European Seventh Framework Programme
FP7-288879-Collaborative Project

D4.43 Protocols for data aggregation and data centric communications.

The CALIPSO Consortium

TCS, Thales Communications & Security, France
CNRS, Centre National de la Recherche Scientifique, France
SICS, Swedish Institute of Computer Science, Sweden
UPA, University of Parma, Italy
DRZ, Disney Research Zurich, Switzerland
WOS, Worldsensing, Spain
CISCO, Cisco Systems International B.V., Netherlands

Document Information

Contract Number	288879
Deliverable Name	Protocols for data aggregation and data centric communications.
Deliverable number	D4.43
Editor(s)	Andrzej Duda (CNRS)
Author(s)	Andrzej Duda (CNRS) Michał Król (CNRS) Martin Heusse (CNRS) Simon Duquennoy (SICS) Pietro Gonizzi (UPA) Marc Domingo (WOS) Paolo Medagliani (TCS)
Reviewer(s)	Jérémie Leguay (TCS), Paolo Medagliani (TCS)
Dissemination level	Public
Contractual date of delivery	9/2013
Delivery date	9/2013
Status	Final
Keywords	CALIPSO, data-centric communications, data aggregation

This project is funded under 7th Framework Program

Contents

1. Overview

This deliverable shows CALIPSO contributions to Task 4.3, "Data aggregation and data centric communications". First, we provide a high-level overview of the technical contributions of this document, putting them in realation with the CALIPSO architecture and the application scenarios. Then, we present each contribution in details.

2. Overview of technical contributions

In Section 3.1, we report on the current work on data aggregation within Calipso and explain the limitations of the proposed approaches.

In Section 3.2, 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. Results show that RPL can be used for robust and energy-efficient distributed data storage and retrieval. This work is published in IEEE IWCMC 2013 [7].

In Section 3.3, we present the Hybrid Data Collection Protocol (HDCP) that addresses the problem of transferring data from some intended sources to a given sink over the RAWMAC layer.

In Section 3.4, we present Datacast, a new communication primitive that provides support for in-network processing and aggregation of sensor data. The presented work is preliminary, not yet published, and subject to on-going work of evaluation.

2.1. Relation to the CALIPSO architecture

Figure 1 highlights how each contribution fits to the corresponding components of the CALIPSO functional architecture. The contributions of this document are concentrated at the network plane.

First, we report on actual solutions for data aggregation. Then, we present a in-network data storage mechanism that allows to locally store data, providing their transmission only in case of local memory shortage. This implementation allows to reduce communications and meet energy efficiency requirements. We also describe a data collection strategy based on the protocols used at lower layers. This strategy identifies two classes of data, that is low-priority and high-priority data, and guarantees that, even in case of transmission of burst of low-priority data, high priority data can always reach the collection point. Finally, we propose solutions for in-network storage and processing. Such support can enable flexible operation of applications that require to collect, aggregate or process sensor data. Reducing the volume of data conveyed in the network leads to energy savings and increased performance.

2.2. Relation to the CALIPSO application scenarios

Table 1 presents the relation between modules described in this deliverable and the respective CALIPSO application scenarios. The table is given for informational purposes only and it does not yet reflect exactly what the field trials will contain and evaluate. We rather expect the field trials to focus on a selected subset of these modules.

The Critical Infrastructure scenario is also based on RPL and 802.15.4 with data robustness as a main requirement. We propose practical solutions to reliably store redundant copies of data within the WSN using RPL for an efficient data placement. In addition, in case of alarms

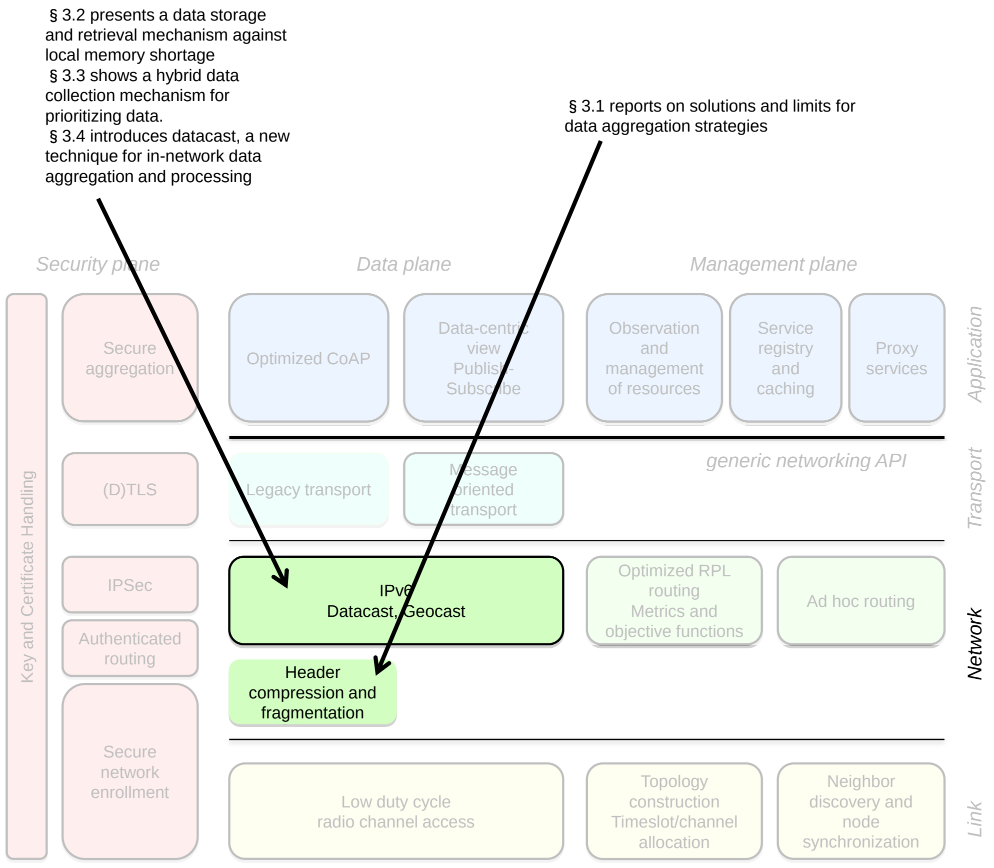

§ 3.2 presents a data storage and retrieval mechanism against local memory shortage
§ 3.3 shows a hybrid data collection mechanism for prioritizing data.
§ 3.4 introduces datacast, a new technique for in-network data aggregation and processing
§ 3.1 reports on solutions and limits for data aggregation strategies
Security plane
Data plane
Management plane
Key and Certificate Handling
Secure aggregation
Optimized CoAP
Data-centric view Publish Subscribe
Observation and management of resources
Service registry and caching
Proxy services
Application
(D)TLS
Legacy transport
Message oriented transport
generic networking API
Transport
IPSec
Authenticated routing
IPv6 Datacast, Geocast
Header compression and fragmentation
Optimized RPL routing Metrics and objective functions
Ad hoc routing
Network
Secure network enrollment
Low duty cycle radio channel access
Topology construction Timeslot/channel allocation
Neighbor discovery and node synchronization
Link

Figure 1: Relation of the contributions to the CALIPSO functional architecture.

or, more in general, of priority data, the critical infrastructure scenario requires for a strategy to accelerate and guarantee their reception.

Smart parking and critical infrastructure scenarios may also benefit of in-network data aggregation to efficiently collect data limiting energy consumption associated to useless packet transmissions.

Module	Scenario	Justification
Distributed In-Network storage	Critical Infrastructure	Critical Infrastructure scenario requires resilient in-network data storage techniques which minimize the risk of data loss
Hybrid Data Collection Protocol	Critical Infrastructure	The protocol improves data transport in the Critical Infrastructure scenario
Datacast	Critical Infrastructure, Smart Parking	Datacast enables in-network processing and aggregation.

Table 1: Relation between modules and application scenarios

3. Technical Contribution

3.1. Report on Data Aggregation

As already mentioned before, data compression was discarded since it requires to break the notion of layered protocol stack implying a cross layer approach. In this section Local Controllers for 6LowPAN and its contexts are going to be explored in order to reduce the size of the headers in quasi-static applications where the context remains the same. This approach is only explored and not gone further since it was found that its impact is not as high as initially expected.

Low power and Lossy Networks (LLNs) allow to interconnect a large number of resource-constrained devices, forming a wireless mesh network. To be connected to the Internet, a small number of border routers (BRs) usually serve as gateways between each LLN and the Internet. Such LLNs have a wide range of applications, including building and home automation, industrial process control and smart urban environments.

A chief component of these networks is the wireless communication technology. The 6LoWPAN, ROLL and CoRE IETF Working Groups have defined protocols at various layers of the LLN protocol stack, including an IPv6 adaptation layer, 6LoWPAN [19], a routing protocol, RPL [22] and a web transfer protocol, CoAP [21]. From a standardization point of view, header compression can be used to reduce the amount of information being sent by nodes operating on these networks, this can be achieved by the use of 6LoWPAN Contexts as defined by RFC6775 [20]. In the scope of CALIPSO, data aggregation has been studied in the context of standards and specifically the header compression support provided by 6LoWPAN. The later is motivated by the need for widely usable aggregation techniques in very different scenarios as the targeted by CALIPSO and by the requirement to keep standards compatibility and do not break the network layered approach. Data aggregation as studied by Krishnamachari. et.al [12] and by Alzaid et.al. [1] rely on the possibility to correlate data by temporal or spatial correspondence which might not be possible in time sensitive applications (e.g Parking application). Other approaches such as those that aggregate data along a multi-hop route break the notion of a layered protocol stack as require some cross layering in order to aggregate the data in a hop by hop basis.

3.1.1. 6LoWPAN Contexts

A Context is an important concept in a 6LoWPAN network which is used to store prefixes of addresses subsequently used for address compression. The main idea is to have a Context field in the 6LoWPAN header that can be used to indicate whether some fields on the header are compressed or not. The Context is identified by a Context prefix that uniquely maps to the elided information.

The 6LoWPAN Context Option (6CO) carries prefix information for 6LoWPAN header compression and is similar to the PIO of RFC4861 [15]. However, the prefixes can be remote as well as local to the LoWPAN, since header compression potentially applies to all IPv6 addresses. This option allows for the dissemination of multiple contexts identified by a Context ID for use as specified in RFC6282 [10]. A context may be a prefix of any length or an address (/128), and up to 16 6COs may be carried in a Router Advertisment (RA) message.

3.1.2. Network Formation and Context Distribution

A LBR handles the sending of Router Advertisments and processing of Neighbor Solicitations from hosts as described by RFC6775. A LBR has always to include its own address as the LBR address and disseminate it through the network. In addition, a LBR is configured with the prefix or prefixes that are assigned to the LoWPAN and advertises those in RAs as described in RFC4861. When 6LoWPAN uses header compression [RFC6282] with Context information, the LBR needs to manage the Context IDs and advertise those in the RAs by including 6COs in its RAs so that direct neighbors are informed about the Context IDs and the data that can be elided from 6LoWPAN Headers.

3.1.3. Storing Contexts

Contexts can be stored inside or outside the LoWPAN boundary. LoWPAN Border Routers (LBRs), maintain a conceptual data structure for the context information they receive from network nodes or routers. This structure is called the context table. It includes a Context ID, a unique prefix (from the Context Prefix field in the 6CO), a Compression bit indicating whether the context can be used to decompress incoming headers or not (receive mode) and the Context Lifetime.

When a node receives a Router Advertisement including a Context field, this is used to add or update the information in the context table. If the Context ID field in the packet matches an existing context table entry, then that entry is updated with the new information otherwise a new entry is added to the context table. When the LBR changes the context information, a node might not immediately notice the update and therefore some inconsistency might appear in the context information.

When the Context Lifetime for a context table entry expires, the entry is placed in a receive-only mode, the entry then is held in receive-only mode for a period of twice the default Router Lifetime, after which the entry is removed. A host should inspect the various lifetimes to determine when it should next initiate sending a Router Solicitation (RS) to ask for any updates to the information. The lifetimes that matter are the default Router Lifetime, the Valid Lifetime in the PIOs, and the Valid Lifetime in the 6CO. The host has to send one or more unicast RSs to the router well before the shortest of those lifetimes (across all the prefixes and all the contexts) expires and then switch to multicast RS messages if there is no response to the unicast messages.

3.1.4. Using Contexts

If the LoWPAN uses header compression as defined by RFC6282 with Context information, then the LBR must be configured with context information and related Contexts IDs. If the LoWPAN has multiple LBRs, then they have to be configured with the same context information and Context IDs. As noted in RFC6282, maintaining consistency of context information is crucial for ensuring that packets will be decompressed correctly. The context information carried in RA messages originates at LBRs and must be disseminated to all the routers and hosts within the LoWPAN. RAs include one 6CO for each context.

For the dissemination of context information using the 6CO, a strict life cycle is used in order to ensure that the context information stays synchronized throughout the LoWPAN. New context information has to be introduced into the LoWPAN with a read only flag, to ensure that it is known by all nodes that may have to perform header decompression based on this context information. Only when it is reasonable to assume that this information is successfully disseminated the context information should be updated so it can be used to compress headers, i.e enabling the actual use of the context information for compression.

Conversely, to avoid the situation where nodes send packets that make use of previous values of contexts, which would result in ambiguity when receiving a packet that uses a recently changed context – old values of a context have to be removed until new values are assigned to this specific context.

As soon as the Context information is installed in the network, nodes might be able to substitute the context information in a message by its Content ID, in the case that source and destination prefixes are not within the same LoWPAN, those can be substituted by the Context ID saving up to 12Bytes of data per packet.

3.2. Distributed In-Network Data Storage

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 [14], 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 intermittent 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 [11], studying weather conditions in national parks [2], 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 [6], 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) [22]. We evaluate our approach in the Contiki operating system using the Cooja [18] simulator. In particular, we show how RPL can be used for robust and energy-efficient distributed data storage and retrieval.

3.2.1. Related Work

Various schemes to efficiently store and process sensed data in WSNs have been proposed in the past years [9]. 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 [17]. 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.

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 [16]. 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 [13] 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 [5]. 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 [3] 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.

3.2.2. RPL Overview

RPL [22] 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.

3.2.3. 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.

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 2. 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)

Symbol	Description	Unit
N	Number of RPL nodes	scalar
B_i	Node i's buffer size, $i \in \{1, \dots, N\}$	scalar
r_i	Node i's sensing rate, $i \in \{1, \dots, N\}$	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 2: Main system parameters

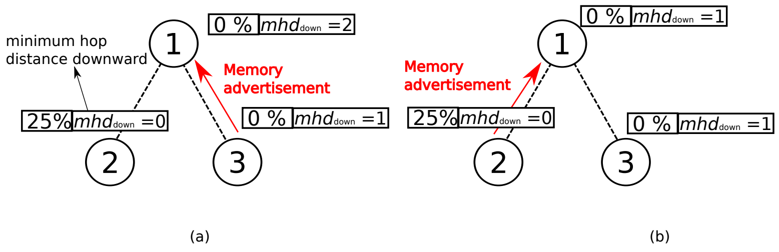


Figure 2: Messages exchanged for memory advertisements. (a) Memory availability is announced by node 3, at 1 hop in its down direction; (b) Node 1 updates mhd_{down} to 1 as it becomes aware of a closer node with some available memory.

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. 2. In Fig. 2(a), node 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. 2(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 $i \in \{1, \dots, N\}$. 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.

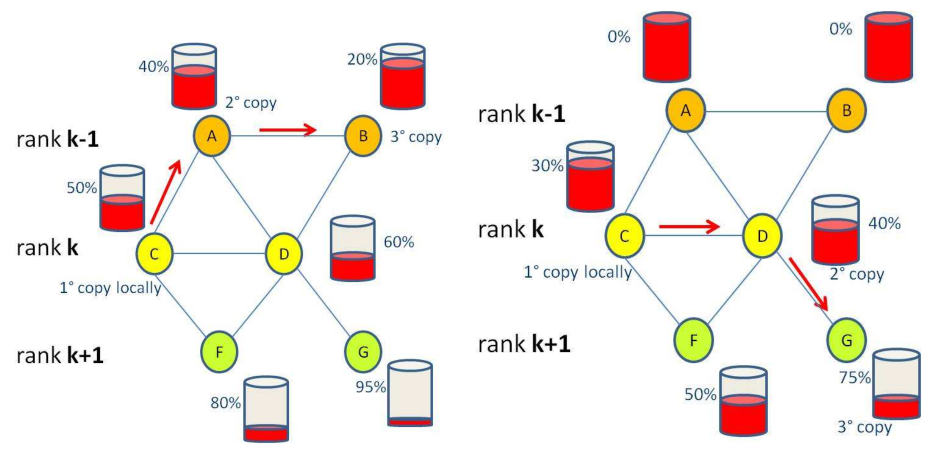
If a donor node can be selected, node i sends to it a copy of the data unit, specifying how many other copies are still to be distributed in the WSN. According to the principle outlined above, the number of required copies is set to either $R-1$ (if node i can store the original data locally) or R (if 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.

3.2.4. 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 [18]. The simulated scenario, depicted in Fig. 4a, 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



(a) (b)

Figure 3: 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.

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: 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.

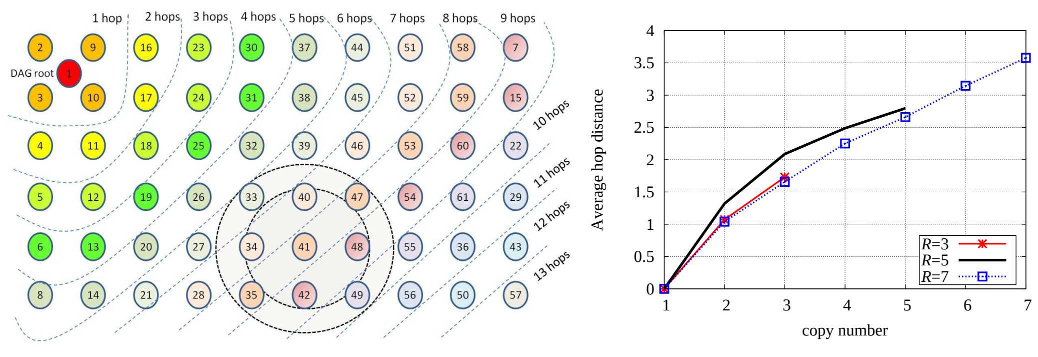
Impact of the Number of Replicas

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. 4b 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 [6] 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.

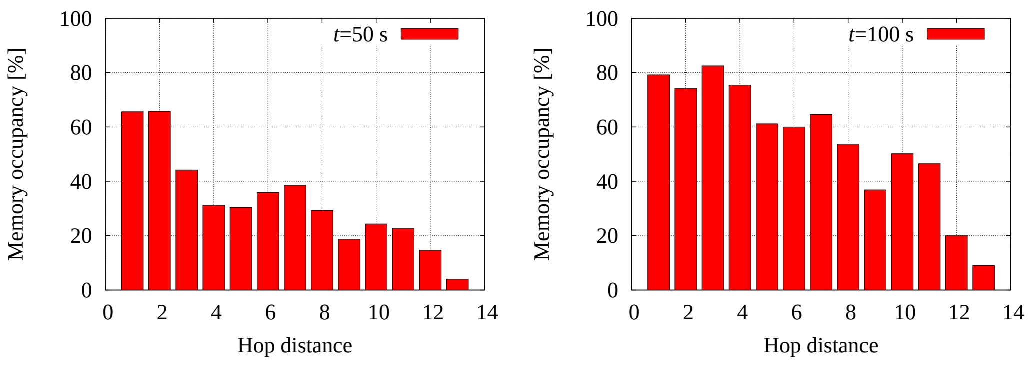
At this point, it is of interest to evaluate the data placement throughout the WSN over time. As discussed previously, the mechanism distributes replicas prioritizing donor nodes closer to the DAG root. Fig. 5 shows the average saturation level of the memories of the nodes at different hop distances 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. 6a, 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



(a) (b)

Figure 4: (a) 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. (b) 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.



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

Figure 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.

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. 6b, 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

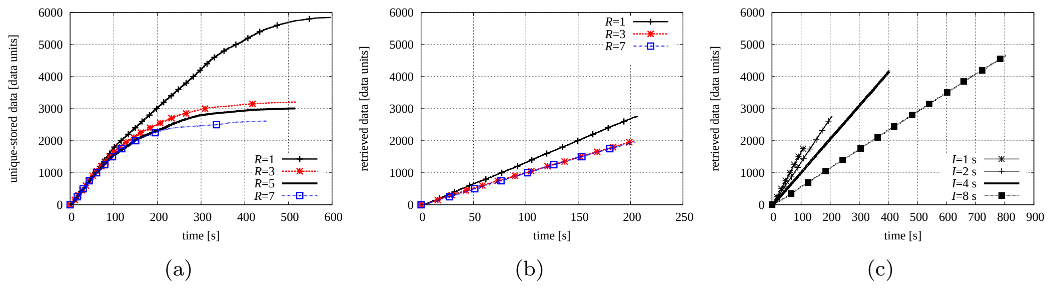


Figure 6: Stored and retrieved data considering memory size equal to 100 data units, $N = 60$ storage nodes, and a memory advertisement period of 30 s; (a) unique stored data; (b) retrieved data for several values of R, with $I = 2$ s; (c) retrieved data for several values of I, with $R = 1$.

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. 6c shows that, with no replication ($R = 1$), the amount of retrieved data significantly increases for higher values of I.

To summarize, the percentage of retrieved data amongst the total unique stored data in the system is shown in Table 3, 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 3: Percentage of retrieved data.

3.2.5. 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).

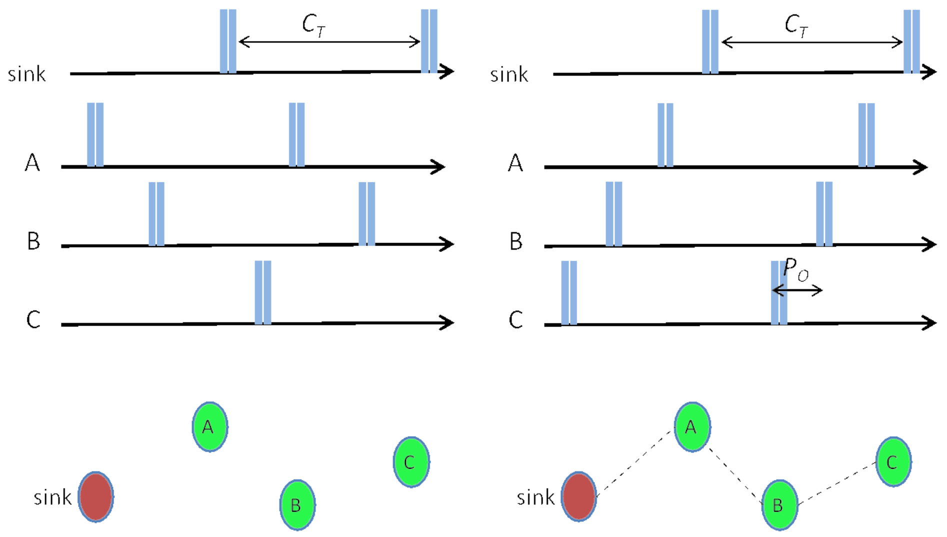
Future research activities will include the use of other parameters in the replication strategy, such as the energy consumption of the nodes or the reachability of nodes, especially if they operate a low-power MAC layer with duty-cycles (e.g., ContikiMAC or X-MAC). We also envision to study dynamic reconfigurations 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.

3.3. Advanced Cross-layer Data Collection Protocol

3.3.1. Preliminaries on RAWMAC

RAWMAC is a MAC protocol optimized for low delay events. Its goals are delay minimization and energy efficiency. In order to achieve this, it exploits on one side the basic features of ContikiMAC such as duty-cycling, low power listening, and phase lockup and, on the other side, the information on the collection tree structure coming from the RPL routing layer. The phase locking mechanism provided in ContikiMAC allows the sending node to wake up when the destination node is on and ready to receive a packet. In RAWMAC this concept has been extended and integrated with the information coming from the routing plane. Exploiting the RPL information about tree structure, a child node can synchronize with the preferred parent and shift its wake up phase according to the one of its parent. In such a way, the nodes create "waves" of aligned wake-up phases from leaves nodes to the sink. Upward delay results to be minimized because the time required to transfer a data packet is basically only the processing time of each node on the path plus the delay for entering the wave at the first hop. Similarly, energy consumption remains limited since nodes start data transmission only when they know the intended parent will be ready to receive the packet.

The basic principles of RAWMAC are shown in Figure 7. At network startup, node are

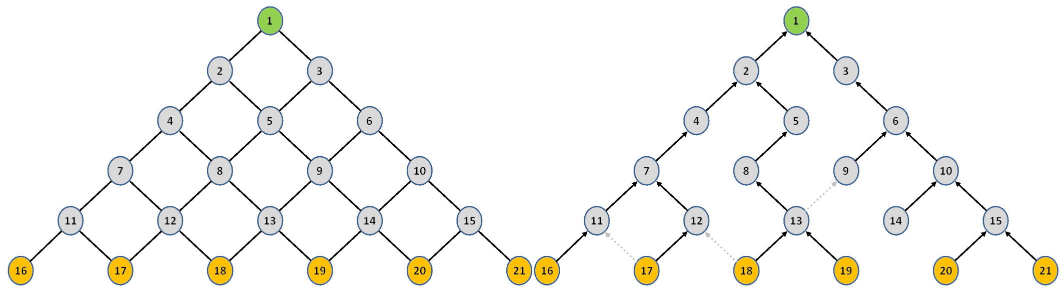


(a) Initial configuration (b) Phase shifting

Figure 7: Design principle of RAWMAC: wake-up phase alignment. In (a), the network topology is not formed yet, and nodes have random wake-up phases. In (b), after the RPL DAG is set, each node shifts its wake-up phase according to that of its parent in the DAG.

unsynchronized and the RPL tree has not been formed yet. First, nodes broadcast DIO messages so that they can create the collection tree. At the same time, the nodes exchange DAO messages from the children to the parents. Each time a DAO message generated by a node is acknowledged by the parent node, the child learns the phase of the parent and synchronizes its LPL with it. In this way, a propagation wave is created from the leaves to the root of the tree. In order to avoid loss of synchronization between nodes, each time there is an acknowledged unicast transmission from one child to its parent, this synchronization is updated.

Since packets transmitted on the wireless medium may collide, RAWMAC inherits the collision avoidance mechanism of ContikiMAC. Before transmitting, the node checks the channel



(a) Network topology (b) DAG configuration

Figure 8: Evaluation scenario. In (a), the overall topology is depicted. In (b), the configuration of the RPL DAG is shown. Nodes select a preferred parent towards the DAG root. Dashed arrows represent sporadic parent changes.

performing consecutive low-power listening operations, referred to as *Clear Channel Assessment* (CCA). If the medium is free, the packet is transmitted, otherwise it is delayed. The amount of this delay is determined by a random exponential backoff algorithm: the node selects a random instant within a given interval and waits that this interval expires before making a new transmission attempt. If the medium is detected as busy once more, the interval, on which the random instant is chosen, is doubled, so that eventual congestions are solved. The maximum number of transmission attempts is set by the standard. After that, if the packet is not transmitted, it is discarded by the node. This technique is referred to as *non-persistent Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA).

3.3.2. Application Requirements

Smart surveillance applications require efficient data collection from deployed nodes to one or more sink nodes. A typical network topology is shown in Figure 8a where some nodes acting as leaves of the collection tree sense the environment in order to collect some data of interest. Since data must be efficiently routed from the source to the sink, a routing protocol must be run inside the network. When nodes are static and the conditions of the channel do not vary very often, the RPL protocol, due to its proactive way of building paths, emerges as a suitable solution for surveillance applications. Once the tree is created, each node identifies a preferred parent, as shown in Figure 8b. This mechanism is exploited by RAWMAC to create the wave of propagation.

According to the requirements of the application, it is possible to identify two types of traffic: (i) data and (ii) alarms. For each of this class of traffic, there are different requirements in terms of performance. Generic data can be transmitted allowing a higher delay and, eventually, stored locally inside the nodes when the channel is busy. According to this strategy, especially when traffic is elevated, it is possible to bound the retransmission attempts preferring the local storage, so that energy is saved. When memory is becoming full, the node must reserve transmission resources in order to empty its memory and transmit as quickly as possible data to the sink. The time the collection operation takes has a severe impact on the performance of the system. A possible technique to efficiently collect data is presented in [4]. The storing node can initiate a burst transmission of data to the sink. The channel assessment operation is carried out only for the first transmitted packet. For the other packets, the sender starts the transmission as soon as the previous packet has been received and acknowledged by the destination node. The advantage of this technique is that it is energetically efficient since no CCA is carried out except for the first packet. However, when nodes are involved in the reception of long data burst, they cannot serve other nodes in the network.

On the other side, alarms must be sent to the sink as soon as possible, prioritizing them over generic traffic and ensuring data collection with minimum delay. Simultaneously, alarms packets should also benefit of a high reliability. The bound on delay can be achieved using, for instance, the RAWMAC protocol, whereas the reliability can be guaranteed simply using both link layer acknowledgment and end-to-end acknowledgment. However, the eventual retransmission of packets is in contrast with bounds on delay since retransmission increments the overall end-to-end delay. Therefore, specific strategies must be put in place to try to meet both the bounds on delay and the required reliability. In addition, the collection of delay tolerant data traffic may reserve the channel for a sufficiently long period of time. This creates a strong trade off with the transmission of alarms that, even if prioritized, cannot reach the sink in a suitable delay since another node is already transmitting. Also in this case, a strategy to deal with the transmission of two different types of traffic is required. In the following section we will describe how we plan to implement the data collection protocol so that objectives of delay and reliability are met, even in the presence of traffic with different priority.

3.3.3. Data Collection Protocol on Top of RAWMAC

The proposed Hybrid Data Collection Protocol (HDCP) addresses the problem of transferring data from some intended sources to a given sink. Specifically, it deals with both alarms and generic data, handling properly the different priority of alarms on the generic data. On the other side, it provides for reliability and performace bounds on delay, especially for alarm messages. As it will be shown later, the proposed solution inherits from [4] a burst transmission functionality so that the impact of low-priority collection of large amounts of data on network performance is limited. HDCP exploits cross-layer information to guarantee low-delay reliable transmissions of priority data. Using the “wave” created at the MAC layer, it allows for fast data collection of alarms. In addition, leveraging on the knowledge of the RPL tree, it may reserve transmission resources on the path between the source node and the sink, allowing efficient burst data collection.In the following, we will describe how the proposed protocol handles alarms and data singularly and how alarms are prioritized on data in the case of concurrent traffic.

Alarms

Since alarms must be transmitted as quickly as possible to the sink, HDCP exploits the “waves” created at MAC layer. As soon as an alarm is generated by a leaf node and there is no data transmission on the collection path, the source node waits for the next available activity period of the receiver and then starts transmission. On its turn, exploiting the RAWMAC functionalities, the receiving node acknowledges packet reception and, in its turn, forwards the packet towards the sink. In Figure 9 the RAWMAC transmission mechanism for energy efficient packet forwarding is shown.

In order to guarantee reliability of packet transmission, HDCP uses a two-level confirmation mechanism: (i) link layer ACK message and (ii) end-to-end ACK message. The former ACK is used between transmitter and receiver at each hop in order to prove the correct reception of a packet. In the case no ACK is received, packet is retransmitted. The latter, instead, is used to confirm the correct reception of the packet by the sink, improving the reliability of the whole mechanism. In fact, as explained in [8], the delay of the first transmission hop can be approximated as

$$d_{\mathrm{1st-hop}} = C_{\mathrm{T}}/2 + P_{\mathrm{min}} \tag{1}$$

where C_{T} is the sleeping interval (cycle time) of the node and P_{min} is minimum forwarding time needed by a node to forward a packet to a neighbor. From Eq. (1) it follows that the In

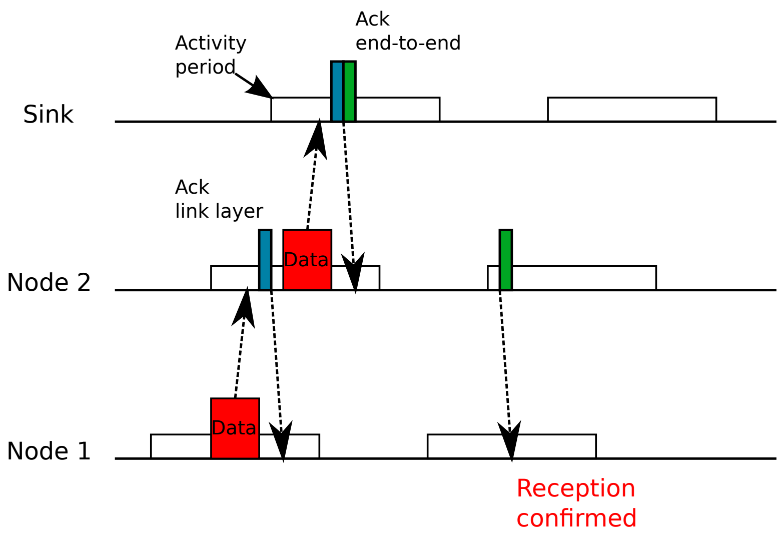


Figure 9: Transmission of alarms on RAWMAC. Each node on the path first confirms packet reception through a link-layer ACK. Once packet is received by the sink, this confirms the good reception through an end-to-end ACK message.

this case, the upward delay $D_{\mathrm{up,R}}$ can be expressed in two ways, depending on the value of P_{o}, the offset between the wake-up phases of two nodes:

$$D_{\mathrm{up,R}} = \begin{cases} (h-1)\cdot P_{\mathrm{o}} + d_{\mathrm{1st-hop}} & P_{\mathrm{o}} > P_{\mathrm{min}} \\ (h-1)\cdot (P_{\mathrm{o}} + C_{\mathrm{T}}) + d_{\mathrm{1st-hop}} & 0 \le P_{\mathrm{o}} \le P_{\mathrm{min}} \end{cases} \tag{2}$$

From Eq. (3) we can see that, basically, depends on the number of hops traversed by a packet and the delay for entering the wave. However, in case of retransmissions due to packet losses the delay is impacted also by the time required to renter inside the following wave. The upward delay $D_{\mathrm{up,R,Retx}}$ can be then refined as

$$D_{\mathrm{up,R,Retx}} = \begin{cases} (h-1)\cdot P_{\mathrm{o}} + d_{\mathrm{1st-hop}} + N_{\mathrm{reTx}}\cdot C_{\mathrm{T}} & P_{\mathrm{o}} > P_{\mathrm{min}} \\ (h-1)\cdot (P_{\mathrm{o}} + C_{\mathrm{T}}) + d_{\mathrm{1st-hop}} + N_{\mathrm{reTx}}\cdot C_{\mathrm{T}} & 0 \le P_{\mathrm{o}} \le P_{\mathrm{min}} \end{cases} \tag{3}$$

where N_{reTx} denotes the number of times a packet is retransmitted. This means that if a strict bound on delay is given by system requirements, it may happen that in case of multiple retransmissions of the same packet the bound on delay is no more respected. In such a case, our protocol prefers to discard the packet and let eventual other alarms be transmitted to the sink. In order to keep trace of the number of retransmissions of a packet, a specific header must be inserted into the MAC layer packet. Since the nodes have the same duty-cycle, each node, looking at the value of this field, can decide if forwarding the packet or dropping it because it will not respect the bound on delay.

Given that alarms packets need to be transmitted as soon as possible, the RAWMAC will be coupled with a 1-persistent backoff protocol: when a packet needs to be transmitted, if the medium is detected as free by the transmitting node, the packet is sent immediately. Otherwise, it is delayed by a random time interval. At the end of this waiting window, the node tries immediately to resend the packet. If the medium is free, packet is transmitted, otherwise it is delayed once more. This procedure is repeated three times. After that, packet is discarded in order to avoid sending too outdated information. We point out that the overall delay is calculated since the generation instant of the packet and not since the instant of its transmission.

Thus, this backoff procedure has an impact on delay: in the case the delay accumulated by a packet on its path is too elevated and does not respect the imposed bound, the packet is discarded.

Delay-tolerant Data

Differently from alarms, generic data have not constraints on delay and reliability. For such a reason, their transmission can be delayed, to avoid overloading the channel, preferring a temporary local storage of data. Alternatively, the protocol can opportunistically exploit the ongoing waves to transmit data whether no one is using the channel. This choice depends on application needs. The way HDCP will handle data traffic will be forcedly different.

In case of opportunistic data transmission, the node senses the channel through the CCA mechanism. If it is free, it may transmit the packet exploiting the "waves" provided by RAW-MAC, otherwise it stores data into its local memory for a later transmission. Avoiding re-transmission attempts allows to reduce energy consumption of nodes and collisions between packets. This feature is optional and may depend on application needs. For instance, when data collection is not priority, nodes can directly store data locally. Given the loose constraints, data packets use the *p-persistent* protocol: when the channel is free, the node transmits with probability p and delays its transmission with probability $1 - p$. Once this interval has passed, the node retries transmitting using the same policy. Instead, if the channel is busy, the node simply delays the transmission. At the following transmission attempts it will use the same scheme.

When memory occupation passes a given threshold, the node must wake up to send stored data as quickly as possible to the sink. For such a reason, it exploits the solution described in [4] to send data as burst. In fact, the goals of this collection phase are the maximization of energy efficiency and the minimization of the impact on transmission of other data or alarms. The node that needs to start the transmission reserves one or more slots on the nodes on the path to the sink. In order to keep this mechanism independent from any transport protocol, this reservation is carried out through an exchange of SYN/ACK-like messages below the transport protocol. If no other reservations are active, each node that receives the RES command sets allocates a the requested temporal slots to the demanding node, otherwise it simply discards the RES and the burst transmission cannot begin. Since the transmission of the RES is carried out using the "wave" created through RAWMAC, the choice of the parent to which transmit the packet is driven by the RPL information available for each node. Once all the nodes have set the slot, the sink node answers through an ACK which certifies that the slot has been set on all the nodes and the transmission can begin. In this case, the nodes cannot transfer other data except for those of the node has reserved the slot. If no data flows for a given period of time, the reservation is canceled to avoid reserving the resource uselessly. A logical scheme of the path reservation mechanism is shown in Figure 10.

According to the protocol, non-priority data are transmitted in burst between two consecutive active phases, so that the wave communication is reserved for other types of traffic or for the exchange of ACK messages. Normally, data requires a CCA in order to start transmission. However, this operation is quite time and energy consuming. For such a reason, basing on the assumption that during the collection phase only the node that has reserved a session is served, for bursts of data it is possible to carry out the CCA check only for the first transmitted packet. For subsequent packets, the source node can transmit a packet as soon as it has received the ACK of the previous one. In Figure 11 it is shown how this burst mechanism handles data transmission.

As it is, this solution is efficient only for a 1-hop transmission since with a consecutive burst transmission the receiving node cannot forward data packet unless until the end of the

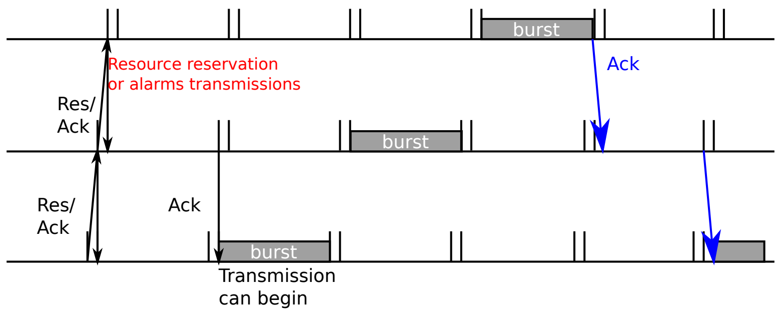


Figure 10: Scheme of the resource reservation procedure and transmission of bursts.

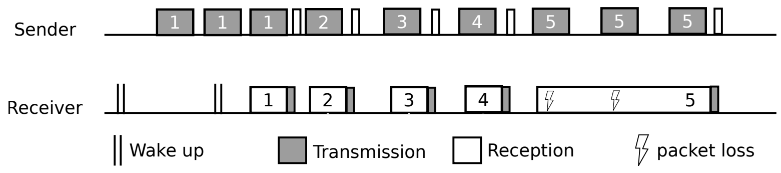


Figure 11: Transmission of data using the burst mechanism. CCA is carried out only for the first packet. In case of missed acknowledgment, packet is retransmitted.

reception of all the data packets. Several alternatives are possible, like alternating transmission and reception phases or using different transmission channel to reduce interference. What we consider in this work is shown in Figure 10. The sender transmits its burst of data during two activity phases and then it goes back to sleep waiting for an ACK. The receiver, on its turn, forwards data packets to its preferred parent and goes back to sleep. Once the sink node receives the whole burst, it confirms the correct reception through an ACK. When the node originating the burst receives the ACK, it starts with the transmission of the subsequent burst, if available. The notification of the presence of other burst to be transmitted is carried out, as suggested in [4], exploiting the *pending bit* in the 802.15.4 header to know if there is other incoming data that requires it to keep its radio interface on. Intermediary nodes simply forward data, without performing any reassembling operation. It will only be the sink node that will be in charge of reassembling packets. This will simplify the operations of intermediary nodes.

How to Prioritize Alarms

In surveillance scenarios, it is possible to have different classes of traffic, with different requirements in terms of performance and reliability. In the design of a data collection protocol it is mandatory to respect all the constraints coming from application layer but, at the same time, to provide a fair treatment to all the traffic types. This means to avoid that the transmission of a type of traffic prevents other nodes from sending either their same-class traffic or other-class traffic. For instance, referring to previous sections, the fact of reserving a slot for data collection must not prevent the other nodes from sending their data or transmitting alarms. For such a reason, it is important to have a mechanism where periodically nodes can compete for medium access, so that equity is guaranteed.

In Figure 12, we show how we are going to implement this coexistence. Nodes use the waves to reserve slots as described in the previous paragraph. The fact of maintaining a wave structure allows a generic node that has an alarm to transmit during the next wave, keeping minimized the delay and assuring the required reliability and ensuring that alarms will be transmitted with higher priority than other data. If the transmission of generic data inside the wave is enabled,

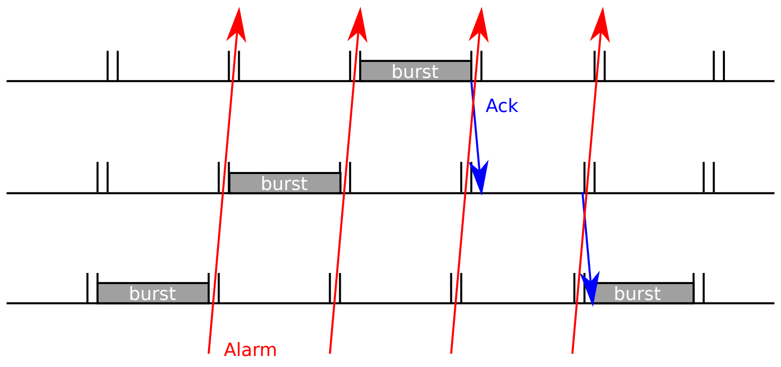


Figure 12: Strategy for the coexistence of alarms and data packets. Alarms are sent inside the wave, so that their delay constrains are guaranteed.

the fact that alarms have a 1-persistent retransmission scheme and data have a *p-persistent* scheme makes more likely that alarms will be transmitted rather than generic data that, in their turn, will be stored in memory since the channel is not available.

The length of the transmitted buffer introduces a trade off between memory occupation and system performance. Since nodes are strongly limited in memory, data are not only stored in RAM memory but also in flash memory, which is more energy consuming. This means that buffer emptying should occur often, to keep memory occupation low. However, on the other side, reserving resources for a transmission implies the reduction of the overall network capacity, since only the node which is reserving the slot will effectively send data. In addition, this operation is quite time and energy consuming since it requires the exchange of a RES/ACK message with the nodes on the path from the leaf to the sink. Therefore, it may be preferable to speed up data collection storing data locally and emptying the memory at one time.

3.4. Datacast

3.4.1. Introduction

Aggregation and in-network processing are interesting features in Wireless Sensor Networks. Transforming for instance many data packets into a single one allows to greatly decrease the number of exchanged messages and increase the lifetime of a network. From the point of view of an application, sensor data can be easily processed and its volume reduced by intermediate nodes. Moreover, event driven networks may benefit from some processing or filtering capabilities in intermediate nodes: they may act as filters forwarding only packets meeting some criteria.

To aggregate data, we usually need some dedicated nodes with more resources—a bit higher amount of memory and CPU power. There are already many proposals for performing aggregation in Wireless Sensor Networks. We can distinguish two basic approaches. In the static one, all nodes know the sensor node responsible for aggregating their data. Such a solution does not require any special communication support, but we need to place aggregating nodes very carefully for efficient aggregation and nodes need to know the aggregating node. However, in many cases such as in environmental monitoring, specific placement may be impossible. The static placement for aggregation may also perform badly after topology changes and it is not very flexible. In the second dynamic approach, we can choose aggregating nodes on demand, which is much more flexible, but it implies some overhead of managing the nodes, which may weaken the advantages of the aggregation.

We propose Datacast as a simple communication primitive that enables flexible data aggregation and in-network processing as well as event filtering while keeping the overhead minimal.

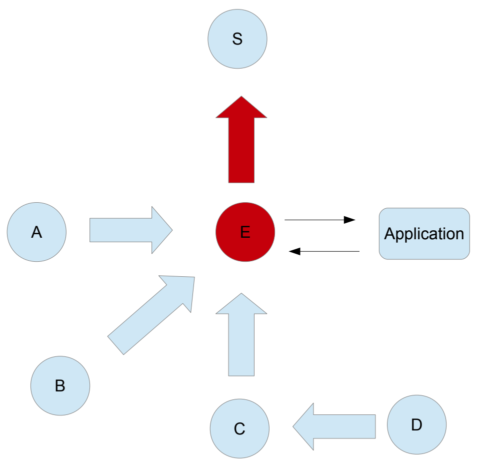


Figure 13: Datacast

3.4.2. Datacast

Datacast is a protocol that allows marking some packets as containing data for aggregation, processing, or filtering at the application layer (cf. Figure 13). A datacast packet has the same structure as a regular IPv6 packet, but it uses an additional hop-by-hop header to mark its special communication semantics. A node sends a datacast packet to a destination node (sink in case of convergecast type of traffic, e.g. Node S) and it takes a given route defined by routing tables towards a destination (for instance from Node D to Node S through Node E in Figure 13). If the packet arriving at a processing node is recognized as datacast, the IP stack passes it over to the application layer who decides what to do with the packet (at Node E in Figure 13). It can be discarded, forwarded to its original destination without modification, or processed and then forwarded. Such a set of possible actions allows to efficiently support aggregation or filtering by first receiving some packets and then generating aggregated data or forwarding only packets meeting some given conditions. A packet not intercepted by any of intermediate nodes will be delivered to its final destination just as a regular unicast packet. In Datacast, in-network processing is transparent; i.e. sensor nodes that generate data, do not have to know which nodes perform some processing of packets or even if there is a node that performs some aggregation of filtering. They just mark packets, send them to a destination node and the packets will be processed by any node capable of doing it. If there is no sensor capable of processing Datacast packets on the path to the destination, they will be delivered in their original form. Such a transparent operation makes Datacast flexible without the introduction of the overhead for spreading the information about the nodes that process packets in the whole network.

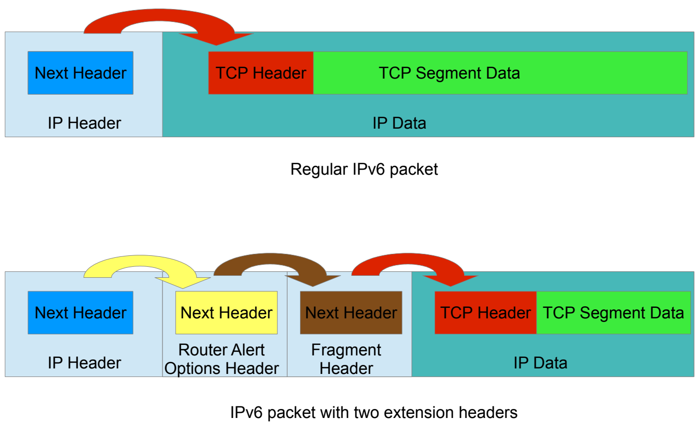


Figure 14: Next header in IPv6

3.4.3. Implementation

To mark packets as Datacast, we use the feature of IPv6 extended headers: an IPv6 packet contains its basic header and one or several extended headers (cf. Figure 14). The basic IP header provides the information about the next header, which is usually an UDP or TCP header in case of regular IP packets. If needed, the basic header may point to optional IPv6 headers that hold additional information. The hop-by-hop header contains the information that all nodes on the path to the destination need to consider. We use a special version of the Routing Alert option (RFC 2711) that forces all routers to examine a given packet more closely. Nodes that do not recognize the option, just ignore it. The option has a fixed size and contains a value field informing what it is about (we have set this value for Datacast to 12).

After finding this option in an IPv6 packet, the protocol stack checks the destination port indicated at the transport layer and will look for a socket associated with the same port. If such a socket exists, the packet is passed to the application instead of forwarding it to its final destination. The application has access to the source and destination IP addresses so it can distinguish between different flows and apply appropriate processing to the packet.

We have implemented Datacast in Contiki OS, so it has become an integral part of its micro IPv6 stack. Currently, Datacast only supports UDP packets. To receive a Datacast packet, an application has to create an UDP socket and bind it to a corresponding port. To inform the application about new data, the IP stack will generate a special event similar to the reception of a regular unicast packet.

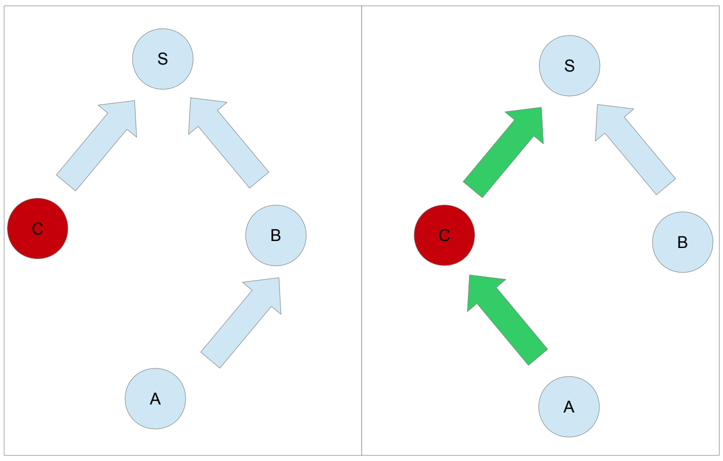


Figure 15: Changing routing tables to decrease the number of exchanged messages. Node A may change its preferred parent to Node C, if Node C is capable of aggregating data.

3.4.4. Ongoing work

RPL uses a DODAG tree with the sink as a root to collect data from sensor nodes. The optimal DODAG minimizes the overall cost of path from all nodes to the sink. Introducing aggregation through Datacast changes this situation. We want to minimize the cost of gathering data taking into account aggregating nodes. RPL allows nodes to have different instances of the routing scheme. It means that we may have a different topology for regular data traffic and for data subject to aggregation. Nodes performing aggregation shall advertise this capability to other nodes allowing to route packets efficiently. We want to develop a new metric allowing to meet these requirements. Nodes performing aggregation shall be more likely chosen as preferred parents as shown on Figure 15.

4. Conclusion

In this deliverable, we have reported on four parts of the work related to data aggregation and data centric communications. As explained in the section on data aggregation, we have analyzed possible approaches to data aggregation and explored some directions. The work has not gone further since it was found that its impact is not as high as initially expected. We have worked on other aspects related to in-network storage and processing, and come up with new interesting results. Some work on these aspects will continue during the third year of the project.

References

[1] H. Alzaid, E. Foo, and J. G. Nieto. Secure data aggregation in wireless sensor network: a survey. In *Proceedings of the sixth Australasian conference on Information security - Volume 81*, AISC '08, pages 93–105, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[2] 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.

[3] 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), pages 450 –459, Cambridge, MA, USA, april 2007.

[4] S. Duquennoy, F. Österlind, and A. Dunkels. Lossy Links, Low Power, High Throughput. In *Proc. of the ACM Conference on Networked Embedded Sensor Systems, (ACM SenSys 2011)*, Seattle, WA, USA, Nov. 2011.

[5] 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), pages 1–14, Berkeley, CA, USA, november 2009.

[6] 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.

[7] P. Gonizzi, G. Ferrari, P. Medagliani, and J. Leguay. Data Storage and Retrieval with RPL Routing. In *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference* (IWCMC 2013), Cagliari, Italy, July 2013.

[8] P. Gonizzi, P. Medagliani, J. Leguay, and G. Ferrari. RAWMAC: A Routing Aware Wave-based MAC Protocol for WSNs. In *Fifth Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Como, Italy, Sept. 2013. Submitted.

[9] 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.

[10] P. T. J. Hui. RFC 6282: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks.

[11] 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.

[12] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCSW '02, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[13] 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. Research report, INRIA, July 2011.

[14] 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), pages 374–381, Nashville, TN, USA, 2006.

[15] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. RFC 4861: Neighbor Discovery for IP version 6 (IPv6).

[16] 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.

[17] 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), pages 29–38, Istanbul, Turkey, April 2007.

[18] 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*, pages 641 –648, nov. 2006.

[19] Z. Shelby and C. Bormann. *6 LoWPAN: The Wireless Embedded Internet*. Wiley Publishing, 2010.

[20] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. RFC 6775: Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs).

[21] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). *IETF CoRE Working Group draft*, Sep 2012.

[22] 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.