t_j}, \qquad (2)$$

where $S^{(r-1)}$ is the set of donor nodes for the previous $r-1$ copies. Upon selection of the donor, the rth node thus sends it a copy, decrementing the number of required copies by 1. The replication process continues recursively until either the last (Rth) copy is stored or an intermediate (r^*th, $r^* < R$) 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.

In Fig. 1, we show three illustrative scenarios with $R = 3$. In Fig. 1a, 3 copies of the data unit $D1$, generated by node 1, are stored (respectively in nodes 1, 3, and 5). In Fig. 1b, copies of the data unit $D2$ are stored (respectively in nodes 1, 4, and 2)—note that in this case replicas do not propagate beyond 1 hop from the generator node. In Fig. 1c, the replication process stops at node 4 and no suitable donor node can further be found. In this case, the last replica is not stored.

4. Analytical performance evaluation

Implementing distributed data storage with replication raises several trade-offs. Indeed, storing multiple copies of a single data reduces the amount of unique data that can be stored in the WSN, whereas it improves the reliability against the risk of node failure and complete data loss. Also, the latter reliability goes against the lifetime operation of the WSN, as it entails more costly communication for the distribution of replicas across the WSN. In order to characterize these trade-offs, in the following subsections we derive analytical bounds for key performance metrics of interest. The portfolio of metrics includes: the *network storage capacity*, interpreted as the amount of unique data that can be stored in the WSN; the *time to reach the storage capacity*, given by the time required by the WSN to fill the memories of all nodes; the *dropped data*, i.e., the amount of data lost because of local memory shortage; and the *system robustness*, given by the percentage of recoverable distinct data in the presence of a "bomb-like"failure event



(a) 2-hop path.

(b) 1-hop path.

(c) No suitable donor

Fig. 1. Hop-by-hop replication in the case of $R = 3$ desired replicas, for several scenarios: (a) replicas propagate up to 2 hops from the source node; (b) replicas are stored at 1-hop; and (c) the replication process stops at an intermediate donor node and the last (Rth) copy is dropped.

involving all nodes within a certain spatial range. An accurate analytical evaluation (also through the derivation of upper/lower bounds) of the system robustness, as it will be defined, is an open problem.

4.1. Network storage capacity

We derive analytical expressions for upper and lower bounds of the network storage capacity, considering the cases with and without replication.

We first consider the case *without* replication, when only the original copy is kept ($R = 1$). Given the number of nodes N, the buffer sizes $\{B_i\}$, and the sensing rates $\{r_{\mathrm{sens},i}\}$, the network storage capacity C (dimension: [data units]) is simply given by

$$C = \sum_{i=1}^{N} B_i. \tag{3}$$

With 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:

$$C_\mathrm{r} \geqslant \frac{C}{R}. \tag{4}$$

Obviously, $C_\mathrm{r} = C$ when $R = 1$, i.e., only the original copy is kept and no replication is performed at all. On the other hand, the lower bound (4) can be actually reached only if R replicas of each generated data unit can be effectively stored across the WSN. This can happen only if the storage spaces $\{B_i\}$ at the nodes are very large and/or the sensing rates $\{r_{\mathrm{sens},i}\}$ are much lower than the retrieval rate $1/T$ of the sink.

4.2. Time to reach storage capacity and data drop

Another key performance metric is the time required to reach the network storage capacity. This time is of interest for an operator when parameterising the period at which the sink has to retrieve the data from the WSN. In order to provide an analytical expression for this metric, we consider the three following cases.

In the case with *local* storage and *without* replication, the time required by the ith node to fill its local buffer autonomously is $t_i =$ -$B_i/r_{\mathrm{sens},i}$. Therefore, the time interval to reach the network storage capacity corresponds to the longest storage filling time across all nodes, i.e.,

$$t_{\mathrm{cap}-l} = \max_{i \in \{1,\ldots,N\}} t_i. \tag{5}$$

Obviously, nodes which fill their buffers faster will drop newly sensed data because of local buffer overflow. Assuming that the sink retrieves the stored data when there is no available storage space left at any node in the network,[1] the total amount of dropped data can be expressed as

$$D_{\mathrm{drop}-l} = \sum_{i=1, i \neq j}^{N} (t_{\mathrm{cap}-l} - t_i) \cdot 1/T_{\mathrm{sens,i}}. \tag{6}$$

Note that the more heterogeneous the times $\{t_i\}$, the larger the amount of dropped data. On the other hand, should all filling times be equal, i.e., $t_i = t_{\mathrm{cap}-l}$, $\forall i$, it would follow that $D_{\mathrm{drop}-l} = 0$, i.e., all nodes fill up their storage memories simultaneously.

In the case with *distributed* storage *without* replication, a performance benchmark can be obtained considering an *ideal* WSN where nodes can communicate with any other node, considering instantaneous transmissions. In this case, the WSN is equivalent to a single super-node with a storage capacity C equal to $\sum_{i=1}^{N} B_i$ and sensing rate equal to $\sum_{i=1}^{N} r_{\mathrm{sens},i}$. Thus, the time required to reach the storage capacity can be given the following expression:

$$t_{\mathrm{cap-ideal}} = \frac{C}{\sum_{i=1}^{N} r_{\mathrm{sens},i}} = \frac{C}{\sum_{i=1}^{N} 1/T_{\mathrm{sens},i}}. \tag{7}$$

The amount of dropped data can then be expressed as follows:

$$D_{\mathrm{drop-ideal}} = \begin{cases} 0 & t < t_{\mathrm{cap-ideal}}, \\ (t - t_{\mathrm{cap-ideal}}) \cdot \sum_{i=1}^{N} r_i & t > t_{\mathrm{cap-ideal}}. \end{cases} \tag{8}$$

In the case with *distributed* storage *with* replication (according to the greedy mechanism proposed in Section 3), the time to reach the storage capacity can be lower-bounded using (7) after replacing C with C_r. Taking into account the lower bound (4), one thus obtains

$$t_{\mathrm{cap-d}} = \frac{C_\mathrm{r}}{\sum_{i=1}^{N} r_{\mathrm{sens},i}} \geqslant \frac{C}{R \sum_{i=1}^{N} r_{\mathrm{sens},i}} = \frac{t_{\mathrm{cap-ideal}}}{R}. \tag{9}$$

In Fig. 2, we present analytical results (using Matlab) relative to the network storage capacity. They refer to a scenario with $N = 10$ nodes, each with a buffer of dimension $B = 250$ data units and sensing rate $\{r_{\mathrm{sens},i}\}$ uniformly distributed in the interval [1,10] data units/s. Regarding the *data stored local* curve (local storage without replication), the slope decreases each time a node fills its local buffer and the time to reach the storage capacity ($C = 2500$ data units) is $t_{\mathrm{cap}-1} = 120$ s. The first dropped data occurs when the first node saturates the buffer (around 25 s). On the other hand, with ideal distribution and no replication (*data stored ideal* $R = 1$ curve), the WSN is equivalent to a single super-node: therefore, the amount of stored data increases linearly up to the storage capacity and no dropped data occurs until this point (around 40 s). Once the capacity is reached, the curve flattens and data starts to be dropped. By adding replication ($R = 3$ and $R = 5$), the total sensing rate is multiplied by R (see the denominator of the lower-bound in (9)): the capacity C is reached faster and data dropping starts earlier. In order to avoid dropped data, in practical scenarios data retrieval from the sink should be more frequent, preventing the local memories from saturating.

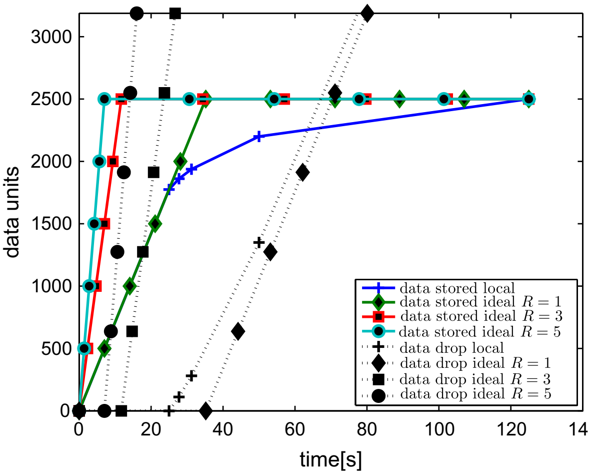


Fig. 2. Stored and dropped data, as functions of time, with local and ideal distributed storage. For both cases, curves with and without replication are depicted.

[1] This is a pessimistic assumption, as the sink might not wait till all nodes fill their buffers.

5. Performance results

In this section, we apply the analytical framework proposed in Section 4 to study the performance of our distributed storage mechanism. We complement this analysis with a simulation campaign, performed in the Cooja network simulator [3], and a large-scale experimental validation, carried out with the SensLab testbed [2]. Even if the SensLab platform is suitable for testing WSN-based applications [19], to the best of our knowledge, no studies with experimental results collected in SensLab have been published so far, but for preliminary experimental results which we presented in [1]. In the remainder of this section, we focus on a direct comparison between analytical, simulation, and experimental data, in order to clearly highlight the accuracy of the proposed framework.

In the remainder of this section, we study how the system performance is affected by (i) the (common) transmit power P_t; (ii) the sensing interval of the nodes T_{sens}; and (iii) the number of replicas R. We also investigate the system robustness against bomb-like events. Before presenting performance results, we summarize the experimental (SensLab) and simulation (Cooja) setups.

5.1. Setup

The SensLab platform offers 1024 sensors, equally distributed at 4 sites in France (Grenoble, Strasbourg, Lille, Rennes), where researchers can deploy their codes and run experiments. Each node's platform embeds a TI MSP430 micro-controller and operates in various frequency bands depending on the radio chip (either CC1100 or CC2420).

All the experimental results presented in the following are obtained from the Lille site of SensLab. The deployed WSN platform is the wsn430v14, which adopts the CC2420 radio chip, conformed to the IEEE 802.15.4 standard. An overview of the Lille site is shown in Fig. 3. Nodes are installed in a regular grid, placed on vertical and horizontal trays. We have chosen the Lille site since the deployed wsn430v14 platform embeds the same MSP430 micro-controller and the same CC2420 radio transceiver of the *Tmote sky* platform, which can be easily emulated in the Cooja simulator, thus allowing a direct comparison between simulations and experiments.

The proposed distributed storage mechanism has been implemented in Contiki, which is an open source operating system for the IoT. It allows tiny, battery-operated low-power systems to communicate with the Internet. Contiki provides two wireless networking stacks: (i) a full IP network stack (with standard IP protocols such as UDP, TCP, and HTTP), denoted as uIP and (ii) the Rime stack, a lightweight protocol stack that supports simple primitives, such as sending a message to all neighbors or to a specified neighbor. In the implementation considered for this work, we adopt the latter.

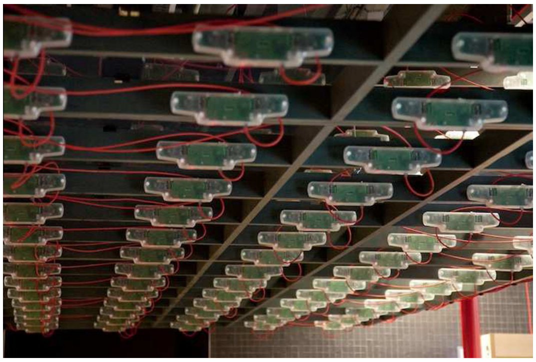


Fig. 3. The Lille site of SensLab.

At the link layer, nodes run the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism, coupled with ContikiMAC [20], a low-power asynchronous radio duty cycling protocol. All nodes have the same buffer size, equal to $B = 100$ data units. The memory advertisement period T_{adv} is set to 25 s. The scenario consists of 80 nodes,[2] placed on the same horizontal tray, i.e., at the same height. A graphical representation of the experimental setup is shown in Fig. 4a.

5.2. Impact of transmit power

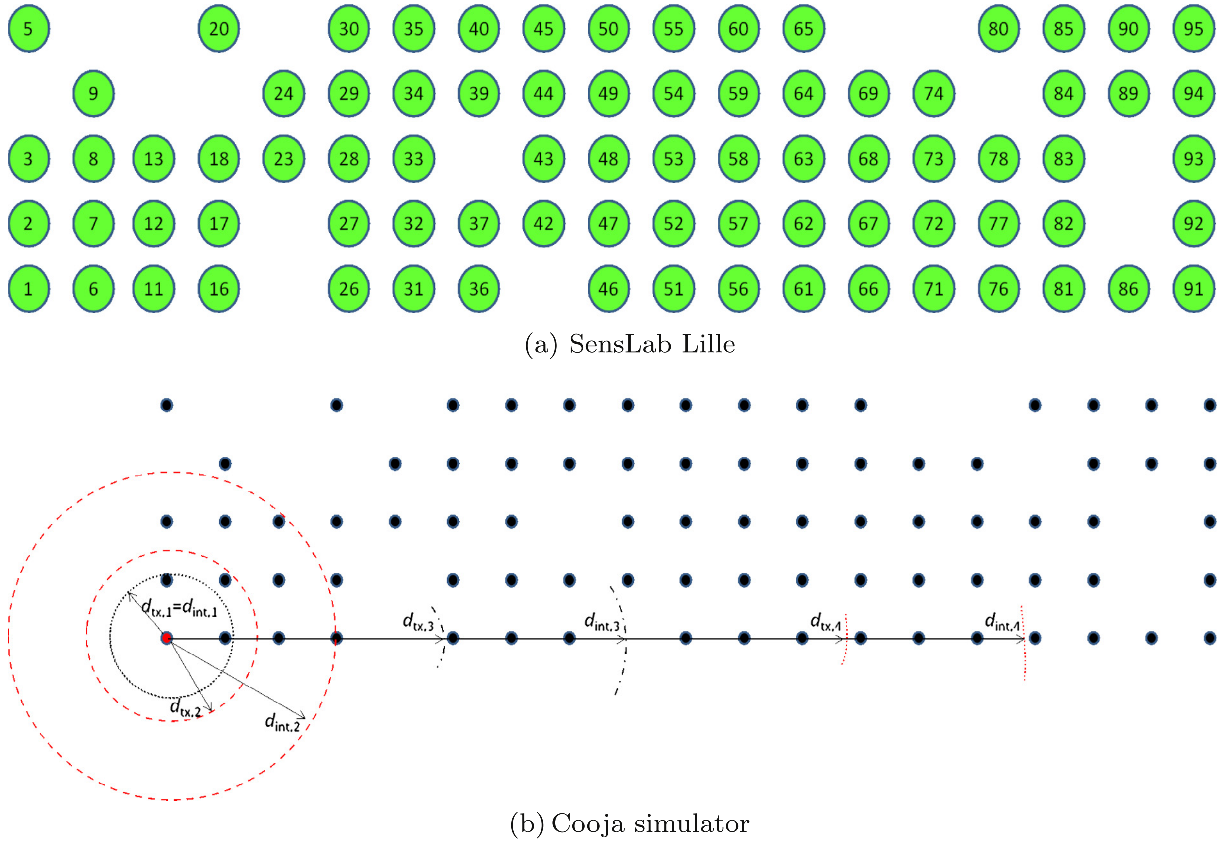
We argue that our greedy approach is significantly influenced by the network topology. For instance, in a dense network, where nodes have several neighbors, our data distribution mechanism could work better than in a sparse network with only a few direct neighbors. The network topology depends on the transmit power of the nodes. The higher the transmit power P_t, the higher the number of neighbors a node can communicate with. However, it is hard to compute the exact number of detected neighbors in SensLab. This is due to the highly variable propagation conditions of the environment, because of: reflections and shadowing effects; radio interference with experiments run by other users; presence of people in the room; instabilities of the nodes. For this reason, experiments are executed with different power settings, namely, -20 dBm and -25 dBm, according to the CC2420 datasheet. Higher values of the transmit power cause high interference, which leads to many collisions and degrades the overall communication performance.

On the simulation side, Cooja offers a unit disk communication model, composed by an inner transmission circle, with radius d_{tx}, and an outer interference circle, with radius $d_{int} \geqslant d_{tx}$. A node communicates with nodes within the circle with radius d_{tx} and interferes with nodes located within the outer circle with radius d_{int}. No interactions occur with nodes located outside the interference circle. Four combinations of d_{tx} and d_{int}, numbered from 1 to 4, have been considered in the simulations, as shown in Fig. 4b.

At this point, it is of interest to evaluate the time required by the system to reach the network storage capacity. In Fig. 5, the amount of stored data is shown, as a function of time, by comparing the analytical prediction with experimental and simulation results.

- Considering the experimental results, it can be observed that the capacity, equal to 8000 data units, is reached later when a lower transmit power is used—for instance, −25 dBm—since fewer neighbors are detected. Consequently, data cannot be distributed efficiently through the network. On the other hand, with a higher transmit power—namely, −20 dBm—each node has a "larger" neighborhood and the capacity is reached earlier.
- The analytical framework is applied considering the case with local storage ("local storage (anal)" curve), i.e., where nodes fill their own local buffers autonomously, according to (5) in Section 4.2. In this case, the time to reach 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. Note how the "local storage (exp)" curve, obtained by setting the transmit power to 0 in SensLab, is equivalent to the analytical one.
- Considering the simulation results, it can be observed that increasing the communication range d_{tx} leads to a delayed data storage. In this case, the CSMA module detects the radio channel

[2] It has been found that some SensLab nodes do not work properly or are not available for testing. This prevents us to deploy all the 256 nodes in Lille.



(a) SensLab Lille

(b) Cooja simulator

Fig. 4. Network topology: (a) scenario tested in SensLab Lille and (b) equivalent topology simulated in Cooja, for various transmission ranges.

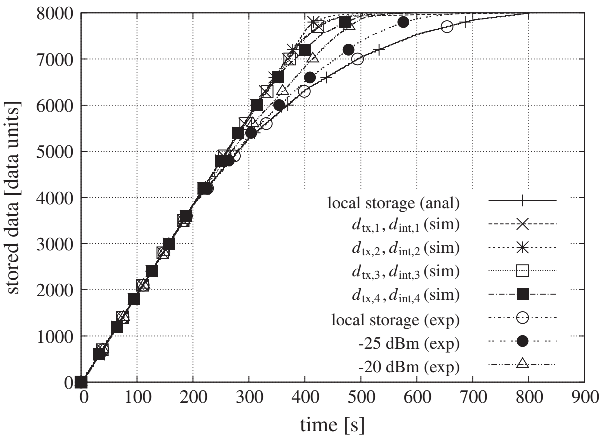


Fig. 5. Data stored in the system, for various values of the transmit power. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{\text{adv}} = 25$ s.

busy, being occupied by radio activities from other nodes, and postpones the transmission. Consequently, the time to reach the storage capacity increases.

Overall, an excellent agreement between analysis, simulations, and experiments can be observed.

We also evaluate the amount of dropped data due to local memory shortage. Results are shown, considering all configurations of Fig. 5, in Fig. 6. Nodes drop newly acquired data once their local memories are full and no neighbor is available for donating extra storage space. It can be observed that data dropping starts early for lower transmission ranges, e.g., with [$d_{tx,1}$, $d_{int,1}$] and [$d_{tx,2}$, $d_{int,2}$] configurations, respectively. In these cases, data is distributed more rapidly and the capacity is reached earlier. In the same figure, the performance predicted by our analytical framework with local storage, according to (6), and the experimental performance ("local storage (exp)" curve, −20 dBm and −25 dBm cases) are shown. As observed for Fig. 5, in this case an excellent agreement between analysis, simulations, and experiments can be observed as well.

In order to monitor the dynamic filling of the memories, in Fig. 7 the number of saturated nodes, i.e., nodes which have their local memories completely filled, is shown as a function of time. Again, it can be observed that the scenarios with lower transmit power are associated with a faster saturation of the nodes' memories. The step function is related to the case with local storage (both analytical and experimental results).

It has been anticipated that the CSMA module delays the transmission of radio packets when detecting a busy radio channel. In this case, packets are queued in the output MAC buffer up to successful delivering. The transmission fails if no acknowledgment is received back from the destination or if the queue is full.[3] In Fig. 8, the number of failed transmissions is shown as a function of time. In this case, only experimental and simulation results are shown. As expected from the results previously shown, failed transmissions occur more frequently in the presence of a higher transmit power. However, higher congestion can be observed in the experiments, even if they are run at a very low transmit power. The same behavior has been observed for higher values of the transmit power. We remark that a comprehensive performance evaluation of the

[3] In network theory, this situation is referred to as congestion, and it occurs when the arrival rate is greater than the departure rate.

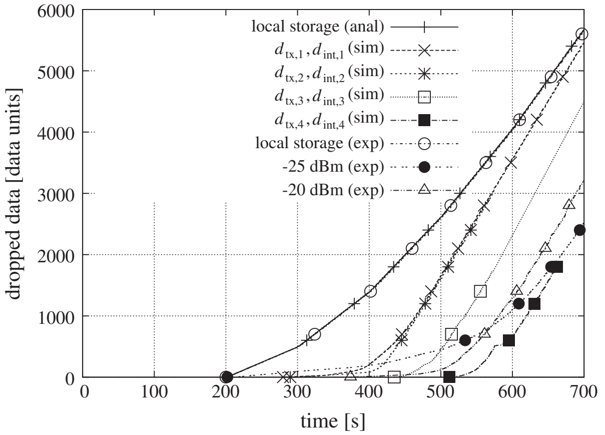


Fig. 6. Data dropped in the system, for various values of the transmit power. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{adv} = 25$ s.

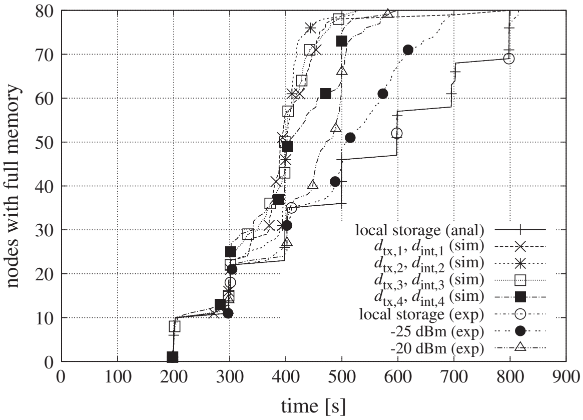


Fig. 7. Number of nodes which fill the local memories, as a function of time, for various values of the transmit power. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{adv} = 25$ s.

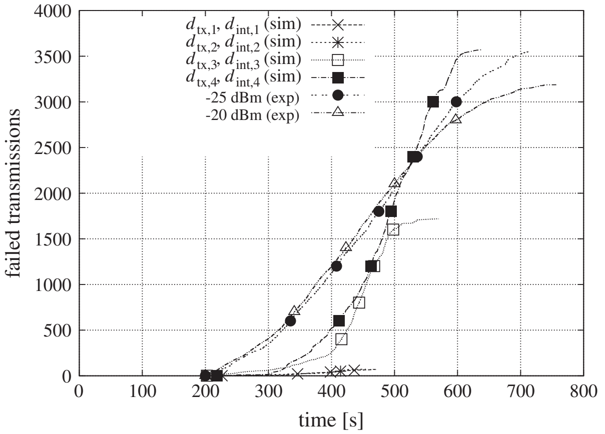


Fig. 8. Number of failed transmissions due to no reception of an acknowledgment from the receiver or to saturation of the CSMA output queue, for various values of the transmit power. Experimental and simulation results are considered. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{adv} = 25$ s.

SensLab testbed goes beyond the scope of this paper and is currently under investigation.

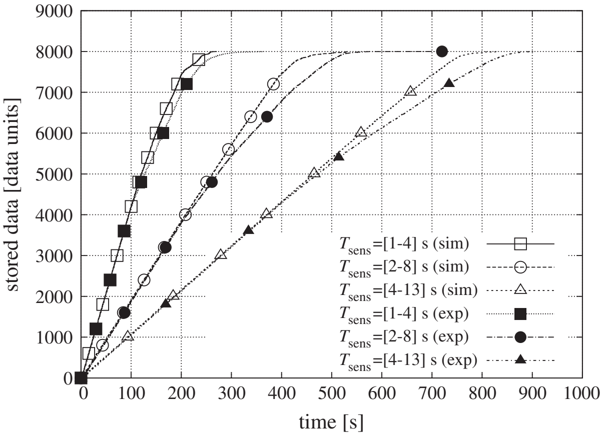


Fig. 9. Data stored in the system, for various values of the sensing interval T_{sens}. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes.

5.3. Impact of the sensing interval

In this subsection, we show the results obtained varying the sensing interval T_{sens} of the nodes. Recall from Section 3 that the sensing interval, defined as $T_{sens} = 1/r_{sens}$, is the time interval between the generation of two consecutive sensed data units by a node. We run various experiments and simulations setting the sensing interval T_{sens} to an integer value in the possible ranges: [1–4] s, [2–8] s and [4–13] s, respectively. We deploy the same scenario as in Section 5.2, with no replication ($R = 1$). The transmit power P_t of the SensLab nodes is set to -20 dBm. The unit disk model in Cooja is set to $[d_{tx,3}, d_{int,3}]$. In Fig. 9, the amount of stored data is shown, as a function of time, in correspondence to the ranges of values of T_{sens} indicated above. It can be observed that the memories of the nodes are filled faster with a shorter sensing interval. In addition, the agreement between simulations and experiments is very good.

In Fig. 10, the number of saturated nodes, i.e., nodes which have their local memories completely filled, is shown as a function of time. Nodes fill their local memories faster when shorter sensing intervals are used, e.g., when T_{sens} is selected in the range [1–4] s. In the simulated cases, the saturation of the memories occurs earlier, since the impact of the CSMA backoff delay is reduced as compared to experiments.

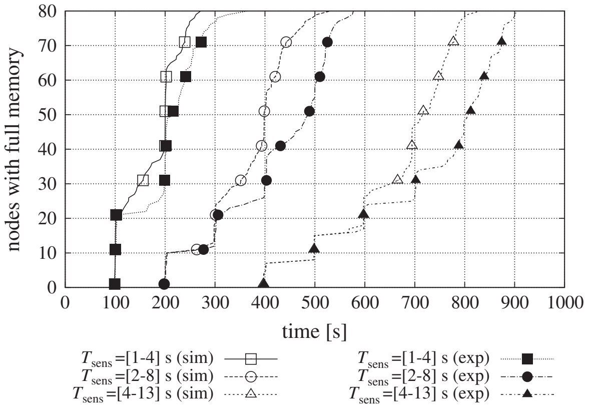


Fig. 10. Number of nodes which fill the local memories, as a function of time, for various values of the sensing interval T_{sens}. For each value of T_{sens}, both experimental and simulated results are shown. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes.

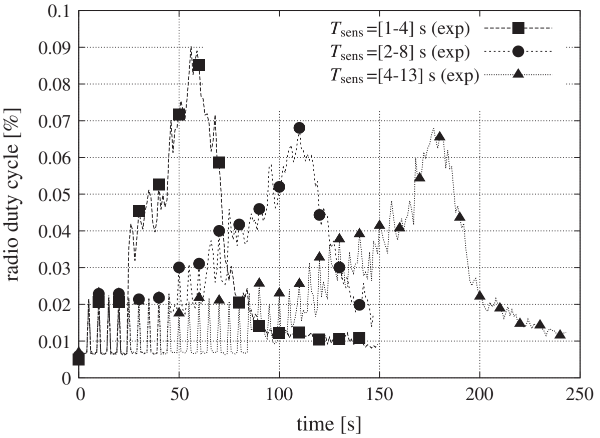


Fig. 11. Average radio duty cycle (percent) over time, for various values of the sensing interval T_{sens}. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{\text{adv}} = 25$s.

In order to track the energy consumption of the nodes in SensLab, we evaluate the average percentage of time the radio of the nodes is turned on. For this purpose, we use Powertrace, a system for network-level power profiling for low-power wireless networks [21]. The obtained results are shown in Fig. 11. As expected, in the case with a short sensing interval, i.e., $T_{\text{sens}} \in [1–4]$ s, the power consumption increases, since nodes tend to distribute data more frequently. In the other two cases, i.e., $T_{\text{sens}} \in [2–8]$ s and $T_{\text{sens}} \in [4–13]$ s, respectively, the energy consumption is lower.

Overall, it can be concluded that results obtained through Cooja simulations and through real experiments in SensLab are in agreement with each other. As mentioned at the end of Section 5.2, a thorough characterization of the SensLab platform is an interesting research topic.

5.4. System robustness

There is often strong spatial correlation among failed nodes. The events that destroy one node may very likely influence a nearby node and destroy it as well. Examples could be a natural disaster (earthquake, fire, etc.) or system crashes (application failure, network errors, external attack).

First of all, robustness against nodes' failure is directly related to how well copies of a given data can spread across the network. As discussed in Section 3, redundancy 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, avoiding loops. For the following tests, the transmit power P_t in SensLab has been increased to -10 dBm, and the unit disk in Cooja has been set to $[d_{\text{tx},3}, d_{\text{int},3}]$.

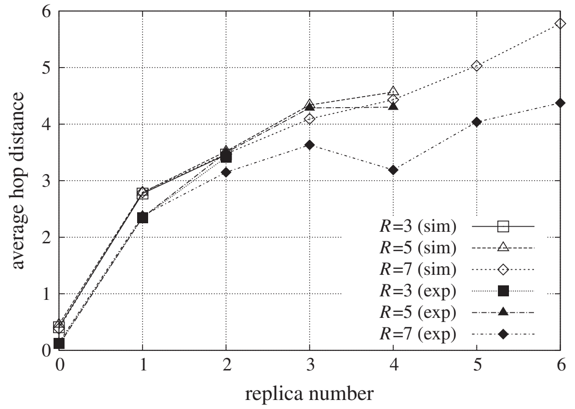


Fig. 12. Average hop distance reached by the kth replica versus k. k varies between 0 and 2 ($R = 3$), 0 and 4 ($R = 5$), 0 and 6 ($R = 7$), respectively. The 0th replica refers to the original data.

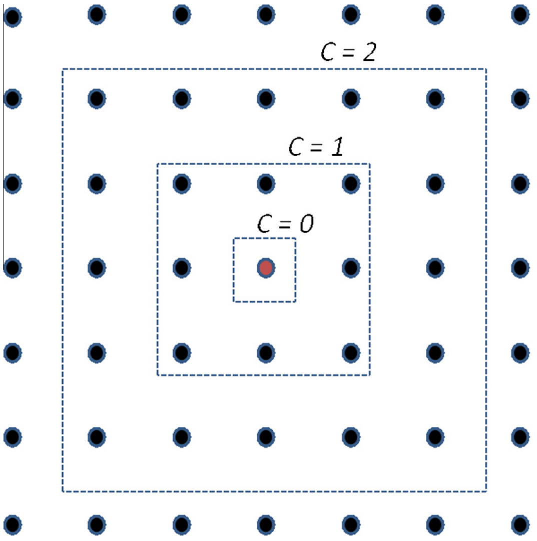


Fig. 13. Node failure in the case of a "bomb-like" failure event, involving the node and several neighboring nodes within a certain spatial range. The c parameter controls the impact of the bomb.

In Fig. 12, the average hop distance (from the generator node) reached by replicas is shown as a function of the replica number—the 0th replica refers to the original data. Various values of R are considered: for each value, the replica number varies between 0 and $R - 1$. It can be observed that, on average, consecutive replicas tend to spread reasonably through the network. Simulated and experimental curves almost coincide for $R = 3$ and $R = 5$. This is in agreement with the proposed mechanism, since donor nodes are selected on the basis of their available memory space and sensing rate, which are the same for simulations and experiments. In the case with $R = 7$, there is a discrepancy between experimental and simulation results for high values of the replica number. This is due to the fact that, according to the results in Fig. 10, in the realistic SensLab scenario nodes saturate their memories more slowly. Therefore, copies are likely to be stored closer to the originator than in the simulation scenario. As already observed, however, the agreement between simulations and experiments is very good. In order to make the agreement excellent, the Cooja simulator needs to be extended to capture the phenomena (mostly at physical layer) that affects the communication performance in SensLab.

At this point, in order to investigate further the robustness of the proposed distributed storage mechanism, we assume a "bomb-like" failure event, involving a node and all its direct neighbors within a certain spatial range. We assume a squared failure range, with edge c, as depicted in Fig. 13. Intuitively, higher values of c correspond to a more violent bomb-like event. The system robustness is defined as the percentage of distinct data units (i.e., not counting replicas) which can still be retrieved after a bomb-like event. According to this definition, assuming that the bomb-like event happens at time t, the system robustness $Rob_{\text{sys},c}(t)$ can be expressed as

$$Rob_{\text{sys},c}(t) = \frac{\sum_{l=1}^{X(t)} Rob_{\ell,c}(t)}{X(t)}, \tag{10}$$

where $X(t) \leqslant C_r$ is the amount of distinct data stored in the system up to time t; and $Rob_{\ell,c}(t)$ equals 1 if at least a replica of the lth data

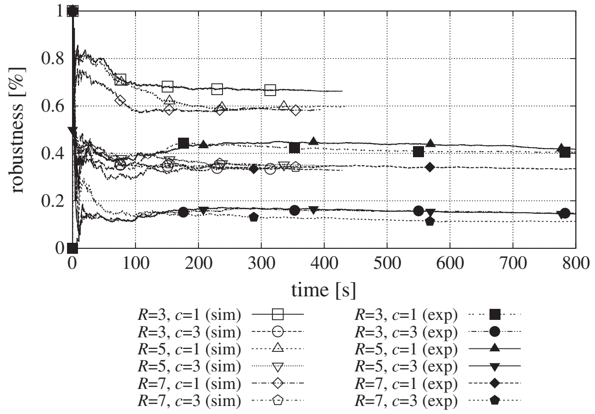


Fig. 14. System robustness, as a function of time, considering various values of *R* and *c*. Both simulated and experimental results are shown.

unit has been stored, at time *t*, outside the *c*th square (otherwise, $Rob_{\ell,c}(t) = 0$). Note that the time instant *t* at which a bomb-like event happens has an impact on the system robustness—obviously, our definition of system robustness depends implicitly on *t*. At the beginning of the collection period (i.e., with limited storage), it is more likely that at least one of the *R* replicas is stored in a surviving node (in a node beyond the direct neighbors). On the other hand, at the end of the collection period the network saturates and nodes' buffers are almost full. Thus, it is more likely that no replica can be stored and, therefore, a bomb-like event may destroy a significant amount of information.

In Fig. 14, the system robustness is shown as a function of time, for various configuration of *R* and *c*. For each considered combination, both simulated and experimental results are shown. Two values of *c* are considered, namely, $c = 1$ and $c = 3$. The robustness increases for lower values of *c*, since the geometrical shape of the failure event is reduced. As for the number of replicas *R*, the case with $R = 3$ surprisingly has the highest robustness. In fact, storing fewer replicas slows down the saturation of the memories, and it allows more data to be replicated. Both simulations and experiments show this trend.

Simulated results show a higher robustness than experimental ones. In particular, the robustness is around 0.7 in the simulated case with $R = 3$ and $c = 1$, while in the corresponding experimental case the robustness reaches 0.5—similar conclusions hold for the remaining configurations. This is caused by (i) the slightly higher spread of the replicas in simulations, as observed in Fig. 12 for $R = 7$, and by (ii) the amount of stored data for each replica number. The latter parameter is investigated in Fig. 15, where the Probability Mass Function (PMF) of the total stored data, as a function of the replica number, is shown, considering various values of *R*. It can be observed that, in the experimental case, the PMF concentrate at the origin, i.e., a considerable amount of data, about 60%, has only the original copy. On the opposite, the PMFs predicted by the simulations, regardless of the value of *R*, have the same trend of the experimental ones, but for a higher average value—indeed, less than 50% of the data has only the original copy. The discrepancy between experimental and simulation results is caused, as already observed, by the collisions in the SensLab testbed, which limit effective data spreading.

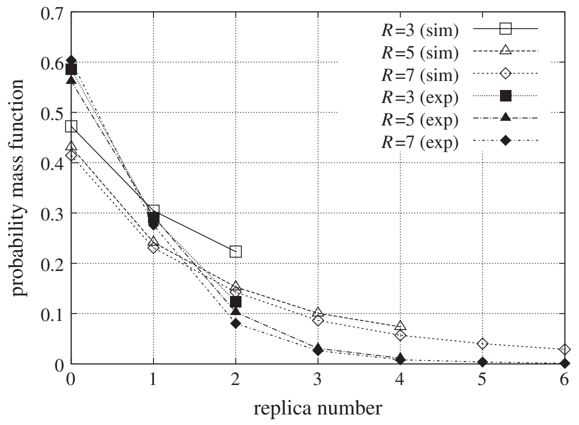


Fig. 15. Distribution function of the stored data in the system, as a function of the copy number, for various values of *R*. Both simulated and experimental results are shown.

5.5. Discussion

The impact of several parameters on the performance of the proposed distributed data storage mechanism has been investigated, through analysis, simulations, and experiments. On the basis of the obtained results, the following observations can be carried out.

Experimental results, which significantly extend the preliminary results in [1], have been validated through an analytical framework and a comprehensive simulation campaign. In particular, the Cooja network simulator has been configured to reproduce the SensLab Lille experimental scenario as accurately as possible. We have shown that a careful calibration of the communication range in the simulator allows to obtain simulation results which have a limited discrepancy with respect to the experimental results. This discrepancy is mainly related to the higher amount of collisions in the SensLab Lille testbed, which cause the CSMA module to delay the transmissions or even to drop packets. Accurate modeling of the propagation conditions, the interference, and the received radio signal strength of the SensLab nodes is an interesting research direction and is currently under investigation.

As for the proposed greedy distributed storage mechanism, it can be concluded that the lightweight hop-by-hop replication scheme guarantees a reasonable spread of the replicas. Therefore, this mechanism is robust in the case of a failure involving several nodes within a limited area. A shortcoming of the proposed approach, however, is that it tries to make exactly *R* replicas of all data. This may be unconvenient when the memories of the nodes are almost full, and may prevent new data to be stored. A solution to this problem could rely on a dynamic self-organization of the replicas, which autonomously decide to replicate or not. For example, the replication process may stop in correspondence of an intermediate *r*th step (with $r < R$) if a prior replica has already been stored far away. In order to prevent dropping of locally generated data, thus increasing robustness, the storage space at each node could also be divided into two blocks: one for local measurements and the other for data coming from other nodes.

Another appealing extension of the proposed mechanism is in the direction of including load balancing, e.g., by equalizing the levels of occupancy of the nodes' memories over time. Replicas may move from a saturated network region to an offloaded one. However, the complexity of this solution could be quite high and may require the use of a centralized approach.

6. Conclusion and future work

This paper has addressed the problem of redundant data distribution for IoT-based observation systems. We have proposed a low-complexity greedy mechanism to distribute and replicate measurements with a minimum signaling overhead. Through a rich analytical framework, extensive (Cooja) simulations, and a large experimental campaign on the SensLab Lille testbed, we have shown how the performance is affected by the main system parameters. Results show a satisfactory agreement between simulations, experiments, and analysis. To the best of our knowledge, this is one of the first works presenting experimental results at large scale on SensLab, supported by simulations and an analytical framework.

Extensions to the proposed mechanism will consider RPL, the IPv6 routing protocol for Low power and Lossy Networks (LLNs) standardized by IETF ROLL [4], which is one of the building blocks of the IoT. We believe that our distributed data storage scheme can be significantly improved with RPL, leading to (i) a more efficient donor node selection algorithm and (ii) a broad propagation of redundant copies through the network. We will also investigate the data retrieval phase using the Directed Acyclic Graph (DAG) structure provided by RPL.

Acknowledgments

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. The authors thank the anonymous reviewers, whose comments significantly improved the quality of this work.

References

[1] P. Gonizzi, G. Ferrari, V. Gay, J. Leguay, Redundant distributed data storage: experimentation with the SensLab testbed, in: Proceedings of the 1st International Conference on Sensor Networks (SENSORNETS), Rome, Italy, 2012.

[2] SensLab: Very Large Scale Open Wireless Sensor Network Testbed. <http://www.senslab.info/>.

[3] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with cooja, in: Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN), 2006, pp. 641–648.

[4] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, Rpl: Ipv6 routing protocol for low-power and lossy networks, in: Internet-draft, IETF ROLL Working Group (March 2012). <http://www.rfc-editor.org/rfc/rfc6550.txt>.

[5] F. Hongping, F. Kangling, Overview of data dissemination strategy in wireless sensor networks, in: International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT), Shenzhen, China, 2010, pp. 260–263.

[6] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, F. Yu, Data-centric storage in sensornets with ght, a geographic hash table, Mobile Networks and Applications 8 (4) (2003) 427–442.

[7] S.R. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, Tinydb: an acquisitional query processing system for sensor networks, ACM Transaction Database Systems 30 (1) (2005) 122–173.

[8] A. Awad, R. Germany, F. Dressler, Data-centric cooperative storage in wireless sensor network, in: 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL), Bratislava, Slovak Republic, 2009, pp. 1–6.

[9] M. Albano, S. Chessa, F. Nidito, S. Pelagatti, Dealing with nonuniformity in data centric storage for wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems 22 (8) (2011) 1398–1406, http://dx.doi.org/10.1109/TPDS.2011.18.

[10] H. Shen, L. Zhao, Z. Li, A distributed spatial–temporal similarity data storage scheme in wireless sensor networks, IEEE Transactions on Mobile Computing 10 (7) (2011) 982–996, http://dx.doi.org/10.1109/TMC.2010.214.

[11] K. Ahmed, M.A. Gregory, Techniques and challenges of data centric storage scheme in wireless sensor network, Journal of Sensor and Actuator Networks 1 (1) (2012) 59–85.

[12] A. Omotayo, M. Hammad, K. Barker, A cost model for storing and retrieving data in wireless sensor networks, in: IEEE 23rd International Conference on Data Engineering Workshop (ICDE), Istanbul, Turkey, 2007, pp. 29–38.

[13] M. Takahashi, B. Tang, N. Jaggi, Energy-efficient data preservation in intermittently connected sensor networks, in: IEEE 30th Conference on Computer Communications Workshops (IEEE INFOCOM 2011), 2011, pp. 590–595.

[14] L. Luo, C. Huang, T. Abdelzaher, J. Stankovic, Envirostore: a cooperative storage system for disconnected operation in sensor networks, in: 26th IEEE International Conference on Computer Communications (INFOCOM'07), Anchorage, Alaska, USA, 2007, pp. 1802–1810.

[15] Y.-C. Tseng, F.-J. Wu, W.-T. Lai, Opportunistic data collection for disconnected wireless sensor networks by mobile mules, Ad Hoc Networks, http://dx.doi.org/10.1016/j.adhoc.2013.01.001.

[16] G. Maia, D.L. Guidoni, A.C. Viana, A.L. Aquino, R.A. Mini, A.A. Loureiro, A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks, Ad Hoc Networks, http://dx.doi.org/10.1016/j.adhoc.2013.01.004.

[17] J. Neumann, N. Hoeller, C. Reinke, V. Linnemann, Redundancy infrastructure for service-oriented wireless sensor networks, in: 9th IEEE International Symposium on Network Computing and Applications (NCA'10), Cambridge, MA, USA, 2010.

[18] K. Piotrowski, P. Langendoerfer, S. Peter, tinyDSM: A highly reliable cooperative data storage for wireless sensor networks, in: International Symposium on Collaborative Technologies and Systems (CTS'09), Baltimore, Maryland, USA, 2009, pp. 225–232.

[19] T. Ducrocq, J. Vandaele, N. Mitton, D. Simplot-Ryl, Large scale geolocalization and routing experimentation with the senslab testbed, in: IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS'10), San Francisco, CA, USA, 2010, pp. 751–753.

[20] A. Dunkels, The Contikimac Radio Duty Cycling Protocol, Tech. Rep., Swedish Institute of Computer Science, December 2011.

[21] A. Dunkels, J. Eriksson, N. Finne, N. Tsiftes, Powertrace: Network-level Power Profiling for Low-power Wireless Networks, Tech. Rep., Swedish Institute of Computer Science, March 2011.