Data Storage and Retrieval with RPL Routing

Pietro Gonizzi, Gianluigi Ferrari
Department of Information Engineering
University of Parma
Parco Area delle Scienze 181/A
Parma, Italy
Email: pietro.gonizzi@studenti.unipr.it, gianluigi.ferrari@unipr.it

Paolo Medagliani, Jérémie Leguay
Thales Communications and Security
160 Bd de Valmy
Colombes Cedex, France
Email: {paolo.medagliani, jeremie.leguay}@thalesgroup.com

***Abstract*—In scenarios like the surveillance of isolated areas, when the border node of a network does not have a permanent connection with the Internet, Wireless Sensor Networks (WSNs) are calling for resilient in-network data storage techniques which minimize the risk of data loss. The efficiency of these techniques can be largely improved exploiting information on the status of the network, such as that used by routing protocols. In particular, one of the most used protocol in Internet of Things (IoT) scenarios is the IPv6 Routing Protocol for Low power and lossy networks (RPL). In this paper, we propose a redundant distributed data storage and retrieval mechanism to increase the resilience and storage capacity of a RPL-based WSN against local memory shortage. We evaluate our approach in the Contiki operating system through extensive analysis with the Cooja simulator.**

***Index Terms*—distributed data storage; RPL; data replication; data retrieval; Contiki; Cooja.**

I. INTRODUCTION

In contrast to 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. Recently, sensors have reached higher capabilities, in terms of processing speed and local storage, than in the past years [1], making them more attractive for in-network storage deployments.

Typically WSNs are composed of a set of unattended nodes, deployed to sense the surrounding environment, and a sink node in charge of collecting data measurements and relaying them to a management entity. There are several reasons which may prevent a sensor node from transmitting data to the sink right after acquisition. For instance, sensor nodes may not always be able to reach the sink node due to intermitting link conditions or duty-cycle operations at the nodes. In addition, when applications do not require real-time collection, storing data units and sending aggregate data bursts can contribute to reduce the amount 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 to collect and store 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 timely performed 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 by means of data replication (i.e., by storing copies of the same data onto various nodes) contributes to increasing the resilience of the WSN.

In this study, we extend the work presented in [4], where only distributed storage (and no data retrieval) is considered. We propose a redundant distributed data storage and retrieval mechanism to increase the resilience and storage capacity of a RPL-based WSN against local memory shortage. RPL, an IETF standard for IPv6 routing in low-power WSNs, is a Distance Vector routing protocol that builds a Destination Oriented Directed Acyclic Graph (DODAG) rooted at one sink (DAG root) [5]. We evaluate our approach in the Contiki operating system using the Cooja [6] simulator. In particular, we show how RPL can be used for robust and energy-efficient distributed data storage and retrieval.

The paper is organized as follows. Section II is dedicated to related works. An overview of RPL is presented in Section III. Section IV is devoted to the design of the data distribution and retrieval mechanisms. Section V presents the performance results. Finally, Section VI concludes the paper.

II. RELATED WORK

Various schemes to efficiently store and process sensed data in WSNs have been proposed in the past years [7]. In a *fully distributed data storage* approach, all nodes participate in sensing and storing in the same way. All nodes, first, store their sensor readings locally and, once their local memories have filled up, they delegate storage to other available nodes. A first significant contribution in this direction is given by Data Farms [8]. The authors propose a fully distributed data storage mechanism with periodical 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 capacity at the price of slightly higher transmission costs. They assume the network has a tree topology and each sensor node knows the return path to the sink node, which periodically retrieves data.

978-1-4673-2480-9/13/$31.00

Data replication consists in adding redundancy to the system by copying data at several donor nodes (within the WSN) to mitigate the risk of node failure. A scoring function for suitably choosing a replicator node is proposed in [9]. 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. Authors in [10] propose ProFlex, a distributed data storage protocol for replicating data measurements from constrained nodes to more powerful nodes.

Data retrieval consists in forwarding the collected sensed data of the WSN to a central base station for further processing. The Collection Tree Protocol (CTP) is probably the routing mechanism most frequently used for multi-hop fixed data retrieval in WSNs [11]. The strengths of CTP are (i) its ability to quickly discover and repair path inconsistencies and (ii) its adaptive beaconing, which reduces protocol overhead and allows the use of small radio duty cycles. Dozer [12] is a data retrieval protocol aiming at achieving very low energy consumption. It builds a tree structure to convey data to the sink, enriched with a Time Division Multiple Access (TDMA) scheme at the MAC layer to synchronize the nodes.

With respect to related works, our study goes beyond. First, we encompass, with a fully distributed mechanism, both data replication and distributed storage. Second, we show how RPL can allow the design of a resilient data placement as well as an efficient data retrieval scheme. To the best of our knowledge, this is the first work addressing data storage and retrieval mechanisms on top of RPL.

III. RPL OVERVIEW

RPL [5] has recently emerged as the standard for routing in low-power IPv6 WSNs. It is based on a DODAG anchored at one or more nodes (DAG root(s)). Each node computes its rank in the RPL tree. This quantity describes the depth of the node in the DODAG. To build and maintain the topology, RPL nodes periodically exchange DODAG Information Object (DIO) messages, in order to propagate routing information downward in the tree. This kind of structure is particularly suitable for multipoint-to-point traffic, where the DAG root is the destination of all data packets.

In support of point-to-multipoint and point-to-point traffic, RPL defines an additional control message, denoted as Destination Advertisement Object (DAO) message, used to populate the routing tables of parent nodes in the DAG (i.e., nodes with lower rank), in order to route packets in the down direction. Routes are computed according to an Objective Function (OF) and a given set of metrics and constraints of interest.

IV. REDUNDANT DATA STORAGE AND RETRIEVAL

In our scenario, nodes of the WSN, upon joining a RPL DODAG, keep on collecting data (acquired with a given sensing rate). In order to prevent data losses, data is replicated 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 space. Information about memory availability is periodically broadcasted, by each node, to all direct neighbors.

Symbol	Description	Unit
N	Number of RPL nodes	scalar
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}
mhd_{down}	minimum hop distance (in the down direction) at which a node with some available space can be found	scalar
mhd_{up}	minimum hop distance (in the up direction) at which a node with some available space can be found	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 DODAG root)	s

TABLE I
MAIN SYSTEM PARAMETERS

Data retrieval is performed by an external agent that periodically connects to the DAG root and gathers all the data from the WSN. The main parameters are listed in Table I. Without loss of generality, we consider a WSN with N fixed RPL nodes and an additional node, who acts as DAG root but does not participate in sensing and storage. Therefore, the overall number of nodes in the WSN is $N+1$. The i-th 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., 1-hop neighbors). Each memory advertisement consists of 6 fields relative to the sending node: (i)the RPL rank of the node; (ii) value of sensing rate; (iii) updated available memory space; (iv) an aggregate that indicates the status of nodes' memories in the down direction in the DODAG; (v) an analogous value for the up direction; and (vi) a sequence number. Each node maintains a table which 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 by the minimum hop distance (mhd_{down} parameter) 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. Similarly, the status of the nodes' memories in the up direction is computed in the same way, but in the inverse direction of the DODAG; in this case, it is given by the minimum hop distance upward (mhd_{up}) parameter.

An illustrative scenario is shown in Fig. 1. In Fig. 1(a), node

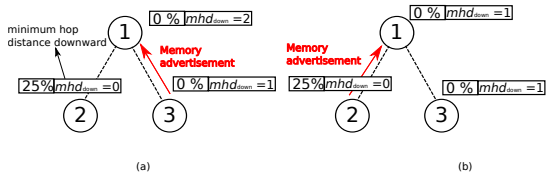


Fig. 1. Messages exchanged for memory advertisements. (a) Memory availability is announced by node 3, at 1 hop in its down direction; (b) Node 1 updates mhd_{down} to 1 as it becomes aware of a closer node with some available memory.

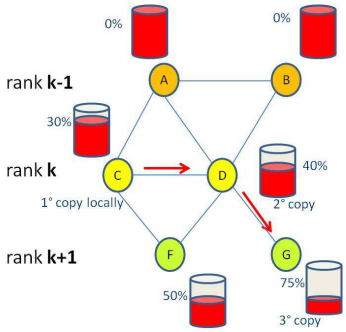


Fig. 2. Hop-by-hop replication in the case of $R = 3$ desired replicas. Data is distributed to nodes in the down direction of the RPL DODAG as the memories of the parents are full.

3 transfers a memory advertisement, to node 1, saying that it has no available space but there is one of its 1-hop children with available space, as stated by mhd_{down}. Node 1, which has no available memory, then sets mhd_{down} to 2, because it has received the information that the closest node with available memory is at 2-hop distance. Then, as shown in Fig. 1(b), node announces 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.

The proposed distributed storage mechanism is fully decentralized, in the sense that all nodes play the same role. It consists in creating at most R copies of each data unit generated by a node and distributing them across the network, storing at most one copy per node. The copies should be stored as closely as possible to the DAG root to reduce the energy consumption of the following retrieval phase. Each copy is referred to as *replica*.

Consider 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 *donor*, with the largest available memory space and the most recent information. Moreover, priority is given to those donors which are parents of node i in the tree, i.e., nodes with lower rank. If no parents can be selected, node i looks for a child in the tree, i.e., a node with higher RPL rank, providing that such node has some available space. If all neighbors have no space locally, then node i checks if one neighbor at least has a neighbor (at 2 hops from node i) with some available space, in the up direction and/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, provided that i has some space locally.

If a donor node can be selected, node i sends to it a copy of the data unit, specifying how many other copies are still to be distributed in the WSN. According to the principle outlined above, 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 with the same rule, prioritizing nodes closer to the DAG root. This allows replicas to spread well throughout the DODAG. At this point, the donor sends the copy to the next chosen donor node, decreasing the number of required copies by 1. The replication process continues recursively until either the last (R-th) 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 lower RPL rank (i.e., closer to the DAG root), the replica may follow a different reversed path along the DODAG, and retake the original direction later.

In Fig. 2, an illustrative example with $R = 3$ desired replicas is shown. Nodes always try to distribute replicas to parents in the up direction of the DODAG, e.g., nodes with a lower rank. As the network saturates, data is distributed towards nodes with higher rank, i.e., in the down direction of the DODAG.

As for data retrieval, retrieval requests are sent periodically by the DAG root and propagated to all nodes of the DODAG. The data retrieval takes place, upon reception of the request, from each node towards the DAG root, following the default RPL path towards the root. Such *many-to-one* traffic pattern, if not carefully handled, can cause (i) many collisions and (ii) high unbalanced and inefficient energy consumption in the whole network. To reduce these risks, a replica of a given data item is sent only if it is the closest to the DAG root, amongst all the stored replicas of that data item.

V. PERFORMANCE RESULTS

The proposed distributed storage and retrieval mechanisms have been implemented in Contiki 2.5 and evaluated in Cooja, a Java-based WSN simulator [6]. The simulated scenario, depicted in Fig. 3, is composed of $N = 60$ storing nodes, placed in a rectangular grid, and an additional node, who acts as DAG root. Each storing node inside the grid communicates with 4 direct neighbors. Moreover, to simulate real conditions, the node interferes with some extra nodes: a collision occurs if a node and at least one amongst its neighbors or its interfering nodes transmit a packet at the same time. For example, node 41 has 4 neighbors: nodes 34, 40, 48, and 42, respectively. The interfering nodes, shown between the two circles, are nodes:

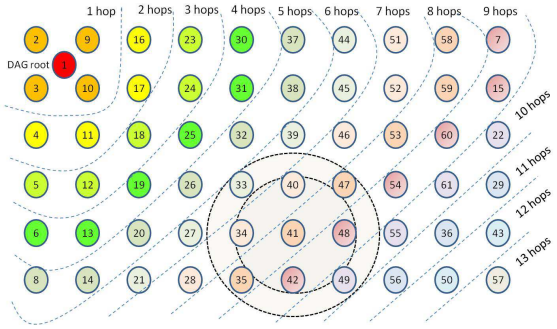


Fig. 3. Scenario evaluated in Cooja. 60 storage nodes are deployed in a regular grid. Each node has 4 direct neighbors. Node 1 is the DAG root. The network has a total size equal to 13 hops.

33, 35, 47 and 49, respectively. Note that nodes along the borders have less neighbors than the inside ones.

The interval between two consecutive data retrieval is $T = 10$ min. The sensing period of the nodes is an integer number chosen randomly and independently in the range [1,9] s. All nodes have the same buffer size, equal to $B = 100$ data units. The memory advertisement period T_{adv} is set to 30 s. We adopt the Expected Transmission Count (ETX) as RPL metric. In particular, the ETX metric minimizes the number of expected transmissions to reach the DAG root.

Impact of the Number of Replicas: Four possible values for the number R of replicas have been considered: 1, 3, 5, and 7, respectively. Note that the case with $R = 1$ is without replication, i.e., only the original copy is stored.

Regarding the spatial distribution of replicas, Fig. 4 shows the average hop distance reached by the redundant copies, from the owner of the original one, for various values of R. Our results show that consecutive replicas tend to move farther and farther from the source node along the DODAG. While the mechanism proposed in [4] was inefficient in terms of storage spreading (in fact, the average hop distance saturated around 2), the current mechanism, thanks to RPL, can lead to a more resilient data preservation in the presence of a failure of an area involving several neighboring nodes.

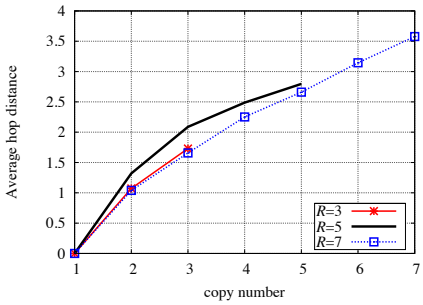
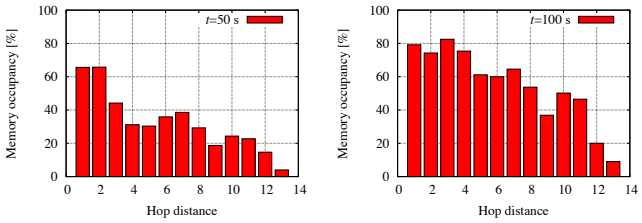


Fig. 4. Average hop distance reached by the k-th replica, as a function of k. k varies between 1 and 3 ($R = 3$), 1 and 5 ($R = 5$), 1 and 7 ($R = 7$), respectively. The distance is calculated from the position of the first replica.

At this point, it is of interest to evaluate the data placement throughout the WSN over time. As discussed previously, the



(a) $t = 50$ s (b) $t = 100$ s

Fig. 5. Percentage of memory occupancy varying the hop distance from the DAG root, at different time instants. Memories of nodes closer to the DAG root saturate faster. The number of replicas R is set to 7.

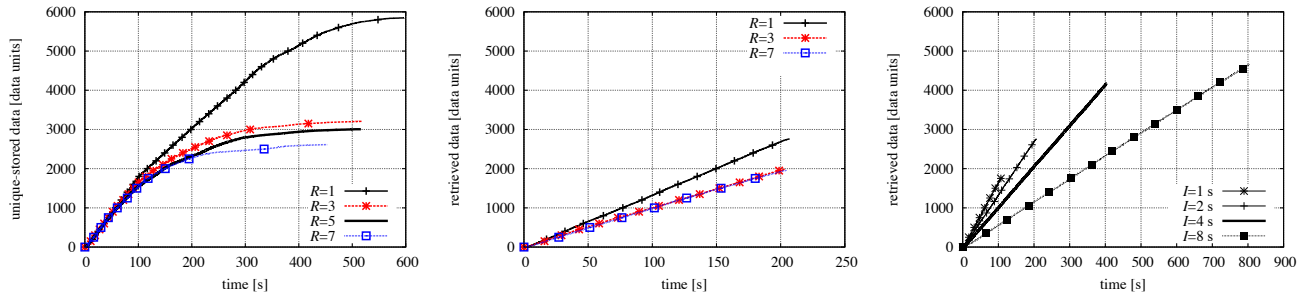
mechanism distributes replicas prioritizing donor nodes closer to the DAG root. Fig. 5 shows the average saturation level of the memories of the nodes at different hop distances from the DAG root, for several observation instants. It can be noticed that the portion of the DODAG closer to the root fills the memories faster. This can be of help in the data retrieval phase, as data needs to follow a shorter path to reach the DAG root.

In Fig. 6(a), the amount of *unique* stored data (i.e., copies are not accounted for) is shown, as a function of time. Without redundancy ($R = 1$), the amount of unique stored data reaches the network storage capacity $C = N \times B = 6000$ data units. In the presence of redundancy ($R > 1$), several copies of each data unit are distributed in the WSN and, therefore, a smaller amount of unique data is stored.

As for data retrieval, in the case with $R > 1$, less data is delivered to the DAG root, since only the closest replica is sent. On the other hand, with $R = 1$, all $N \times B = 6000$ data units are delivered to the sink; this increases the amount of many-to-one traffic, and, consequently, the risk of collisions. To reduce the impact of collisions, data units are transmitted, from each node towards the DAG root, one at a time. The interval between two consecutive transmissions is denoted as I (dimension: [s]). Intuitively, a shorter interval may speed up the retrieval process, but it may also increase the collisions' probability and, therefore, the percentage of data loss at the DAG root. On the other hand, longer values of I may increase the amount of retrieved data at the sink, penalizing the latency of the retrieval phase.

In Fig. 6(b), the amount of retrieved data at the DAG root is shown, as a function of time, for various values of R. The interval I is set to 2 s in this case. As expected, the amount of retrieved data decreases with R, since there is more unique data in the system when no redundancy is required. However, it can be observed that, with $R = 1$, only a small fraction, i.e., about 50% of the stored data $C = N \times B$, is successfully retrieved; the rest is lost because of collisions. For higher values of R, the corresponding percentage of retrieved data increases significantly. For instance, in the case with $R = 7$, about 2000 data units are retrieved, which corresponds to approximately 80% of the total amount of unique data.

Collisions are influenced not only by R, but also by I. Fig. 6(c) shows that, with no replication ($R = 1$), the amount of retrieved data significantly increases for higher values of I.



(a) (b) (c)

Fig. 6. Stored and retrieved data considering memory size equal to 100 data units, $N = 60$ storage nodes, and a memory advertisement period of 30 s; (a) unique stored data; (b) retrieved data for several values of R, with $I = 2$ s; (c) retrieved data for several values of I, with $R = 1$.

To summarize, the percentage of retrieved data amongst the total unique stored data in the system is shown in Table II, for various combinations of R and I.

	$I = 1$ **s**	$I = 2$ **s**	$I = 4$ **s**	$I = 8$ **s**
$R = 1$	33%	50%	68%	82%
$R = 3$	49%	66%	95%	100%
$R = 5$	50%	74%	95%	100%
$R = 7$	55%	80%	100%	100%

TABLE II
PERCENTAGE OF RETRIEVED DATA.

VI. CONCLUSIONS

This paper has addressed the problem of redundant data distribution and retrieval for WSN-based observation systems. A redundant distributed data storage mechanism, built on top of RPL, has been proposed in order to increase the resilience and storage capacity of a WSN against node failure and local memory shortage. The performance has been evaluated extensively through simulations. The mechanism lends directly to the implementation a complimentary data retrieval scheme, whose performance has been evaluated as well. Our results show clearly a trade-off between storage redundancy (which depletes the total available storage memory) and retrieval efficiency (in terms of percentage of retrieved data).

Future research activities will include the use of other parameters in the replication strategy, 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., ContikiMAC or X-MAC). We also envision to study dynamic reconfigurations of node behaviors (e.g., sampling) and communication layers (e.g., transmitting power, duty-cycle) to meet replication demands with minimum energy cost. Finally, we would like to evaluate performance of the mechanism on real testbeds.

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 *The Fifth International Conference on Information Processing in Sensor Networks* (IPSN'06), Nashville, TN, USA, 2006, pp. 374–381.

[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), San Jose, CA, USA, October 2002, pp. 96–107.

[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), Atlanta, GA, USA, September 2002, pp. 68–77.

[4] P. Gonizzi, G. Ferrari, V. Gay, and J. Leguay, "Redundant distributed data storage: experimentation with the SensLab testbed," in *Proc. Int. Conf. Sensor Networks* (SENSORNETS), Rome, Italy, Feb. 2012.

[5] 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), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[6] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings of 31st IEEE Conference on Local Computer Networks*, nov. 2006, pp. 641 –648.

[7] 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), Shenzhen, China, April 2010, pp. 260–263.

[8] A. Omotayo, M. Hammad, and 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, April 2007, pp. 29–38.

[9] 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), Jeju Island, Korea, December 2009, pp. 413–420.

[10] G. Maia, D. L. Guidoni, A. Carneiro 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," INRIA, Research Report, July 2011.

[11] 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), Berkeley, CA, USA, november 2009, pp. 1–14.

[12] 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), Cambridge, MA, USA, april 2007.