Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2020, Vol. 16(6)
© The Author(s) 2020
DOI: 10.1177/1550147720929823
journals.sagepub.com/home/dsn
SAGE

A novel approach for energy- and memory-efficient data loss prevention to support Internet of Things networks

Pooya Hejazi[1] and Gianluigi Ferrari[2]

Abstract
Internet of Things integrates various technologies, including wireless sensor networks, edge computing, and cloud computing, to support a wide range of applications such as environmental monitoring and disaster surveillance. In these types of applications, IoT devices operate using limited resources in terms of battery, communication bandwidth, processing, and memory capacities. In this context, load balancing, fault tolerance, and energy and memory efficiency are among the most important issues related to data dissemination in IoT networks. In order to successfully cope with the abovementioned issues, two main approaches—data-centric storage and distributed data storage—have been proposed in the literature. Both approaches suffer from data loss due to memory and/or energy depletion in the storage nodes. Even though several techniques have been proposed so far to overcome the abovementioned problems, the proposed solutions typically focus on one issue at a time. In this article, we propose a cross-layer optimization approach to increase memory and energy efficiency as well as support load balancing. The optimization problem is a mixed-integer nonlinear programming problem, and we solve it using a genetic algorithm. Moreover, we integrate the data-centric storage features into distributed data storage mechanisms and present a novel heuristic approach, denoted as Collaborative Memory and Energy Management, to solve the underlying optimization problem. We also propose analytical and simulation frameworks for performance evaluation. Our results show that the proposed method outperforms the existing approaches in various IoT scenarios.

Keywords
Storage and retrieval processes, load balancing, fault tolerance, energy efficiency, memory efficiency, data loss

Date received: 8 September 2018; accepted: 4 May 2020

Handling Editor: Mihael Mohorcic

Introduction

With the advent of Internet of Things (IoT) technologies, smart systems—such as smart cars, cyber-physical, intelligent transport systems, vehicle-to everything (V2X), transportation safety, remote medical surgery, smart grids, public protection and disaster relief, wireless control of industrial manufacturing, and smart agriculture—can now be connected to the Internet.[1] IoT comprises wireless sensor networks (WSNs) for data collection and dissemination and communication platforms to move the sensed data to the edge or the cloud for energy-efficient task execution and long-term data storage and analysis.

[1]Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran
[2]Internet of Things (IOT) Lab, Department of Engineering and Architecture, University of Parma, Parma, Italy

Corresponding author:
Gianluigi Ferrari, Internet of Things (IOT) Lab, Department of Engineering and Architecture, University of Parma, Parco Area delle Scienze, 181/A, 43124 Parma, Italy.
Email: gianluigi.ferrari@unipr.it

A WSN is a collection of small sensing devices with limited bandwidth, power, and computational capabilities. The main goal of a WSN is to gather information from a specific environment for applications such as remote monitoring and target tracking. The design of a WSN depends significantly on the application and must consider factors such as the deployment environment, the application's design objectives, the maximum cost, the available hardware, and various system constraints.[2] Fault tolerance, load balancing, and energy and memory efficiency are among the most challenging issues of a WSN. Therefore, many approaches have been proposed in the literature to handle the abovementioned issues in order to make a WSN reliable and scalable. In particular, data storage plays a key role in making a WSN effective.

Data storage approaches can be categorized into two groups: distributed data storage (DDS) and data-centric storage (DCS). To cope with energy efficiency, DDS approaches concentrate on local data storage in sensing nodes. This makes the data storage process very energy-efficient. However, in order to collect specific information stored in the network, all sensors have to be queried by means of a flooding mechanism; therefore, the data retrieval process requires a large amount of traffic and decreases the network lifetime. On the other hand, DCS approaches focus on data-centric mechanisms for storage and retrieval by means of Geographic Hash Tables (GHTs)[3] and geometric routing algorithms such as greedy perimeter stateless routing (GPSR)[4] Although DCS is more energy-consuming than DDS in the storage process, its query process is based on directed dissemination of queries to a specific node, denoted as *storage node*, so that it reduces the query traffic and increases the efficiency of the retrieval process.

Data replication over multiple storage nodes is the mechanism which both groups of data storage techniques use to handle fault tolerance and load balancing. Fault tolerance in the WSN is obtained by preventing data loss caused by energy or memory depletion in the storage node. The approaches reported in literatures[5–11] focus on data loss prevention due to energy shortages, whereas the approaches reported in literatures[12–14] concentrate on efficient memory usage of nodes in the WSN. The mechanism with which a replica is elected in DDS is based on local broadcasts between the neighbors of the storage node. This fully distributed mechanism generates a large amount of traffic and decreases the network lifetime.

In this article, we integrate DCS replication features into DDS to cover both memory and energy efficiency in data loss prevention mechanisms for WSNs. We present a novel mechanism denoted as Collaborative Memory and Energy Management (CoMEM). We divide a WSN into multiple zones: in each zone, we include a monitor node, as proposed by Chuang[9] to act as a gateway for both storage and retrieval processes. Therefore, if an event is sensed throughout the zone, the collected information is forwarded toward the monitor, which stores the summary of the received data locally and chooses the appropriate node for storing the details. On the other hand, in the retrieval process, all queries are routed toward the monitor, which decides whether to answer the query directly (summarized retrieval) or to redirect the query to a specific storage node.

In order to take into account load balancing, in the proposed optimization model, a *load balancing evaluation metric* is defined on the basis of availability of memory and energy in each IoT device. In particular, this metric considers the percentage of memory and energy availability at the storage nodes and the monitor. If this parameter falls below a properly set threshold at the storage node, the monitor will replicate the data over the most memory- and energy-available node within the zone and will balance the storage traffic load between the previous storage node and its new replica. On the other hand, whenever the percentage of the load balancing metric of the monitor falls below the properly determined threshold, the node with the highest load balancing metric is chosen by the current storage node to be the new monitor. As a result, the proposed mechanism collaboratively chooses new replicas and monitors to bring load balancing to the WSN.

Overall, the contributions of the article can be summarized as follows:

1. In order to support load balancing as well as energy and memory efficiency, we define a cross-layer optimization problem and find the optimal solution. Since the mentioned optimization problem is mixed-integer nonlinear programming (MINLP), we use a genetic algorithm to solve it.
2. We propose a heuristic algorithm, denoted as CoMEM, to solve the centralized mathematical problem in a distributed way.
3. We investigate the performance of the proposed heuristic algorithm in a comparative way with other algorithms proposed in the literature to show its memory and energy efficiency as well as load balancing.

The rest of the article is organized as follows: Section 2 is devoted to the related works on memory shortage problems leading to CoMEM in WSNs. In Section 3, we formulate the optimization problem. In Section 4, we propose our approach to integrate DCS features into DDS mechanisms. Section 5 is dedicated to simulation results. Finally, Section 6 concludes the article.

Related works

In the work by Chouikhi et al.,[15] fault tolerance WSN applications are categorized into five groups: (i) node placement, (ii) topology control, (iii) target and event detection, (iv) data gathering and aggregation, and (v) sensor monitoring and surveillance. In this section, we briefly take a look at works previously proposed in the area of memory-efficient data gathering in WSNs. Basic DCS and DDS methods are not designed for WSNs with highly mobile or unreliable nodes. Therefore, enhanced design techniques must be considered to make data dissemination mechanisms robust against data losses due to both energy and memory depletion occurrences in nodes. While several works focus on energy efficiency in WSN, our effort mainly concentrates on both energy- and memory-efficient data dissemination mechanisms.

Efficient cluster-head election is considered in the work by Behera et al.[16] for energy consumption reduction and network lifetime enhancement. The cluster head, a node responsible for handling storage and retrieval communications of the cluster, is selected as the node with the highest residual energy level among the nodes in the cluster. The proposed algorithm chooses the optimum value for the residual energy of the cluster head. The abovementioned value is used to elect cluster heads for different IoT applications. However, the amount of the available memory for cluster-head election is taken into account.

A dynamic cluster-head selection method (DCHSM) is proposed in the work by John et al.[17] to improve energy efficiency and network lifetime of the IoT monitoring zone. First, DCHSM creates clusters in the large-scale monitoring area in order to improve maximum coverage. Afterward, cluster heads are chosen on the basis of perceived probability and survival time. However, clusters are created on the basis of the locations of the targets, which make some positions of the network unreachable. Furthermore, this may cause inefficient routing in the network, thus reducing the network lifetime.

An energy-efficient clustering routing algorithm is proposed in the work by Wang et al.[18] for non-uniform traffic distribution. This algorithm considers uneven cluster formation to support load balancing and energy efficiency. Moreover, a multi-hop routing algorithm leverages the cluster-head rotation mechanism. The abovementioned algorithm considers the distance and energy cost to calculate the energy depletion of individual nodes. In order to avoid packet loss and increase reliability, memory cost can be added as the performance metric to the routing mechanism evaluation.

In the work by Gonizzi et al.[12] a low complexity distributed data replication (LCDDR) mechanism is proposed to prevent data loss due to memory shortages by replicating the data over the most memory-available node, denoted as "donor node." This increases fault tolerance and reliability in the data storage mechanism. However, the abovementioned method has some challenging issues. First, according to the results in the work by Gonizzi et al.,[12] the query process is of no concern in the proposed mechanism. Second, the local broadcast mechanism for donor node election is not energy-efficient for large-scale data dissemination in WSNs. Third, because of the fully distributed donor node election mechanism of LCDDR, a hijacking node can declare itself as the node with the highest available memory and become the donor node in the election process, thus easily sniffing data. Finally, if no donor node can be chosen, data loss will happen because of the lack of a widespread election mechanism.

In the work by Sajjadian Amiri et al.,[13] a hybrid routing algorithm, denoted as Bloom filter-based routing protocol, is proposed; it relies on hierarchical (cluster-based) routing and bloom filter data aggregation. This method decreases the amount of memory usage for the routing table at each node by means of bloom filtering and cluster-head data aggregation. However, cluster-head election is handled by local broadcasts between the neighbors of the current cluster head. Each node broadcasts a parameter, denoted as "coverage-aware cost" and calculated according to its remaining energy, to its neighbors. After the first broadcast, a node waits for a specific amount of time determined by the coverage-aware cost; if there is no announcement with a lower cost, it declares itself as a cluster head to the neighbors by means of another broadcast. This generates a large amount of traffic for cluster-head election. Moreover, the elected node is the node with the highest available energy and memory usage is not taken into account.

Another method, which relies on data compression and thus increases memory efficiency, is presented in the work by Xu et al.[14] In this case, the memory usage of nodes and the wireless bandwidth consumption are affected by a co-design memory mechanism for low-latency in-place lightweight compression. However, as the compression happens on relevant memory pages, the information can only be accessed by multiple packet payload decompression. Therefore, data aggregation at the cluster head incurs high energy consumption. This tends to deplete the energy of the cluster head. Moreover, no election mechanism for a depleted cluster head is proposed.

Memory-based message efficient clustering was presented in the work by Banerjee et al.[19] to enhance the energy efficiency of nodes by reducing the propagation of duplicated messages. Moreover, a heuristic protocol is proposed to define the distribution of the sensors over the specific sensed area. This protocol increases the energy-saving efficiency in the given network.

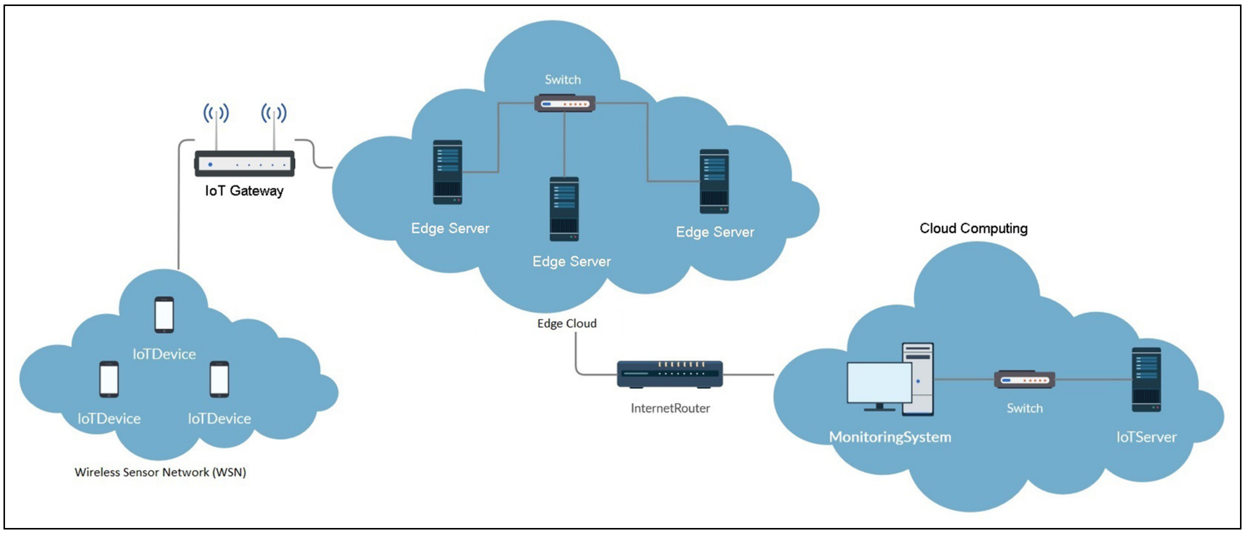

Figure 1. System architecture.

However, no data loss prevention mechanism is proposed and practical issues, such as random distribution of the sensors, are not considered.

A new transport layer protocol, denoted as reliable transport with memory consideration, is presented in the work by Zhou et al.[20] Hop-by-hop retransmission guarantees reception of specific data by the sink. It also prevents the change of transmission rate for congestion control to safeguard the control packets. Furthermore, information on the memory status of each sensor is included in the packet header for memory overflow prevention. However, inserting sensor memory information in each packet continuously causes unnecessary redundant data transmissions. Therefore, this protocol is not energy-efficient.

Tiny distributed shared memory (TinyDSM)[21] is a reliable DDS mechanism which tries to replicate the data in some nodes along the path between the sink and the source. According to this method, each node along the query propagation path, which contains the replicated data, will answer the query directly. Therefore, the amount of query propagation traffic is load-balanced over multiple replicas of the storage node. However, as replicas are periodically updated by the source, the latter becomes the bottleneck and depletes its energy fast. Moreover, TinyDSM does not consider energy and memory constraints in its replica selection mechanism.

In this article, we focus on energy and memory efficiency as well as load balancing support. We show the efficiency of the proposed heuristic data dissemination algorithm, denoted as CoMEM, considering a cross-layer objective function defined on the basis of memory and energy efficiency with load balancing support. We then analyze other performance metrics such as delay, packet loss rate, computational complexity, and scalability of the proposed algorithm in a comparative way with respect to other data dissemination algorithms. Since CoMEM solves the proposed cross-layer optimization problem in both centralized and distributed manner, it reduces the computational complexity at the network level.

Problem statement

In Figure 1, we show the architecture of the IoT system comprising WSN, edge computing, and cloud computing.

WSN is responsible for collecting sensed data and transmitting it toward the IoT gateway to reach the cloud. Moreover, it can handle user-specific tasks assigned to IoT devices. The abovementioned tasks can either be executed by IoT devices or be offloaded to edge servers for energy efficiency and load balancing. Cloud servers are responsible for massive data storage and analysis. In this article, our focus is on the storage and retrieval process in a WSN. It is assumed that IoT devices are randomly deployed in a given region. We also consider power control at each IoT device. It is assumed that the communication channel of each IoT device can handle a variable bit rate depending on the signal interference-to-noise ratio (SINR) of each IoT device over the specified communication channel. As a result, the focus is on power control, routing and information storage, and retrieval in order to minimize memory and energy consumption.

Network model

The network model considered to formalize the optimization problem is based on the introduction of a directed graph $G = (V, E)$, where $v \in V$ denotes an IoT device and $(i,j) \in E$ specifies the link between IoT devices i and j. Moreover, μ_i^r and ε_i^r are the available

Table 1. Network model parameters and decision variables.

Parameter	Description	Parameter	Description
V	Set of IoT devices	E	Set of wireless links between IoT devices
$l_{i,j}$	Link between IoT device *i* and *j*	μ_i^r	The remaining memory of IoT device *i*
ε_i^r	The remaining residual energy of IoT device *i*	μ_i^{cap}	The memory capacity of IoT device *i*
ε_i^{cap}	The energy capacity of IoT device *i*	μ_i^{usage}	The memory usage of IoT device *i*
ε_i^{sense}	The energy usage of IoT device *i* for sensing	$\mu_i^{storage}$	The memory usage of IoT device *i* for storage
$\varepsilon_i^{forwarding}$	The energy usage of IoT device *i* for data forwarding	$\mu_i^{forwarding}$	The memory usage of IoT device *i* for data forwarding
ε_i^{cons}	The energy consumption of IoT device *i*	ε_i^{query}	The energy consumption of IoT device *i* for query and response
β	Sample rate	θ	Query rate
η	Sample size	π	Packet size
$C_{i,j}$	Capacity of link between IoT device *i* and *j*	ζ	Set of zones
T	Set of targets	ET	Set of event types
P^s	Sensing power	Φ_i	The load balancing performance metric of IoT device *i*
P^r	Receive power	$\gamma_{i,j}$	SINR over link *i* and *j*
t^{Sense}	Sensing time	$t^{transmit}$	Transmit time
z_1, z_2, z_3, z_4	Weighted coefficients	α_1, α_2	Scaling parameters
Decision variable	**Description**	**Decision variable**	**Description**
$y_{i,j}^{e,t}$	To show whether link between IoT device *i* and *j* is used to transmit target *t* data of event type e or not	$q_{i,j}^{e,t}$	To show whether link between IoT device *i* and *j* is used to transmit target *t* query of event type e or not
$r_{i,j}^{e,t}$	To show whether link between IoT device *i* and *j* is used to transmit target *t* response of event type e or not	$x_i^{e,t,z}$	To show whether target *t* with event type e in zone *z* is sensed by IoT device *i* or not
m_i^z	To show whether IoT device *i* is the monitor node of Zone z or not	$s_i^{e,z}$	To show whether IoT device *i* is the storage node of event type e or not.
P_i^t	Transmit power of IoT device *i*		

energy and memory of the specific IoT device i, respectively. In order to support load balancing, we define a load balancing performance metric denoted as Φ_i for IoT device i. This can be calculated through the following equation

$$\Phi_i = \frac{z_1}{\mu_i^r + \alpha_1} + \frac{z_2}{\varepsilon_i^r + \alpha_2} \tag{1}$$

where z_1 and z_2 are the weight coefficients of the load balancing performance metric Φ_i; α_1 and α_2 are used as scaling factors to avoid divisions by zero. μ_i^r can be calculated as follows

$$\mu_i^r = \mu_i^{cap} - \mu_i^{usage} \tag{2}$$

where μ_i^{cap} and μ_i^{usage} are the total memory capacity and current memory usage of IoT device i, respectively.

In order to calculate ε_i^r, the following equation can be used

$$\varepsilon_i^r = \varepsilon_i^{cap} - \varepsilon_i^{cons} \tag{3}$$

where ε_i^{cap} and ε_i^{cons} are the total energy capacity and energy consumption of IoT device i, respectively.

All the performance evaluation parameters are summarized in Table 1.

Decision variables

The list of decision variables for the optimization problem are the following.

- $y_{i,j}^{e,t}$: is set to 1 if the flow of target t with the event type e goes through the link between IoT devices i and j; otherwise, it is set to 0.

- $q_{i,j}^{e,t}$: is set to 1 if the flow of query for target t with the event type e goes through the link between IoT devices i and j; otherwise, it is set to 0.
- $r_{i,j}^{e,t}$: is set to 1 if the flow of response for target t with the event type e goes through the link between IoT devices i and j; otherwise, it is set to 0.
- $x_i^{e,t,z}$: is set to 1 if the target t in zone z with event type e is sensed by node i; otherwise, it is set to 0.
- m_i^z: is set to 1 if IoT device i is the monitor node of zone z; otherwise, it is set to 0.
- $s_i^{e,z}$: is set to 1 if IoT device i is the storage node of zone z for event type e; otherwise, it is set to 0.
- P_i^t: is a real variable corresponding to the transmit power of IoT device i.

Objective function

In order to increase memory and energy efficiency as well as load balancing support in the data dissemination mechanism of WSN, the cross-layer optimization problem is based on the following objective function

$$f : \min \sum_{i\in V} (z_3 \times \mu_i^{usage} + z_4 \times \varepsilon_i^{cons}) \times \Phi_i \tag{4}$$

Constraints

To model energy consumption, we apply a power control mechanism to evaluate the SINR. As a result, the following constraint is used to evaluate SINR between IoT devices i and j, denoted by $\gamma_{i,j}$

$$C1 : \gamma_{i,j} = \frac{P_i^t \times h_{i,j}}{\sum_{k\neq i} P_k^t \times h_{k,j} + \nu} \quad \forall i,j \in E. \tag{5}$$

It is obvious that P_i^t has an upper bound. This upper bound is denoted by $P^{\max}$ and leads to the following constraint

$$C2 : P_i^t \leqslant P^{\max} \quad \forall i \in V \tag{6}$$

We assume that the bit rate of each IoT device is variable. Therefore, we denote the channel capacity between IoT devices i and j as $C_{i,j}$. The bit rate is upper-bounded by the abovementioned capacity according to the following Shannon equation

$$C3 : C_{i,j} = w \times log(1 + \gamma_{i,j}) \quad \forall i,j \in E \tag{7}$$

where w is the bandwidth of all (wireless) links. The packet transmission time between IoT devices i and j is defined by the following equation

$$C4 : t_{i,j} = \frac{\pi \times 8}{b_{i,j}} \quad \forall i,j \in E \tag{8}$$

The amount of memory used for sensed data storage and data forwarding must be smaller than or equal to the memory capacity of the node. This leads to the following constraints

$$C5 : \mu_i^{storage} = \sum_{e\in ET} \sum_{z\in Z} s_i^{e,z} \times \eta \times \beta \quad \forall i \in V \tag{9}$$

$$\begin{aligned} C6 : \mu_i^{forwarding} = & \sum_{e\in ET} \sum_{t\in T} \sum_{i,j\in E} y_{i,j}^{e,t} \times \pi \times \beta + \\ & \sum_{e\in ET} \sum_{t\in T} \sum_{i,j\in E} (q_{i,j}^{e,t} + r_{i,j}^{e,t}) \times \pi \times \theta \quad \forall i \in V \end{aligned} \tag{10}$$

$$C7 : \mu_i^{usage} = \mu_i^{storage} + \mu_i^{forwarding} \quad \forall i \in V \tag{11}$$

$$C8 : \mu_i^{usage} \leqslant \mu_i^{cap} \quad \forall i \in V. \tag{12}$$

The amount of energy used for sensing an event and data transmission must be smaller than or equal to the energy capacity of the node. The constraints related to energy consumption of IoT device i can be summarized as follows

$$C9 : \varepsilon_i^{sense} = \sum_{e\in ET} \sum_{t\in T} \sum_{z\in Z} x_i^{e,t,z} \times P^{Sense} \times t^{sense} \quad \forall i \in V \tag{13}$$

$$\begin{aligned} C10 : \varepsilon_i^{forwarding} = (& \sum_{e\in ET} \sum_{t\in T} \sum_{i,j\in E} y_{i,j}^{e,t} \times P_i^t + \\ & \sum_{e\in ET} \sum_{t\in T} \sum_{j,i\in E} y_{j,i}^{e,t} \times P^r) \times t_{i,j} \quad \forall i \in V \end{aligned} \tag{14}$$

$$\begin{aligned} C11 : \varepsilon_i^{Query} = (& \sum_{e\in ET} \sum_{t\in T} \sum_{i,j\in E} (q_{i,j}^{e,t} + r_{i,j}^{e,t}) \times P_i^t + \\ & \sum_{e\in ET} \sum_{t\in T} \sum_{j,i\in E} (q_{j,i}^{e,t} + r_{j,i}^{e,t}) \times P^r) \times t_{i,j} \quad \forall i \in V \end{aligned} \tag{15}$$

$$C12 : \varepsilon_i^{cons} = \varepsilon_i^{sense} + \varepsilon_i^{forwarding} + \varepsilon_i^{query} \quad \forall i \in V \tag{16}$$

$$C13 : \varepsilon_i^{cons} \leqslant \varepsilon_i^{cap} \quad \forall i \in V \tag{17}$$

The sum of the bit rates of the flows over any link must be smaller than or equal to the capacity of that link. This is represented by the following constraint

$$\begin{aligned} C14 : (& \sum_{e\in ET} \sum_{t\in T} y_{i,j}^{e,t} \times \beta + \sum_{e\in ET} \sum_{t\in T} (q_{i,j}^{e,t} + r_{i,j}^{e,t}) \times \theta) \times \\ & \pi \leqslant C_{i,j} \quad \forall i,j \in E. \end{aligned} \tag{18}$$

The flow conservation of the storage process from the target to the storage node is illustrated by the following constraint

$$\begin{aligned} & C_{15} : \\ & \sum_{i,j\in E} y_{i,j}^{e,t} - \sum_{j,i\in E} y_{j,i}^{e,t} = \begin{cases} -1 & \text{if } \sum_{\mathrm{e}\in \mathrm{ET}} \mathrm{s}_{\mathrm{i}}^{\mathrm{e,z}} = 1 \\ +1 & \text{if} \sum_{\mathrm{e}\in \mathrm{ET}} \sum_{\mathrm{t}\in \mathrm{T}} \mathrm{x}_{\mathrm{i}}^{\mathrm{e,t,z}} = 1 \\ 0 & \text{otherwise} \end{cases} \\ & \forall z \in Z, \forall i \in V, \forall e \in ET \end{aligned} \tag{19}$$

The flow conservation of the query process from the gateway to the storage node is illustrated by the following constraint, where G is the set of IoT gateways

$$C_{16}: \sum_{i,j \in E} q_{i,j}^{e,t} - \sum_{j,i \in E} q_{j,i}^{e,t} = \begin{cases} |\mathrm{Z}| & \text{if } \mathrm{i} \in \mathrm{G} \\ -1 & \text{if} \sum_{\mathrm{e} \in \mathrm{ET}} \mathrm{s}_{\mathrm{i}}^{\mathrm{e,z}} = 1 \\ 0 & \text{otherwise} \end{cases}$$
$$\forall z \in Z, \forall i \in V, \forall e \in ET \tag{20}$$

The flow conservation of the response process from the storage node to the gateway is represented by the following constraint

$$C_{17}: \sum_{i,j \in E} r_{i,j}^{e,t} - \sum_{j,i \in E} r_{j,i}^{e,t} = \begin{cases} -1 & \text{if } \mathrm{i} \in \mathrm{G} \\ +1 & \text{if} \sum_{\mathrm{e} \in \mathrm{ET}} \mathrm{s}_{\mathrm{i}}^{\mathrm{e,z}} = 1 \\ 0 & \text{otherwise} \end{cases}$$
$$\forall z \in Z, \forall i \in V, \forall e \in ET. \tag{21}$$

Finally, the overall optimization problem can be formulated as follows

$$\begin{aligned} &\min \sum\nolimits_{i \in V} (z_3 \times \mu_i^{usage} + z_4 \times \varepsilon_i^{cons}) \times \Phi_i \\ &s.t: C1, C2, C3, C4, C5, C6, C7, C8, C9, \\ &C10, C11, C12, C13, C14, C15, C16, C17 \\ &x_i^{e,t,z}, \ y_{i,j}^{e,t}, m_i^z, s_i^{e,z} \in \{0, 1\} \end{aligned} \tag{22}$$

CoMEM

Data storage and retrieval processes are challenging design aspects, as they are directly related to a WSN lifetime and to its topological structure. As anticipated in Section 1, DDS and DCS are the main approaches proposed in the literature for memory- and energy-efficient data storage and retrieval.

- In DDS, the source node, upon sensing a specific data, stores it locally without any communication requirement. Therefore, this approach is the most energy-efficient one in the storage process. Moreover, in order to collect the sensed data in the retrieval phase, all nodes within the network have to be queried and this typically involves the use of flooding mechanisms. Therefore, the retrieval process involves a high amount of traffic and shortens the network lifetime. Figure 2 shows the retrieval process in DDS-based methods. As illustrated in the figure, the IoT gateway sends the query for specific data to all nodes. The nodes, which contain the specific data of interest, will answer the IoT gateway, for instance, this data could refer to specific targets. Nodes and targets are placed over the monitored area randomly, that is, according to a (two-dimensional) spatially uniform distribution. It is assumed that targets are stationary and that each target is detected by the nearest sensor node at a time. Afterward, the network topology forms on the basis of the position of non-sensing nodes, sensor nodes (each detecting a target), and maximum adjacency range (which depends on the physical characteristics of the wireless transmission medium). In order to obtain average performance results, a sufficiently large number of uniform deployments are considered.
- In DCS-based methods, the sensed data is stored in a specific location determined by a GHT mechanism,[3] which maps the specific event type to a specific location. In order to route the sensed data, geometric routing algorithms, such as GPSR,[4] are used to deliver the data to the nearest node to the mapped location. We enhance this mechanism by replacing GPSR with the popular IoT routing protocol denoted as the routing protocol for low power and lossy network (RPL).[22] Although the DCS approach relatively increases the storage process traffic, the retrieval process is very efficient, as all queries are forwarded to the single storage node using the GHT-mapped location. This avoids flooding mechanisms and increases energy efficiency and scalability in the network. Figure 3 shows the storage and retrieval processes in DCS-based methods. As shown in the figure, sensors will forward their sensed data to the storage node. Therefore, the IoT gateway only sends queries to this specific node instead of flooding all nodes with queries.

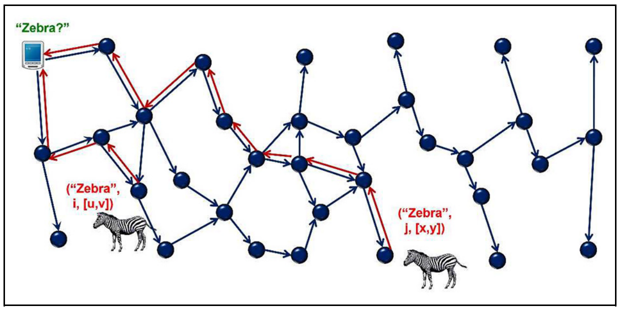

Figure 2. DDS retrieval process.

Both basic versions of DDS and DCS suffer from data loss due to node mobility and node failure. To overcome these limitations, several approaches have been proposed in the literature. All of them are based

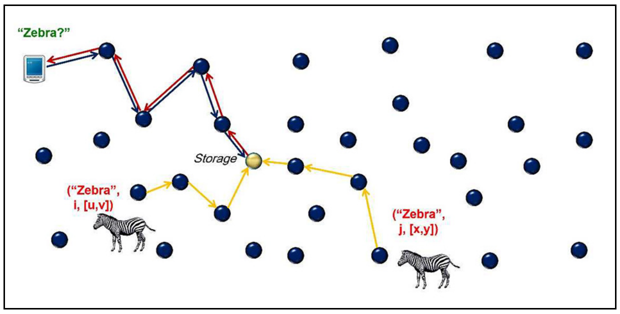

Figure 3. DCS storage and retrieval processes.

on replicating the data, that is, creating replicas over multiple storage nodes to increase fault tolerance and prevent data loss. In DDS-based approaches, due to a fully distributed storage mechanism, the most memory- or energy-available node is elected as a replica based on the local broadcasts between the neighbors of the storage node.[12,13] This mechanism involves the following challenging issues. First, a local broadcast mechanism for replica election is not energy-efficient for highly dense data dissemination. Second, a hijacking node can declare itself as the most suitable node to store the data, so that the data can easily be sniffed. Finally, if no storage node for a replica can be chosen, data loss will be unavoidable. This is due to the lack of a widespread election mechanism.

On the other hand, in DCS-based approaches, the replica is selected by the centralized GHT mechanism: the nearest node to the mapped location, determined by a hash function, will become the replica. Therefore, the energy or memory availability of the new elected replica is of no concern.

According to the method proposed by Chuang,[9] the entire network is divided into partitions called "zones." In each zone, in addition to the storage node for each event type, a new node called "monitor" is introduced to act as a gateway for both storage and retrieval processes. Whenever a data is sensed by a source node in the zone, it is forwarded to the monitor by means of a GHT hash function. The monitor then stores the summary of received data locally and forwards its details to the replica specified for the event type of the received data. This mechanism avoids the single point of failure problem, as multiple storage nodes are used by the monitor for data storage. In the retrieval process, all queries are forwarded to the monitor node. If the queries are related to locally stored data, the monitor will answer directly; otherwise, it will forward the query to the appropriate replica. However, due to the role of the monitor, all storage and retrieval traffic passes through the monitor and its neighbors. Therefore, they have to support a high amount of traffic, especially when the occurrence frequency of sensed events and the sensor spatial density are high. This depletes the energies of the monitor's neighbors and, soon, the monitor is left with no neighbor. A neighborless monitor is not accessible and, therefore, storage and query traffic will be interrupted.

We integrate the DCS data storage mechanism into the DDS one and propose a novel scheme, denoted as CoMEM, which relies on the ideas presented in literature [9,10,12] According to the CoMEM approach, we divide the network into zones on the basis of the target position. The number of zones is assumed to depend on the network size. In some approaches, that is, by Hejazi,[10] the number of zone changes on the basis of the occurrence frequency of the sensed events. Here, we define variable numbers of targets to enable sensing in several IoT devices. We assume that these targets are stationary, so that number of zones is constant. First, the monitor and storage nodes are determined by GHT within each zone. After this centralized election, a self-organized mechanism for the election of the monitor and replicas is considered. Each node within a zone informs the monitor of its remaining memory and energy. This can be carried out during the propagation of sensed data to the monitor or through dedicated packet forwarding. As shown in Table 1, ε_i^{cons} (dimension: (J) is the performance metric associated with the (current) energy consumption of IoT device i and μ_i^{usage} (dimension: (Bytes) is the performance metric associated with the (current) memory usage of IoT device i. The mentioned performance metrics are calculated and normalized by each node and sent to the monitor. The monitor stores the received data in its local performance evaluation table and calculates the "election metric," (ζ_i^r) which is the following heuristic coefficient, simultaneously taking into account energy and memory resources

$$\zeta_i^r = c_1 \times \mu_i^{usage} + c_2 \times \varepsilon_i^{cons} \tag{23}$$

$$c_1 + c_2 = 1 \tag{24}$$

where $c_1 \in [0, 1]$ and $c_2 \in [0, 1]$ are the relative weights for the energy and memory coefficients. The monitor also computes the following variation coefficient

$$\zeta_{var}^{r,i} = \zeta_{\max}^r - \zeta^r \tag{25}$$

where ζ^r is the election coefficient of the storage node and ζ_{max}^r is the maximum value of all election coefficients reported by the nodes. If $\zeta_{var}^{r,i}$ overcomes a predefined threshold, that is, the average value of all election coefficients, the election mechanism represented in Figure 4 starts.

If a node has lower energy and/or memory than at least another node in its zone, then the former will be covered by latter and its coverage number (defined in the election algorithm) will be set to 0. As a result,

```
import E(n) #array of energy coefficients
import M(n) #array of memory coefficients

def cover(n) =  ones(n)

    #evaluating the array of covers
    for i in range(1,n)
         for j in range(1,n)
             if cover(j)<>0 and E(i)<E(j)
                and M(i)<M(j) and i<>j
              #node j covers node i
              cover(i) = 0
              cover(j) = cover(j) + 1

     #finding the appropriate node for replication
     elected = 1;
     for k in range(2,n)
      if cover(k) > cover(elected)
        elected = k
      else if cover(k)=cover(elected) and E(k)>E(elected)
        elected = k
      else if cover(k)=cover(elected) and E(k)=E(elected)
          and M(k)>M(elected)
        elected = k

     return elected
```

Figure 4. Election algorithm.

nodes with coverage numbers greater than zero are more likely to become replicas. With this algorithm, local broadcasts between the neighbors of the storage nodes are avoided, and also, the new replicas are not limited to the neighbors of the storage nodes, which also need to tolerate a high amount of traffic.

This mechanism has the drawback related to the single point of failure due to the presence of the monitor. Since all the interactions for storage, query, and election processes are handled by the monitor, the monitor manages a high amount of traffic and, thus, it depletes its energy quickly. We then propose an efficient election mechanism for the monitor. This election is performed by the storage nodes. The collected ε_i^r of all nodes are periodically sent to the storage node by the monitor. The storage node then calculates the variation of ε_i^r as follows

$$\varepsilon_{var}^{r,i} = \varepsilon_{max}^r - \varepsilon_i^r \tag{26}$$

where ε_{max}^r is the maximum value of the energy performance metric reported by nodes within a zone. If $\varepsilon_{var}^{r,i}$ overcomes a threshold, that is, the average value of ε_i^r of all nodes, the node with ε_{max}^r becomes the new monitor by the storage node announcement throughout the zone. Then, the old monitor transfers its data to the new one. Although this mechanism performs a single broadcast for monitor election in the zone, it can be ignored, as it happens whenever the monitor needs to be changed. Therefore, this is more energy-efficient than the one presented in DDS approaches. Moreover, as the monitor contains all the information related to nodes' identification and placement, a hijacking node can be easily detected and, then, easily banned.

In order to evaluate the performance (according to various metrics) of the proposed distributed storage system for WSNs, various simulation approaches have been proposed with use of discrete event-based simulators (DESs) such as ns2,[23] ns3,[24] TOSSIM,[25] and EmStar.[26] In this work, we have developed a DES which is based on the event-scheduling algorithm explained by Banks.[27] In our DES, we concentrate on events and their effects on the status of the WSN during a predetermined simulation time. Figure 5 illustrates the proposed DES structure, with the three engines described below.

1. The simulation engine is the heart of our DES and implements event scheduling algorithms for non-deterministic, discrete, and dynamic WSNs. This engine has the following main components.

- Snapshot, which contains CLOCK as a base of timing system; state variables to describe the system in different points of time; future event list (FEL), which contains an ordered list of future

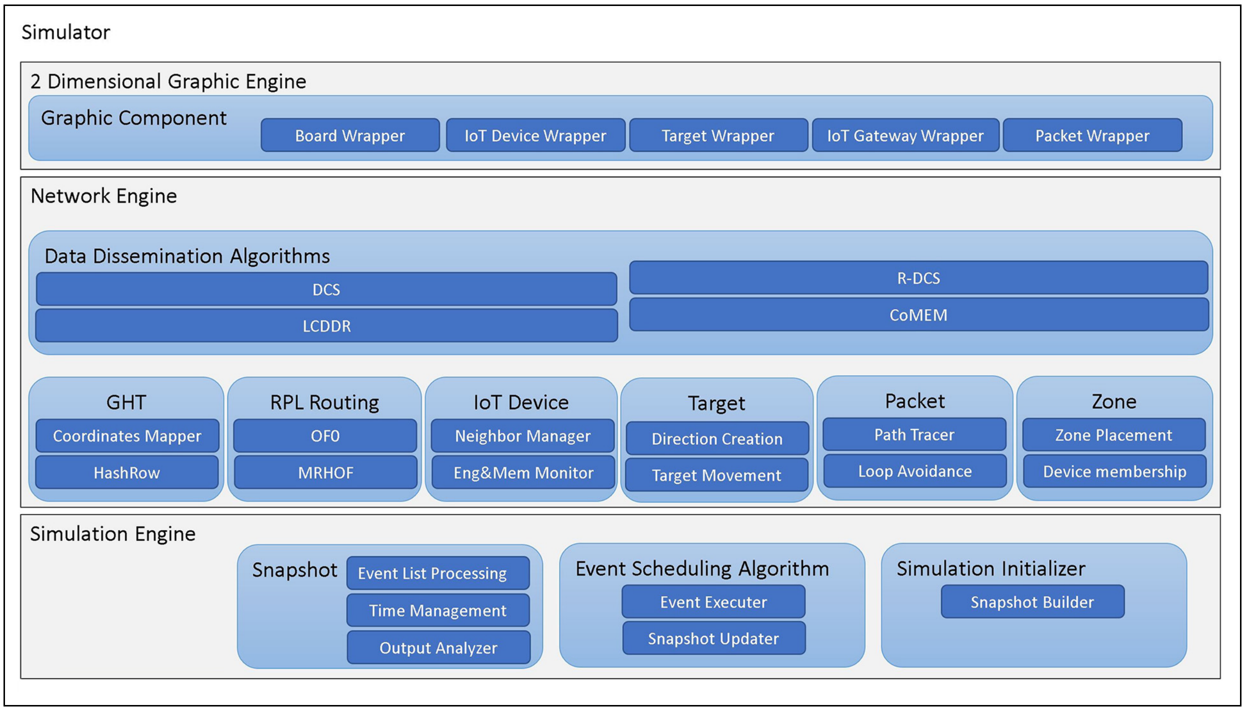

Figure 5. Developed DES structure.

events for event scheduling algorithm execution; and output parameters for evaluation and analysis. It has some functionality in list processing for inserting events into FEL and removing events from it.

- The event scheduling algorithm for event processing and updating the snapshot of the system in different points of time. The snapshot is the container of system state variables, FEL, system queues, and output counters.[27]
- Simulation initializer, which is used to build the first snapshot of the system at the CLOCK = 0.

2. The network engine handles all the interactions of the nodes within the WSN on the basis of the DCS data dissemination mechanisms. This engine includes the following elements.

- GHTs for mapping an event type to a specific location. These hash functions are appropriate to avoid traditional point-to-point routing approaches, which are used in today's Internet and, indeed, use datacentric approaches.
- RPL is a routing algorithm of choice: it can route the packet on the basis of the dynamically built directed acyclic graph (DAG) toward each storage node and the monitor. RPL decouples the routing protocol features from the routing objective function so that any arbitrary routing objective function can be applied to generate a DAG from the source to the destination. We use two basic objective functions denoted as objective function zero (OF0) and minimum rank with hysteresis objective function (MRHOF)[22] to choose the best paths in the network. OF0 and MRHOF consider the hop count and expected transmission count as the performance metric for the path selection, respectively.
- "Node" is the template for sensor node implementation in terms of energy and memory management.
- "Target" is used to define target coordination and target movements within the WSN.
- "Packet" is a container of raw information which is routed toward the storage, monitor, or gateway.
- "Zones" are the partitions of the WSN network and are used in all considered mechanisms, that is, RDCS, ARDCS, and CoMEM. Zones are associated with their boundaries and their members.

3. The two-dimensional graphical engine allows us to explicitly visualize events within the WSN whenever a specific algorithm is going to be used. It shows target movement, node placement, information sensing, dissemination of information toward the monitor or storage node, and the data retrieval process of the gateway. It contains the following packages.

- Board Wrapper: used to draw the boundary of the network and node placement.

- Node Wrapper: used to determine the wireless transmission range of the nodes and characterize transmission and reception activities.
- Target Wrapper: indicates the target position and its movement throughout the network.
- Gateway Wrapper: chooses the position of the gateway to handle the data retrieval process.
- Packet wrapper: needs to show the ensemble of the packets within the WSN.

The DES simulator proposed in this article does not build directly on the methods we used for evaluation and analysis of energy and memory efficiency. According to its layered structure, it can easily be extended to host more applications and other performance evaluations of WSNs with simple modifications. The main advantage of the abovementioned simulator is related to the independency between network functionality and simulation graphical engines. Therefore, any modification in data dissemination protocols has no effect on the abovementioned engines.

Performance results

Energy and memory efficiency are the most important evaluation factors of WSNs, as they are directly related to data loss and network lifetime. In order to evaluate the proposed heuristic algorithm, we first consider memory and energy efficiency by adding a load balancing performance metric on the basis of the cross-layer objective function defined in Section 1. Afterward, we analyze other performance metrics such as delay and data loss to show the efficiency of CoMEM with respect to other related WSN data dissemination methods. Finally, we analyze the computational complexity of each method by considering the simulation runtime (on a workstation with a 16 GByte of DDR3 RAM, Intel Core i7 CPU, and SSD hard drive).

In order to evaluate our proposed method with the metrics outlined in the previous paragraph, we assume that sensor nodes are static—the extension to mobile nodes is under investigation. Since neighboring nodes should not be deployed too close to each other, the locations of nodes are determined randomly but with a minimum inter-node distance. We also assume that the quality of the radio channel does not change dramatically in a short period of time, so that the transmission range is considered to be constant. Moreover, nodes are robust during storage and retrieval processes and do not crash in the presence of traffic bursts. Finally, as our research focuses on a cross-layer optimization problem, wireless communication at physical layers is modeled on the basis of the (IEEE 802.15.4) 6LowPAN standard, with a short transmission range and relatively high node spatial density. The extension to IEEE 802.11b/g protocols (and comparative performance analysis) will be the subject of our future work. In order to handle data transmission, we rely on power control to guarantee energy efficiency at the physical layer. More precisely, the transmission power is set, taking into account (i) the path gain over the link and (ii) the interference from neighboring nodes on the destination, to reach the highest bit rate for the corresponding SINR. Table 2 summarizes the input parameters for IEEE 802.15.4 6LowPAN.[28]

Table 2. IEEE 802.15.4 6LowPAN physical and link layer inputs.

No.	Name	Value
1	Network size	12×12 m
2	Number of nodes	30
3	Minimum distance	1.4 m
4	Number of targets	1-10
5	Node initial energy	16 KJ
6	Frequency band	2.4 GHZ
8	Packet size	127 Bytes
7	Reception energy consumption	$41\ \mu$ J
8	Node RAM	1024 Blocks
9	Sample rate	$0.5\ \frac{sample}{sec}$
10	Sample size	26 Bytes
11	Query rate	$0.1\ \frac{query}{sec}$
12	Receiver sensitivity	3.9×10^{-13}
13	Maximum transmission power	1 mW
14	Transmission gain	0.003
15	Receiver gain	0.003
16	Path loss	2
17	Noise	10^{-13}
18	Channel bandwidth	100 KHZ
19	z_1, z_2, z_3, z_4	0.5
20	α_1, α_2	10^{-5}
21	TH_1, TH_2	0.2
22	Simulation time	80 sec

We can divide the metrics in Table 2 into the following three groups:

- Parameters depending on the application: network size, number of nodes, minimum distance between neighboring nodes, and number of targets.
- Parameters based on the node specification: node initial energy, which is evaluated on the basis of the initial energy of an alkaline battery;[29] receive power computed as the product between the node voltage, the current in receiving mode, and the receiving time, that is, $3\text{ V} \times 11.8\text{ mA} \times 4\text{ ms} = 141\ \mu\text{ J}$; transmission range calculated according to the Friis formula[30] with the path loss exponent set to 2 (i.e. line-of-sight communications are considered); and number of memory slots computed by dividing the available RAM for data storage in the sensor nodes by the PDU dimension.

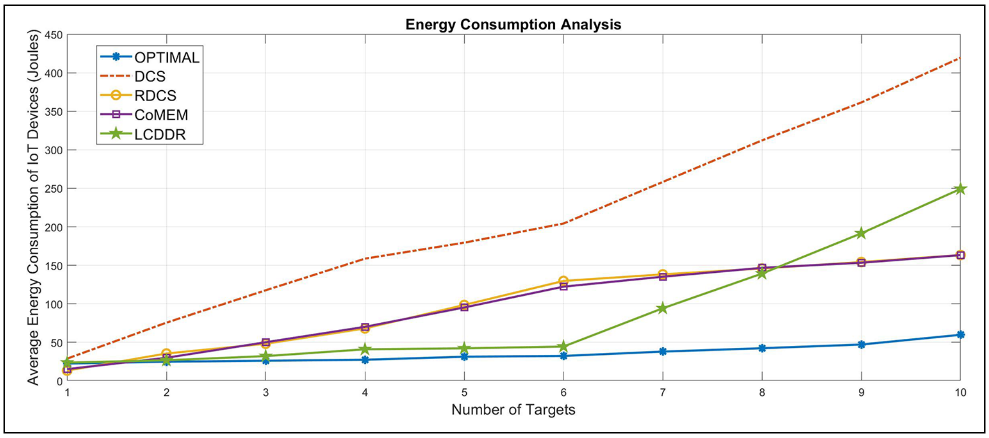

Figure 6. Energy consumption as a function of the number of targets. The performance of CoMeM is compared with those of other relevant algorithms and with that of the optimal solution.

- Parameters based on the network characteristics: frequency band, bit rate, and transmission delay.

In order to evaluate the performance of the storage and retrieval processes, we focus on the energy consumption of a single packet transmission and the total amount of packets which are sent in each process. On the basis of the data provided by Atmel,[28] the energy consumption per packet can be calculated according to the following equation

$$\varepsilon^{packet} = (\varepsilon^{transmit} + \varepsilon^{receive}) \times (d^P + t^{transmit}) \tag{27}$$

where d^P is the propagation delay and $t^{transmit}$ is the transmission time. Moreover, according to the IEEE 802.15.4 standard, the minimum payload size is 81 Bytes. Since 40 Bytes are used for the IPv6 header and 7 Bytes belongs to the UDP header, 33 Bytes are left for data. The use of a JSON data structure consumes 7 more Bytes, so that the space left for the key/value pair (the "data") is 26 Bytes. We evaluate each data storage algorithm by inserting different numbers of targets in the simulated WSN. The number of targets is between 1 and 10 in order to force different levels of storage and retrieval traffic in a short period of time (80 sec).

Before commenting on the specific performance results presented in the following figures, we make the following preliminary observations.

- Obviously, the *energy consumption due to storage* is minimized when using LCDDR, as data is stored locally without any transmission. Therefore, the traffic due to storage is only limited to the election mechanism. R-DCS ranks second. With this algorithm, monitors are also used in the storage process: this creates more traffic and makes R-DCS vulnerable to monitor failure. CoMEM is the method with monitor and storage nodes collaborating in the storage process and uses two election mechanisms for monitor and storage node election. Therefore, the traffic of CoMEM storage is relatively higher than R-DCS. Finally, the basic DCS has the worst performance in terms of storage-related traffic, as it only relies on storage nodes and no fault tolerance mechanism is implemented: therefore, it does not contain any election mechanism for storage and this also increases the traffic due to node failure around the storage nodes.
- In terms of *energy consumption due to the query/response process*, the most efficient result belongs to the basic DCS because only storage nodes are involved in queries and responses. However, this mechanism suffers from a lack of load balancing consideration which increases traffic in the case of node failure around the storage nodes. The second position belongs to R-DCS and CoMEM due to the role of the monitor as a gateway in the retrieval process. R-DCS also suffers from the single point of failure risk related to the energy depletion of the monitor and a lack of load balancing approach. Finally, the LCDDR mechanism incurs the highest amount of traffic due to its inefficient retrieval process: this increases traffic bursts in the retrieval process and decreases the network lifetime. The average energy consumption of nodes for the represented methods is shown in Figure 6 as the function of the number of targets. As shown in the figure, CoMEM and R-DCS have the best energy efficiency with higher numbers of targets and their results are closer to the optimal solution.

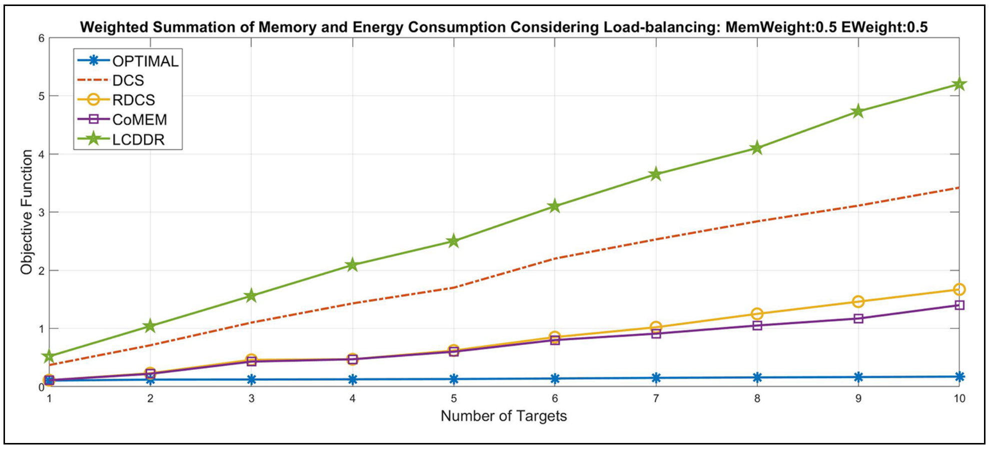

Figure 7. Objective function evaluated as a function of the number of targets. The performance of CoMeM is compared with those of other relevant algorithms and with that of the optimal solution.

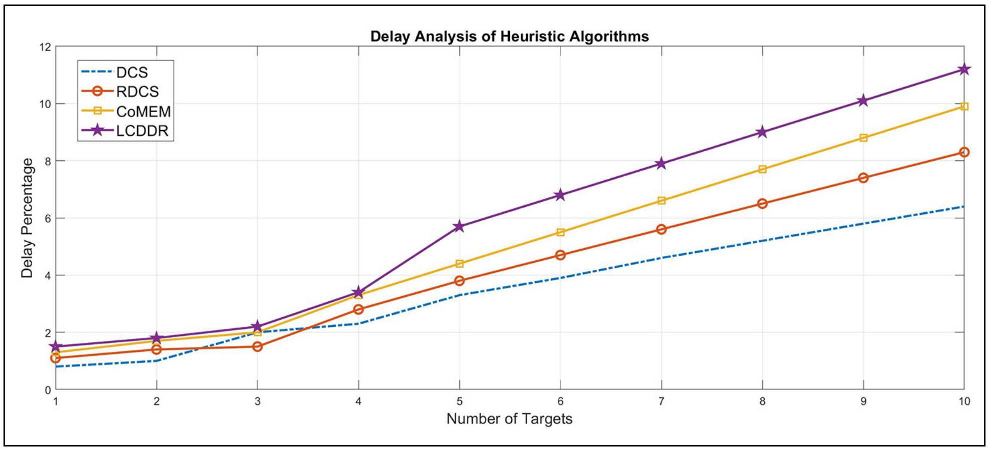

Figure 8. Delay as a function of the number of targets. The performance of CoMeM is compared with those of other relevant algorithms.

The performance results in terms of the value of the objective function evaluation are shown in Figure 7. It can be easily observed that CoMEM has the best performance on energy and memory efficiency as well as load balancing support, making it the closest to the optimal solution. The main reason for this behavior is the collaborative distributed election mechanism which brings load balancing to memory usage and energy consumption in IoT devices.

Since CoMEM has the best performance related to energy and memory efficiency as well as load balancing support, we now consider other performance metrics, namely, delay, data-loss ratio, and computation complexity. In Figure 8, the delay incured by the data dissemination flow from the sensing IoT device to the IoT gateway access is shown as a function of the number of targets. Since CoMEM and R-DCS add a monitor to the storage and retrieval process, the amount of delay they incur is higher than that of basic DCS. Therefore, the best performance for delay evaluation belongs to DCS.

From the data loss point of view, CoMEM has the best performance because of its distributed collaborative election mechanism. It also implements replication to avoid single point of failure problem. This is confirmed by the results in Figure 9, where data loss percentage is shown as a function of the number of targets. Finally, the computational complexities of the considered algorithms, in terms of numbers of mathematical operations (additions and multiplications), are compared in Figure 10. It can be easily observed that the computation complexity of CoMEM, due to the distributed collaborative election mechanism, is very close to that of R-DCS.

In order to observe scalability, we evaluate the objective function defined in equation (4) as the function of

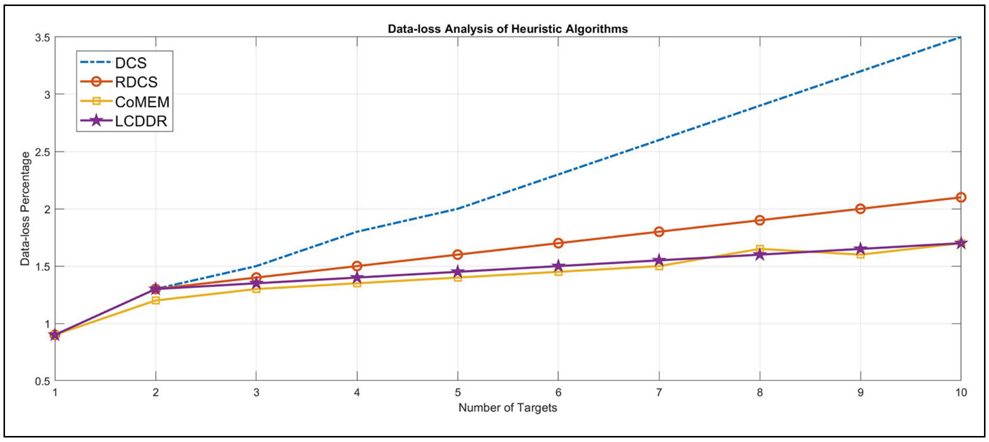

Figure 9. Data loss percentage as a function of the number of targets. The performance of CoMeM is compared with those of other relevant algorithms.

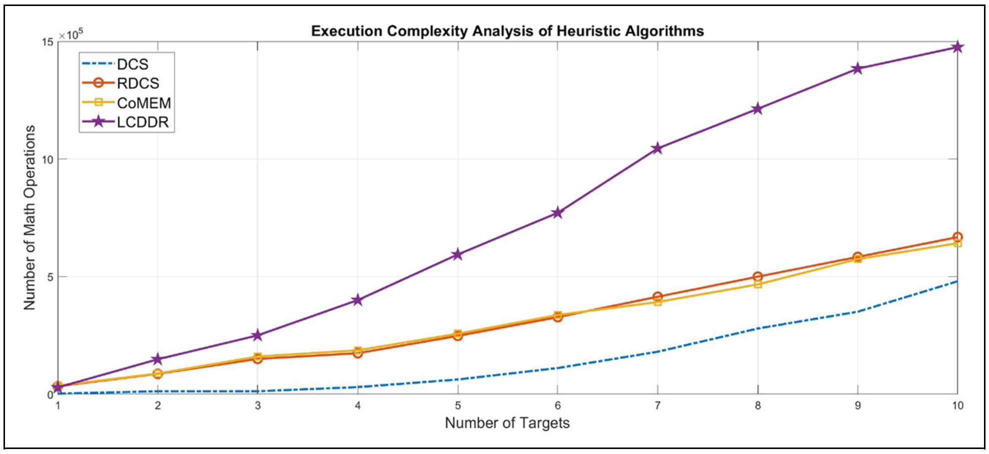

Figure 10. Computational complexity as a function of the number of targets. The performance of CoMeM is compared with those of other relevant algorithms.

different deployed IoT devices. The results are shown in Figure 11. As shown in the abovementioned figure, CoMEM has the best performance due to energy and memory efficiency as well as load balancing support with different IoT devices.

In Table 3, we summarize the performance of the considered algorithms in a comparative way, considering the performance indicators investigated in Figures 6–11. In particular, for each performance indicator, we show the ranking (note that if the performance of two algorithms is basically the same, as is the case for CoMEM and RDCS in two cases, we consider the same ranking) based on the performance results obtained in Figures 6–11 for the largest possible number of targets (namely, 10). In the last row of Table 3, we show the average ranking, based on the corresponding arithmetic average of the per-indicator ranking values, between parentheses—obviously, the smaller the arithmetic average, the higher the ranking. It can be observed that CoMEM is, on average, the best algorithm. In particular, CoMEM ranks first with respect to four indicators: energy consumption, objective function (i.e. combined energy and memory efficiency, together with load balancing), data loss, and scalability. This high performance level, aligned with the starting design goals, comes at the price of increased delay (CoMEM ranks third) and computational complexity (CoMEM ranks second). This is to be expected, as CoMEM was not designed to minimize the delay but, rather, energy and memory efficiency together with effective load balancing; in order to guarantee an overall best trade-off (between energy and memory efficiency, effective load balancing, and scalability), it is natural that the computational complexity increases with respect to that of

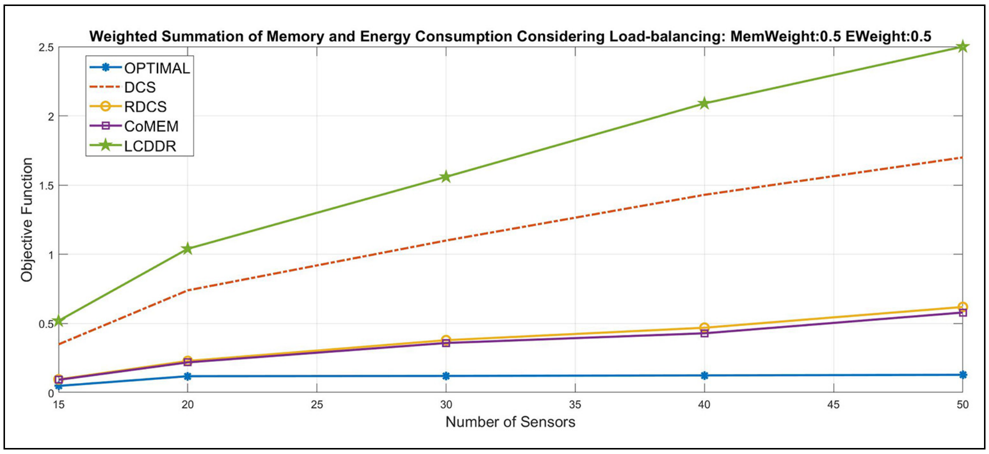

Figure 11. Scalability (in terms of the objective function value) as a function of the number of targets. The performance of CoMeM is compared with those of other relevant algorithms and with that of the optimal solution.

Table 3. Ranking (per performance indicator and average) among the four considered algorithms. In the average row, the corresponding arithmetic average of the per-indicator ranking values is shown between parentheses.

	DCS	RDCS	CoMEM	LCDDR
Energy consumption	3	1	1	2
Objective function	3	2	1	4
Delay	1	2	3	4
Data loss	3	2	1	1
Computational complexity	1	2	2	3
Scalability	2	1	1	3
Average	3 (2.16)	2 (1.66)	1 (1.5)	4 (2.83)

DCS: data-centric storage; CoMEM: Collaborative Memory and Energy Management; LCDDR: low complexity distributed data replication.

simple DCS. Neverthless, one can observe that the delay and computational complexity "penalties" incurred by CoMEM are very limited.

Conclusions

In this article, we have introduced a cross-layer optimization problem which accounts for both energy and memory efficiency of data dissemination in IoT networks as well as load balancing support. This optimization problem is MINLP and has been solved using a genetic algorithm. Afterward, we have proposed a heuristic algorithm, denoted as CoMEM, to overcome the abovementioned optimization problem. Finally, we show that our proposed heuristic algorithm outperforms other related data dissemination methods considering packet loss, delay, and computational complexity as the performance metrics.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Pooya Hejazi https://orcid.org/0000-0001-5877-5368
Gianluigi Ferrari https://orcid.org/0000-0001-6688-0934

References

1. Shafi M, Molisch AF, Smith PJ, et al. G: A tutorial overview of standards, trials, challenges, deployment, and practice. *IEEE J Select Area Commun* 2017; 35(6): 1201–1221.
2. Yick J, Mukherjee B and Ghosal D. Wireless sensor network survey. *Comput Netw* 2008; 52(12): 2292–2330.
3. Ratnasamy S, Karp B, Yin L, et al. GHT: a geographic hash table for data-centric storage. In: *Proceedings of the 1st ACM international workshop on wireless sensor networks and applications*, Atlanta, GA, 28 September 2002, pp.78–87. New York: ACM.
4. Karp B and Kung HT. GPSR: Greedy perimeter stateless routing for wireless networks. In: *Proceedings of the 6th annual international conference on mobile computing and networking*, Boston, MA, 6–11 August 2000, pp. 243–254. New York: ACM.

5. Aly M, Pruhs K and Chrysanthis PK. KDDCS: a load-balanced in-network data-centric storage scheme for sensor networks. In: *Proceedings of the 15th ACM international conference on information and knowledge management*, Arlington, MA, 6 November 2006, pp.317–326. New York: ACM.
6. Lai Y, Wang Y and Chen H. Energy-efficient robust data-centric storage in wireless sensor networks. In: *International conference on wireless communications, networking and mobile computing Wicom*, Shanghai, China, 21–25 September 2007, pp.2735–2738. New York: IEEE.
7. Cuevas A, Uruena M, De Veciana G, et al. Dynamic data-centric storage for long-term storage in wireless sensor and actor networks. *Wirel Netw* 2014; 20(1): 141–153.
8. Ahmed K and Gregory MA. Distributed efficient similarity search mechanism in wireless sensor networks. *Sensors* 2015; 15(3): 5474–5503.
9. Chuang AG. Resilient data-centric storage in wireless sensor networks. *IEEE Distrib Syst* 2003; 4(11): 1–12.
10. Hejazi P. An adaptive hybrid schema for data-centric storage in wireless sensor networks. *Int J Distrib Sens Netw* 2016; 12(10): 6665.
11. Dudkowski D, Marron PJ and Rothermel K. An efficient resilience mechanism for data centric storage in mobile ad hoc networks. In: *International conference on mobile data management (MDM'06)*, Nara, Japan, 10–12 May 2006. New York: IEEE.
12. Gonizzi P, Ferrari G, Gay V, et al. Data dissemination scheme for distributed storage for IoT observation systems at large scale. *Inform Fusion* 2015; 22: 16–25.
13. Sajjadian Amiri S, Tabatabaee Malazi H and Ahmadi M. Memory efficient routing using bloom filters in large scale sensor networks. *Wirel Pers Commun* 2015; 86(3): 1221–1240.
14. Xu H, Li Y, Collinge WO, et al. Improving efficiency of wireless sensor networks through lightweight in-memory compression. In: *Sixth International green computing conference and sustainable computing conference (IGSC)*, Las Vegas, NV, 14–16 December 2015, pp.1–8. New York: IEEE.
15. Chouikhi S, El Korbi I, Ghamri-Doudane Y, et al. A survey on fault tolerance in small and large scale wireless sensor networks. *Comput Commun* 2015; 69: 22–37.
16. Behera TM, Mohapatra SK, Samal UC, et al. Residual energy based cluster-head selection in WSNs for IoT application. *IEEE Internet Things J* 2019; 6: 5132–5139.
17. John A, Rajput A and Babu KV. Dynamic cluster-head selection in wireless sensor network for Internet of Things applications. In: *International conference on innovations in electrical, electronics, instrumentation and media technology (ICEEIMT)*, Coimbatore, India, 3–4 February 2017, pp. 45–48. New York: IEEE.
18. Wang Z, Qin X and Liu B. An energy-efficient clustering routing algorithm for WSN-assisted IoT. In: *IEEE wireless communications and networking conference (WCNC)*, Barcelona, 15–18 April 2018, pp.1–6. New York: IEEE.
19. Banerjee J, Mitra SK, Ghosh P, et al. Memory based message efficient clustering (MMEC) for enhancement of lifetime in wireless sensor networks using a node deployment protocol. In: *Proceedings of the 2011 international conference on communication, computing and security*, Rourkela, Odisha, India, February 2011, pp.71–76. New York: ACM.
20. Zhou H, Guan X and Wu C. Reliable transport with memory consideration in wireless sensor networks. In: *IEEE international conference on communications*, Beijing, China, 19–23 May 2008, pp.2819–2824. New York: IEEE.
21. Piotrowski K, Langendoerfer P and Peter S. tinyDSM: A highly reliable cooperative data storage for wireless sensor networks. In: *International symposium on collaborative technologies and systems*, Baltimore, MD, 18–22 May 2009, pp.225–232. New York: IEEE.
22. Kharrufa H, Al-Kashoash HA and Kemp AH. RPL-based routing protocols in IoT applications: a review. *IEEE Sens J* 2019; 19(15): 5952–5967.
23. Isi.edu. The Network Simulator—ns-2, 2018, http://www.isi.edu/nsnam/ns/ (accessed 21 August 2018).
24. Nsnam.org. ns-3, 2018, https://www.nsnam.org/ (accessed 21 August 2018).
25. Tinyos.stanford.edu. TOSSIM—TinyOS Wiki, 2018, http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM (accessed 21 August 2018).
26. Usenix.org. EmStar: a Software Environment for Developing and Deploying Wireless Sensor Networks, 2018, https://www.usenix.org/legacy/event/usenix04/tech/general/full_papers/girod/girod_html/eu.html (accessed 21 August 2018).
27. Banks J, Carson II JS, Nelson BL, et al. *Discrete event system simulation*. 5th ed. Pearson Education, 2010.
28. Atmel.com, 2018, http://www.atmel.com/images/Atmel-8351-MCU_Wireless-AT86RF233_Datasheet.pdf (accessed 21 August 2018).
29. AAbattery. AA battery, https://en.wikipedia.org/wiki/AAbattery (accessed 21 August 2018).
30. Friis. Friis transmission equation, 2018, https://en.wikipedia.org/wiki/Friis_transmission_equation (accessed 21 August 2018).