Wireless Netw
DOI 10.1007/s11276-013-0598-5

Dynamic Data-Centric Storage for long-term storage in Wireless Sensor and Actor Networks

Ángel Cuevas · Manuel Urueña · Gustavo de Veciana · Rubén Cuevas · Noël Crespi

Abstract Data-Centric Storage (DCS) appears as a novel information storage and delivery mechanism for Wireless Sensor and Actor Networks in which a rendezvous node (home node) is selected to store and serve all the information of a particular application. However, DCS was not designed to provide long-term data availability. In this paper we present a Dynamic DCS solution to enable a long-term storage system. Dynamic DCS proposes to periodically change home nodes over the time based on periods of fixed duration called epochs. This makes it possible to perform temporal queries to previous home nodes in order to retrieve information from the past. We evaluate our proposal using extensive simulations, and reveal that Dynamic DCS makes sensor events available at least 85 % of the maximum lifetime provided by an optimal (but non practical) solution. Finally, we show that Dynamic DCS could easily adapt its storage performance to the requirements of an application by just tuning the epoch duration.

Á. Cuevas (✉) · M. Urueña · R. Cuevas
Universidad Carlos III de Madrid, Madrid, Spain
e-mail: acrumin@it.uc3m.es

M. Urueña
e-mail: muruenya@it.uc3m.es

R. Cuevas
e-mail: rcuevas@it.uc3m.es

G. de Veciana
University of Texas at Austin, Austin, TX, USA
e-mail: gustavo@ece.utexas.edu

N. Crespi
Institut Mines-Télécom, Télécom SudParis, Evry, France
e-mail: noel.crespi@it-sudparis.eu

Published online: 14 May 2013

Keywords Wireless Sensor and Actor Network (WSAN) · Data-Centric Storage (DCS) · Data availability · Epoch

1 Introduction

Data-Centric Storage (DCS) [16] was introduced as a novel distributed information storage and delivery mechanism in which a node is selected as a rendezvous point to store all events of a particular application. For instance, one node in the network stores all temperature events, another node stores humidity events, another one fire events, etc. Then nodes that produce the information related to a particular application (called producer nodes) compute the rendezvous node (also called home node) for that application, and store its information into it. In turn, a node that wants to retrieve information from that application (called consumer node) only needs to query the associated home node, which replies with the application's stored data. The efficiency of DCS comes from the fact that each node is able to locally compute the home node for any application, and then, by using the underlying routing layer, can store/retrieve application data on/from that home node.

Data-Centric Storage appears as a suitable information and delivery mechanism for Wireless Sensor and Actor Networks (WSANs) [2, 13]. In this novel type of network a new player, the actor or actuator node, performs some action based on the information retrieved from other sensors in the network. Actor nodes go from very simple devices that turn on/off a LED, to very sophisticated mobile robots that can obtain samples of a terrain. In WSANs all actor nodes are information consumers that query sensor nodes in order to retrieve network information. Therefore, DCS allows actors (consumer nodes) to

carry out their activity by easily retrieving from the home node all the information that sensors (producer nodes) of a particular application have generated. It must be noted that both sensor and actor nodes can act as home node (i.e. store data event) as long as they belong to the WSANs).

In addition, many applications may require to have long-term access to past data, i.e. actor nodes may need some historical information to decide whether they need to actuate or not. However, the original DCS does not provide long-term storage for sensor events since it does not distribute the storage load among network nodes. For instance, in a WSAN with 100 nodes and 5 applications, only 5 nodes will use their memory to store data events. In addition, since all the traffic of an application is stored in a single node, old events will quickly be replaced by new ones, reducing sensor events lifetime. This problem is especially relevant for applications that generate a continuous stream of events (e.g. measuring the temperature every second).

In this paper we propose a novel Dynamic DCS solution to enable a long-term storage system that allows information consumers to easily access historical sensor events. Towards this end, we utilize a system similar to that proposed in [4] in which several home nodes (or replicas) are selected at random and only for a limited amount of time. After that time a new set of home nodes is selected for the next period. In order to change the home nodes over time we divide the time into periods of a fixed duration called *epochs*. During an epoch all application events are stored in the selected home nodes. Therefore, a node will only overwrite old events after being chosen as a replica multiple times in several different epochs, thus extending sensor event availability. Since consumers are able to know all home nodes at any given epoch, an event is considered to be available as long as it is accessible in at least one of the home nodes that initially stored it.

Considering the proposed solution, on one hand, it seems clear that changing the home node (or home nodes) over the time will suppose a significant extension on event availability. If the home node selected for a particular application never changes, it quickly saturates its memory and needs to remove old events in order to store new ones. This effect is especially significant for high load applications. This paper demonstrates that our solution extends events availability more than 30 times as compared to those static DCS proposals.

On the other hand, it is not clear whether it is better to use only one home node per application, or to set up several replicas (home nodes) in which the events are also stored. In this paper we analyze and determine what is the optimal number of replicas that should be used in a Dynamic DCS system in order to extend sensor events lifetime. The main conclusion is that to maximize event availability in the medium and long-term the best solution is to use a single home node per application.

Finally, we evaluate the performance of Dynamic DCS long-term storage proposal and compare it with three other distributed storage mechanisms: (1) A static DCS mechanism with a single home node [16]. (2) Local Storage in which each sensor locally stores the information it generates, and (3) an ideal Round Robin Storage in which all network storage capacity is used before deleting an event.

We validate Dynamic DCS as a long-term storage system because (if a reasonable epoch duration is selected) it presents a median event lifetime longer than 85 % of the optimal median lifetime offered by the Round Robin solution. In addition, we demonstrate that our solution is highly flexible and can be easily adapted to applications' requirements by just tuning the epoch duration. Then the performance of our solution ranges between a Round Robin Storage when we select very short epochs, and a DCS static behaviour in case of using long epochs.

The remainder of this paper is structured as follows: Sect. 2 describes related work from the literature. In Sect. 3 we introduce Dynamic DCS, and in Sect. 4 we present a model to provide the optimal number of replicas to maximize sensor events lifetime. Section 5 presents an evaluation of our proposal, and we conclude the paper in Sect. 6.

2 Related work

The original Data-Centric Storage proposal [16] already compared DCS to other storage mechanisms such as Local Storage and External Storage. In particular, the performance of DCS was measured in terms of reliability, number of messages sent and received per node and nodes mobility. However, they did not study the time that sensor events are available in the network, which is the scope of our paper.

Some works [1, 4, 6, 7, 15] demonstrate that using several home nodes (replicas) for a particular application reduces the overall network traffic, which leads to a lower energy consumption in the network. Some of those studies [7, 15] propose allocating replica nodes following a grid-structure, while the authors in [1] propose to use a uniform replica deployment. Finally, we demonstrate in [4] that allocating the replicas at random reduces the overall network traffic compared to previous mechanisms, while being the simplest replication algorithm.

In [4] we also claim that static DCS is an unfair system in terms of energy consumption distribution, independently of the number of replicas used. Nodes selected as replicas and their surrounding neighbours will naturally expend more energy than other nodes within the network. We

address this issue and propose to change the replicas over the time in order to balance the energy consumption among all network nodes. Our Dynamic DCS proposal thus utilizes both elements: multiple replicas that change over time to enable long-term storage of sensor events. The results demonstrate that distributing the energy consumption has a huge impact on extending the network lifetime.

There are two studies that propose rotating the home node in order to balance the storage task. In the first one [9], the authors propose to divide the network in a grid. Each application is assigned to one of the grid cells, so that one node inside of the selected cell will be the home node for that application. After a pre-defined time a node in the cell with the immediate lower ID is selected as the new home node for that application, and so on. The second proposal [10] also divides the network in a grid and assigns each application to one cell. However, in this case the home node rotation for an application occurs within the same cell. Then whenever the initial home node reaches a storage threshold, another node within the same cell overtakes the home node role. Although both proposals focus on alleviating the storage congestion problem of using a single static home node, they do not discuss anything regarding the time that sensor events are available under these proposed solutions.

We did not find any previous work in the field of "pure" DCS networks that deeply studies event availability. However, there exist a research line on erasure coding that is similar in spirit and present a broader application field further than WSNs and WSANs [3, 5].

3 Dynamic Data-Centric Storage

In this Section we present our dynamic Data-Centric Storage proposal for the long-term storage of sensor events. Our proposal is a DCS system in which each application has one or more home nodes that change over the time.

We first outline the different roles that a node could perform in an application, then we briefly describe the original DCS proposal, and finally we introduce the proposed solution.

A node in a DCS network could play one (or more) of the following roles:

- *Producer nodes*: Those nodes that generate events for a particular application. Generally, producers are sensor nodes. A producer node could generate from very basic events, such as a temperature sample, until very complex ones, such as an intruder detection. Furthermore, producer nodes are usually equipped with quite a few sensors so in many cases they produce events streams that require a considerable amount of memory. We must notice that nowadays there is cheap flash memories whose capacity can reach hundred of Megabytes that can be used in scenarios with applications requiring to store quite a lot of information. However, in some other cases we may like to use very cheap sensor nodes (e.g. some few dollars) that will be equipped with small memories, which will be able to deal only with a moderate amount of information.
- *Consumer nodes*: Those nodes consuming events of a particular application. An obvious example of a consumer node in a traditional WSN is the base station, which is the one querying the network in order to retrieve the events. In this paper we focus on WSANs where actor nodes act as consumer nodes. Other possible deployments eliminate the presence of a base station and the information is retrieved from time to time by physically accessing the place where the network is deployed and collecting the information using some kind of mobile device such as a PDA or laptop. Thus, it is worth noting that the nature and number of consumers in a network vary considerably according to different scenarios.We notice that in some cases the same physical node can produce events and send consumption queries at the same time, thus playing the role of producer (i.e. sensor) and consumer (i.e. actor) in the network.
- *Home nodes or Replicas*: These nodes are the ones selected to store and deliver the information for a particular application.

3.1 DCS overview

Shenker et al. [16] introduced the concept of Data-Centric Storage in which a node, called home node, is selected to store and serve all the events of a particular application. Therefore, producer nodes store the events they generate in the home node while consumer nodes query the home node to retrieve that application events. It must be noted that the original DCS proposal relies on geographical information and is known as Geographical Hash Table (GHT). All the nodes within the network know their own geographical position and the network dimension. GHT relies on a geographical routing service, Greedy Perimeter Stateless Routing (GPSR) [8], to forward messages from source nodes to destination coordinates. GPSR is a very efficient routing protocol in terms of energy consumption for WSANs. Furthermore, the authors propose using a single home node per application. The location of the home node (HN) is computed using a hash function over the application's name, i.e. $HN\_location = hash('APP')$. However, it is very unlikely that the output coordinates match any of the actual nodes' location. Then, the closest node to the

hash output location is the one selected as the home node for that application.

All producers and consumers of a particular application obtain the same home node output location because all of them use the same hash function and know the application name. The authors claim that by just using GPSR producer events and consumers queries forwarded towards the hash output coordinates will ultimately reach the home node. Therefore, this novel storage and delivery mechanism does not require that consumers and producers of a particular application have prior knowledge of each other, since by just knowing the application name they can participate in the application data flow.

3.2 Dynamic DCS for long-term storage

Although the original DCS solution is a very suitable storage mechanism for WSANs, it was not designed to offer long-term event storage. All the events of a particular application are stored in a single node, the home node. Therefore, new events can quickly overwrite old ones since all application traffic is concentrated into a single node. Hence, the original DCS does not use the network storage resources efficiently. For instance in a network of 100 nodes and 10 applications, only 10 nodes (each application's home node) use their memory to store events, and thus 90 % of the network storage resources are wasted.

In this paper we extend the original DCS proposal by using several home nodes (replicas) placed at random that change over time following the spirit of our previous proposal [4]. Our goal is to efficiently use the storage resources of the network to maximize the time window in which sensor events are available. In order to change the replicas over the time, the time is divided into periods of the same duration called *epochs*. Then, a new set of replicas is selected at the beginning of each epoch. It is assumed that all nodes in the network know the epoch duration as well as the number of replicas for those applications in which they want to participate.

Under the proposed schema, new input parameters, the number of replicas and an epoch ID, are required by the hash function to allow consumers and producers to compute who are/were the replication nodes in a particular epoch. Thus, the new hash function is: $HN\_location = hash('APP' \oplus epochID \oplus i), \forall i \in [1, r]$, where r represents the number of replicas deployed in the field for the application $'APP'$.

Producer nodes store their events in the closest replica, which in turns replicates the information in all the remaining replicas using a minimum spanning tree as proposed in [4]. Therefore, consumer queries only need to access one of those replicas in order to retrieve all the events of an application generated in a particular epoch.

When the current epoch ends a new set of replicas is chosen. The replicas in the previous epoch do not transfer any data to the new ones, unlike [4] where replicas in epoch i transfer the stored events to replicas in epoch $i + 1$ and delete them. Then, any node in the network selected as a replica stores events until its memory is full. From that moment it deletes the oldest information to store new sensor events. Hence, it is very likely that a particular event stored by a node would be only overwritten after that node has been selected as replica several times.

In addition, the defined system allows consumers to realize temporal queries in order to retrieve historical data. The fact of using a fixed epoch duration facilitates to compute the set of replicas that stored events of a particular application at some particular moment in the past. To accomplish this task, consumers only need to know the application's name (e.g. temperature), from which time they want the information (e.g. 3 h ago), the epoch duration for the application of interest (e.g. 1 h), and the number of replicas being used by that application. Then, once the right epoch has been identified, consumer nodes just need to send unicast queries to any of the home nodes in that epoch. Therefore, historical information will be available in the network until it is overwritten in all home nodes of that epoch.

4 Sensor events lifetime analysis

In this section we analyze different distributed storage mechanism in terms of sensor event availability. We first present a very simple storage mechanism such as Local Storage. Next, we describe an ideal Round Robin Storage mechanism. Finally, we show how our Dynamic DCS proposal can be utilized for long-term storage.

4.1 Local storage

This is the simplest storage mechanism in which each node locally stores the events it generates. The energy consumption to store those events is negligible since the radio transceiver is not used. However, there is no way to know where the information of a particular application in a particular time-window is stored. Therefore, in order to retrieve historical information, consumer nodes need to flood the network. This converts Local Storage in a highly inefficient storage system in terms of energy in most cases when compared to Dynamic DCS. Our solution avoids the need for flooding since a consumer node can easily determine the home nodes storing the desired information. However, it must be noted that in those cases where producers event rate is considerably larger than consumers query rate, Local Storage appears as an efficient solution in terms of energy.

In order to estimate the events lifetime, we assume a network where N nodes with a storage capacity of s events are deployed in a network with k applications $(APP_1, APP_2, \ldots, APP_k)$. In addition, we consider all events to be of the same size. Then, the probability that a node i generates an event of an application j in each time unit is $p_{e_{i,j}}, i \in [1, N], j \in [1, k]$ Therefore, a node i generates $\sum_{j=1}^{k} p_{e_{i,j}}$ events per time unit, and the event lifetime is the time required to saturate its memory. Then for a particular node i the Local Storage event lifetime $lt\_LS_{node_i}$ is:

$$lt\_LS_{node_i} = \frac{s}{\sum_{j=1}^{k} p_{e_{i,j}}} \text{time units}$$

In Local Storage the event lifetime very much depends on which node it is generated. This is, events generated by very active nodes will be available short time, whereas nodes with a low event rate will have their events available long periods. Therefore, Local Storage is an unfair system from the events lifetime point of view in heterogeneous scenarios in which different nodes contribute different loads to network applications. This unfairness basically means that even in the case where the network is plenty of storage resources, very active nodes will be overwriting old data quickly. However, in a homogeneous scenario in which all nodes have the same probability of generating events for each application, the system becomes fair, and all events experience a similar lifetime.

4.2 Round robin storage

Round Robin node selection appears as an optimal storage solution since it efficiently uses all storage capacity of the network. Then, in this mechanism a node is initially selected as storage point for all network applications. That node receives events until its memory is full. At that moment, a new node is selected to perform the storage task, and so on, until all nodes in the network have been selected, and thus all network storage capacity has been used. At this point, the initial node, which contains the oldest data, is again selected and it overwrites the oldest events to store the new ones.

Such Round Robin Storage requires a global synchronization of network nodes. This can be achieved either with a centralized solution in which a base station decides and notifies to the rest of the network who is the storage node at any moment, or with some sophisticated and complex distributed algorithm that allows synchronizing all nodes in the network to homogeneously select the current storage node. It must be noted that DCS neither requires a central node nor complex distributed algorithms to select home nodes. Although Round Robin Storage presents some issues to be implemented in practice, it is still a very interesting approach to be used as a benchmark to test the performance of our solution.

In order to model events lifetime in a Round Robin Storage system, we again assume a network in which N nodes are equipped with a memory able to store s events. In addition, we consider k applications running on the network, all of them generating events of the same size. Then, if $p_{e_{i,j}}$, $i \in [1, N]$, $j \in [1, k]$ is the probability that a node i generates an event of an application j per time unit, the expected lifetime (lt_RR) is:

$$lt\_RR = N * \frac{s}{\sum_{i=1}^{N} \sum_{j=1}^{K} p_{e_{i,j}}} \text{time units}$$

Then, in the Round Robin storage analysis the data lifetime depends on the traffic generated by all applications per time unit. In addition, events from different applications will have a similar lifetime independently of the load generated by each of them.

4.3 Dynamic DCS

Once we have analyzed two alternative storage models, now we present a model to compute data lifetime when employing our Dynamic DCS proposal. In addition, using the same model we will be able to answer one of the key issues of this paper: *what is the optimal number of replicas that maximizes event availability?* Finally, we validate the model via simulation and discuss the results.

We consider a slotted discrete time (epoch) system model where N nodes are available to store data in the system. Each epoch r (home nodes) of the N nodes are selected at random. We assume a uniform network deployment in which all nodes have the same probability of being selected as replicas. The selected r nodes are used to store the data on that epoch. The r nodes are assumed to be distinct, and store on average e events during each epoch. Each node is assumed to be able to store s events. When a node's memory is full, then the oldest event is deleted, and the new one is stored in its place.

The objective is to maximize the availability of events t epochs into the future, by selecting the optimal r. If r is too large then data will be initially available at many nodes i.e., high redundancy, but the events will be quickly overwritten over time. If $r = 1$ then data is stored in a single node, so if it gets overwritten at that node prior to t epochs it will no longer be available.

Lets us consider the performance for our mechanism. Suppose data is stored at epoch 0 in r nodes and each node's memory is full. So the new data is the newest piece of information in each of these r nodes. To compute the likelihood that data will be available t epochs into the future at one of these original nodes, we need to check whether the nodes are selected $S = s/e$ times in the

t epochs in the future. Let $A_i(0,t]$ denote the number of times that the ith node is selected after epoch 0 before epoch t. Clearly $A_i(0,t] \sim \text{Binomial}(\frac{r}{N}, t)$. In order for the data to be available at time t it must be that the $1,\ldots r$ original nodes were not all chosen more than S times each to store data, i.e., the probability that an event is not available after t epochs is given by

$$P(A_i(0,t] > S, \forall i = 1,\ldots r)$$

i.e., the data must no longer be available at any of the nodes where it was originally stored. Next, **assuming** that $N \geq r$ one can roughly consider that $A_i(0,t]$ are independent which gives:

$$P(A_i(0,t] > S, \forall i = 1,\ldots r) = \prod_{i=1}^{r} P(A_i(0,t] > S)$$

And it can be computed as follows:

$$P(A_i(0,t] > S, \forall i = 1,\ldots r) = (1 - P(A_i(0,t] \leq S))^r$$
$$= \left(1 - \sum_{i=1}^{S} P(A(0,t] = i)\right)^r$$
$$= \left(1 - \sum_{i=1}^{S} \binom{t}{i} \left(\frac{r}{N}\right)^i \left(1 - \frac{r}{N}\right)^{t-i}\right)^r$$

Then, the optimal value of r that extends the data lifetime, r^*, will be the one minimizing previous expression for $P(A_i(0,t] > S)$.

Finally, we can use previous model to compute the events lifetime for the Dynamic DCS solution:

$$lt\_DDCS = \sum_{i=1}^{\infty} iP^r(A(0,i] > S)\text{epochs}$$

The previous formula provides event availability in number of epochs. Therefore, we only need to multiply it by the epoch duration in order to obtain events lifetime in time units.

4.3.1 Model validation and discussion

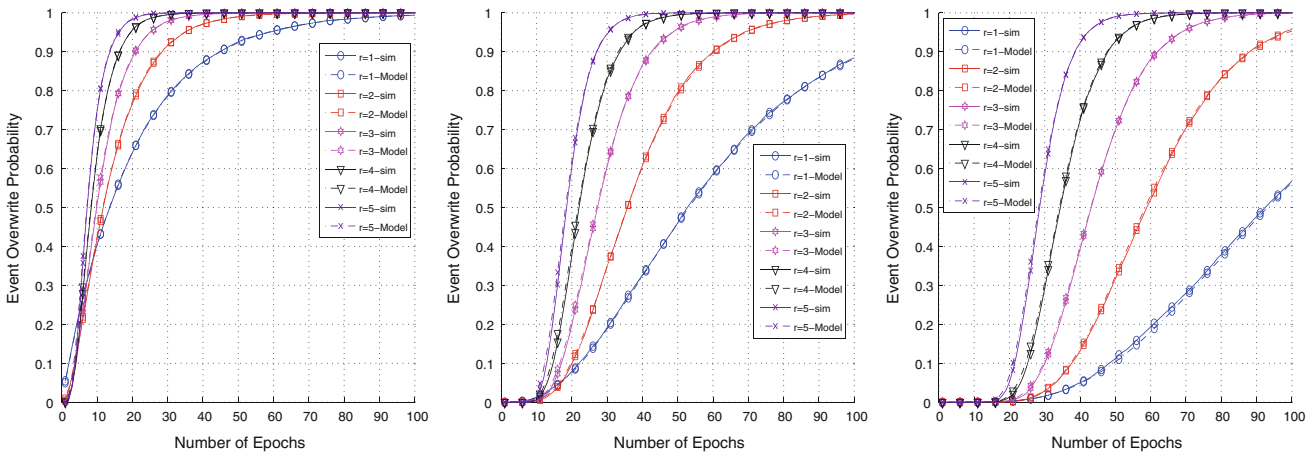
We validate the model via simulation. We consider a uniform deployment (e.g. grid) in which all nodes have the same probability of being chosen as home node.

We evaluate three different scenarios in terms of number of nodes: small size (20 nodes), medium size (100 nodes) and large size (500 nodes). In addition for each scenario we consider three different storage capacities: (1) small memory case in which a node has only capacity to store the events of one epoch, that means $S = 1$; (2) medium memory in which a node can store information of three different epochs, $S = 3$; and, (3) high storage capacity in which a node can store information of five epochs, thus $S = 5$.

We compute the event overwrite probability at different epochs for a number of replicas varying from 1 to 5. This let us compare whether the previous model correctly captures the event overwrite probability. Furthermore, by comparing the results for different number of replicas we are able to conclude what is the number of replicas that minimizes the event overwrite probability for a particular number of epochs.

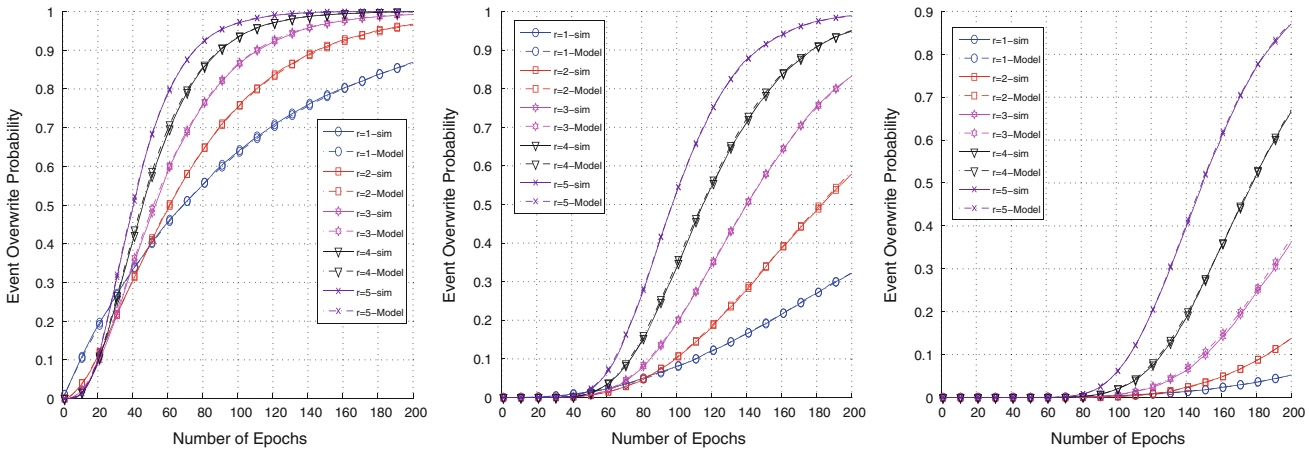
Figures 1, 2 and 3 represent the results for the small (20 nodes), medium (100 nodes) and large (500 nodes) network size respectively. For each of them subfigure (a), (b) and (c) show the low ($S = 1$), medium ($S = 3$) and high ($S = 5$) memory capacity respectively.

First of all, the simulations results demonstrate that the proposed model accurately captures the event overwrite probability. In all the cases the simulation and model lines collapse together, so they cannot be differentiated in the figures.



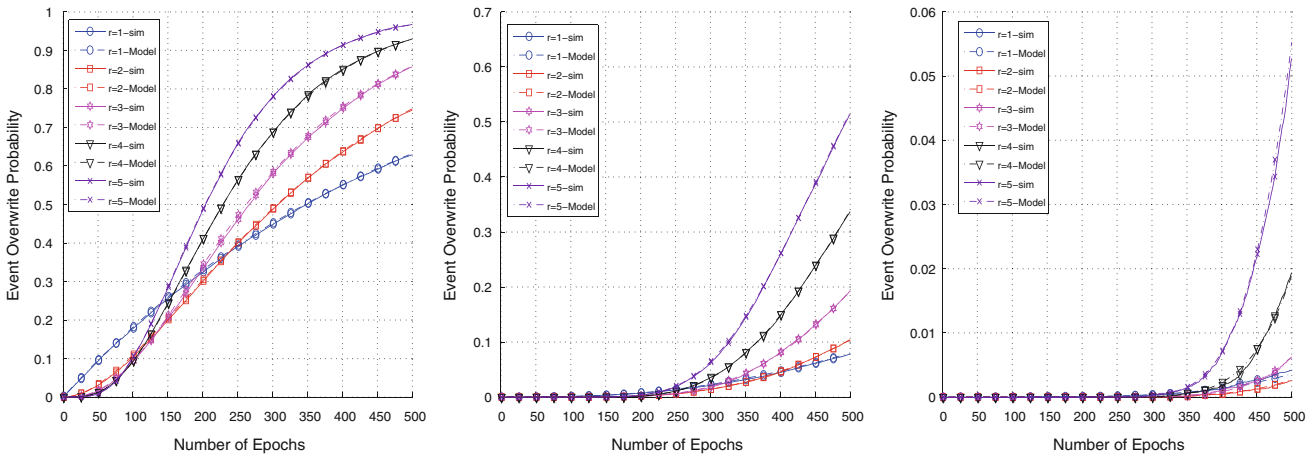
(a) Low memory (S=1) **(b)** Medium memory (S=3) **(c)** High memory (S=5)

Fig. 1 Event overwrite probability for a small size network ($N = 20$ nodes, $r = 1$–5 replicas, $t = 1$–100 epochs)



(a) Low memory (S=1) **(b)** Medium memory (S=3) **(c)** High memory (S=5)

Fig. 2 Event overwrite probability for a medium size network (N = 100 nodes, r = 1–5 replicas, t = 1–200 epochs)



(a) Low memory (S=1) **(b)** Medium memory (S=3) **(c)** High memory (S=5)

Fig. 3 Event overwrite probability for a large size network (N = 500 nodes, r = 1–5 replicas, t = 1–500 epochs)

To answer the question referring to the optimal number of replicas to maximize sensor event lifetime we can also use the model results. By looking at the graphs, the main conclusion is that in the long-term the best solution is always using 1 replica.

Furthermore, we advise to select one replica in all medium and high memory cases. Although under these conditions using more than one replica provides a lower event overwrite probability in the short term, the advantage is negligible in practice compared to the results obtained for a single replica. For instance, in the medium-size medium-memory scenario (Fig. 2(b)) from 1 to 80 epochs using a single replica is not the optimal solution. However, if we look at 40 epochs the optimal solution (4 replicas) reports an event overwrite probability of 0.002, whereas selecting only 1 replica 0.008. Therefore, that difference has a negligible impact in practice since both probabilities are very low.

The only scenarios in which using more than 1 replica would be worthwhile are those cases where: (1) the nodes' memory only has capacity to store information of a single epoch, (2) the application requires short-term storage (5 epochs for low-memory, 20 epochs for medium-memory, 250 epochs for high-memory). For instance, in the 100 nodes scenario (Fig. 2(a)), after 20 epochs the data loss probability in case of using 3 replicas is below 0.1, whereas the data loss probability is almost 0.2 when only a single replica is in place. Lets us consider a practical case in which the network information is collected once a day (24 h), the epoch duration is 1 h and the nodes memory only can store th information of one epoch. Then, in practice using a single replica means that on average we roughly loose 5 h of information every day, whereas this value is reduced to just 2 and a half hours when deploying 3 replicas, which is the optimal value provided by the model. We are aware that for some real applications losing 2.5 out

of 24 h is not acceptable, and in those cases we would need to increase the memory of the nodes until $S > 3$ to ensure they are able to collect 24 h of data. Therefore, it is importance to notice that previous example has been used just to illustrate a case where $r = 1$ is not the best option and what would be performance improvement in case of using the optimal number of replicas provided by our model, i.e. $r = 3$.

It is worth noting that the low-memory scenario refers to those cases in which a node can just store information of a single epoch. This could happen because, as suggested by the employed taxonomy, the nodes may be equipped with low storage capacity. However, in cases where an application generates a huge amount of events, even if the nodes have a high memory capacity, it could only store the information of one epoch, and thus the premises that rule the low-memory scenario are also applicable.

Finally, it must be highlighted that the defined model accurately captures the event loss probability in case the probability of selecting a node as replica follows a uniform distribution. However, this is not the case in many scenarios. For instance, in a network based on geographical information in which the nodes are randomly deployed, the probability of being selected depends on the area of responsibility of each node and it is no longer uniform.

Therefore, we have also evaluated via simulation the number of replicas that minimizes the event loss probability, and thus maximizes the events lifetime, in non-uniform scenarios. Although the event overwrite probability is different compared to the uniform scenario, the main remarks are still the same. (1) Using a single replica is the best option in the long-term, independently of the scenario size and nodes' storage capacity. (2) For medium and high memory scenarios, even if in the short-term the event overwrite probability is lower when using more than one replica, the difference compared to using a single replica is negligible. (3) Exclusively in those scenarios where the sensors are equipped with low storage capacity ($S = 1$) makes sense to deploy more than one replica to maximize the events lifetime.

In a nutshell, we should use more than one replica exclusively for those scenarios where network nodes can only store events of one epoch, and when, in addition, the application only requires short-term storage.

5 Performance evaluation

In this section we evaluate the performance of the proposed solution by means of simulation and compare it with Local Storage and Round Robin Storage solutions. We notice that the goal of this section is to evaluate the network storage performance in terms of event lifetime, thus we use network nodes that generate events (i.e. sensors) and do not focus on whether they also generate queries or not (i.e. they are also actors), or on adding pure actor node that generate queries in the network, since this is irrelevant to evaluate event lifetime. We assume a uniform DCS scenario in which all nodes have the same probability of being selected as a replica. For each experiment we obtain the lifetime of 30,000 events. In order to do that we generate 1,000,000 events to be sure we overwrite the 30,000 evaluated events.

We must notice that our goal is to evaluate the performance of the proposed solution in terms of data lifetime. Therefore, we are aware that sensor/actor nodes can present failures that can make them unreachable during short periods of time (e.g. low signal strength) or for long periods (e.g. battery depletion), but this is an inherent characteristic of sensor nodes for whatever solution, and thus it affects all the proposals we are comparing in this paper. Hence, in order to avoid distraction from our main goal of evaluating the data storage duration, we have avoided introducing nodes failures in our simulations.

5.1 Homogeneous scenario

We define a network with $N = 100$ nodes, in which each node is assigned a probability of generating an event per time unit and application. We simulate three applications operating in the network, named as $APP1$, $APP2$ and $APP3$. All nodes have a probability 0.01, 0.02 and 0.03 of generating an event for $APP1$, $APP2$ and $APP3$ per time unit, respectively. This means an overall traffic load of 6 events per time unit. In addition, sensors are equipped with a memory that can store $s = 300$ events. In order to evaluate the performance of Dynamic DCS we define an epoch duration of 50 time units for the three applications. In addition, we obtain the results for 1, 2 and 3 replicas.

Figure 4 presents the CDF of the events lifetime for Round Robin Storage, Local Storage, static DCS with a single replica, and Dynamic DCS for 1, 2 and 3 replicas. The first conclusion is that, as expected, using a static DCS proposal leads to very poor results. The median lifetime for the events is only 130 time units, 34 times lower than the median for the Dynamic DCS with 1 replica (4,370 time units). Therefore, static DCS solutions cannot be used as long-term storage systems

The results also validate the conclusions obtained in Sect. 4 since using a single replica extends the data lifetime when compared to those Dynamic DCS cases in which more replicas are deployed.

In addition, in a homogeneous scenario like the one employed in this experiment, we can apply the formulas obtained in Sect. 4 for event lifetime in Round Robin and Local Storage. Those formulas provide an event lifetime of 5,000 time units for both mechanisms. Looking at Fig. 4 it

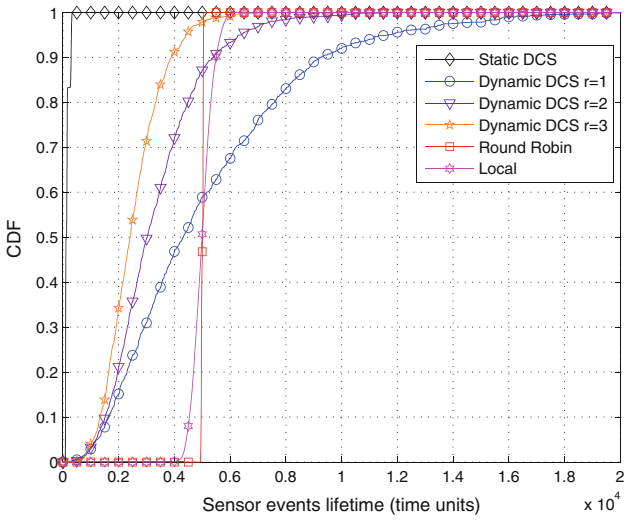


Fig. 4 CDF of event lifetime for Local Storage, Round Robin Storage, Static DCS and Dynamic DCS in a homogeneous scenario

can be appreciated that all events lifetime are very close to 5,000. However, it must be noted that Local Storage presents a higher variance due to te random generation of events. The sudden rise observed in the Round Robin Storage approach is due to the fact that it uses all network storage capacity before overwriting old events. Therefore, since the network generates on average 6 event per time unit and a node memory can store up to 300 events, each sensor will serve as storage node during 50 s on average. The network size is 100 nodes, thus each event will be overwritten on average after 5,000 time units (i.e. 50 time units $\times$ 100 network nodes). Therefore, all events will experience a lifetime around 5,000 time units, and this is why Fig. 4 shows such a sudden rise for Round Robin solution.

When Dynamic DCS is in place sensor events last in median 4,370 time units. In case the time units refers to seconds (6 events per second are generated in the network) the median event lifetime would be 73 min. In case, the time unit is mapped to 1 h (6 events per hour) then the median lifetime of the events grows up to 182 days. Finally, if we measure the time units in days (6 events per day), sensor events would typically be accessible almost 12 years. Obviously, the event lifetime very much depends on the load of the network. The higher the network load, the shorter the event availability. However, we can provide an objective result to measure the goodness of the proposed solution by comparing it with Round Robin Storage. The described Round Robin Storage is an optimal solution that maximizes the median event lifetime because it efficiently uses all network storage before overwriting an event. Therefore, with the epoch duration selected, our solution offers a median event lifetime that corresponds to 87 % of the maximum median lifetime provided by the Round Robin storage simulation (5,012 time units). Therefore, we claim that Dynamic DCS storage system is quite close to the benchmark in median.

Furthermore, it must be highlighted that the Round Robin Storage solution is optimal when the goal is to maximize the median event lifetime. However, if a particular application aims to keep some portion of the application data available as long as possible, then the best option is to use Dynamic DCS. None of the 30,000 events has a lifetime longer than 5062 time units under the Round Robin Storage mechanism, whereas more than 41 % of events last longer than that value when Dynamic DCS is in place. In addition, more than 30 % of events are available longer than 6,000 time units, and even 8 % of events still duplicate the Round Robin expected lifetime (5,000 time units).

Therefore, depending on the applications requirements Dynamic DCS could be presented as the best distributed storage mechanism in terms of maximizing the data availability in the sensornet, even improving the ideal Round Robin Storage.

5.2 Heterogeneous scenario

In this experiment, we evaluate a scenario with 100 nodes that are assigned different probabilities of generating events. We again use three applications but now we define three different nodes profiles. Out of the 100 nodes, 33 have a probability 0.01 of generating one event for each application per time unit, 34 nodes have a probability 0.02, and the last 33 nodes a probability 0.03. Under these conditions the traffic load is again 6 events per time unit. The epoch duration for all applications is again 50 time units.

Figure 5 represents the events lifetime CDF. We can appreciate that Dynamic DCS and Round Robin Storage are not affected by nodes load heterogeneity. The ideal Round Robin Storage, as it was previously stated, uses all the network storage capacity before deleting an event, while Dynamic DCS approximates that behaviour and also distributes the storage load among network nodes. Therefore, this experiment concludes that Dynamic DCS (and also Round Robin Storage) is a fair storage mechanism that does not prioritises any network application, but treats all events in the same way.

In addition, we again verify in a different scenario that Dynamic DCS is close to the optimum solution in median. In this particular experiment the median events lifetime of Dynamic DCS (4355 time units) is again an 87 % of the Round Robin median (5020 time units).

Finally, Local Storage results demonstrate that it depends greatly on the traffic load generated by each particular node. Then, the graph shows three clear steps

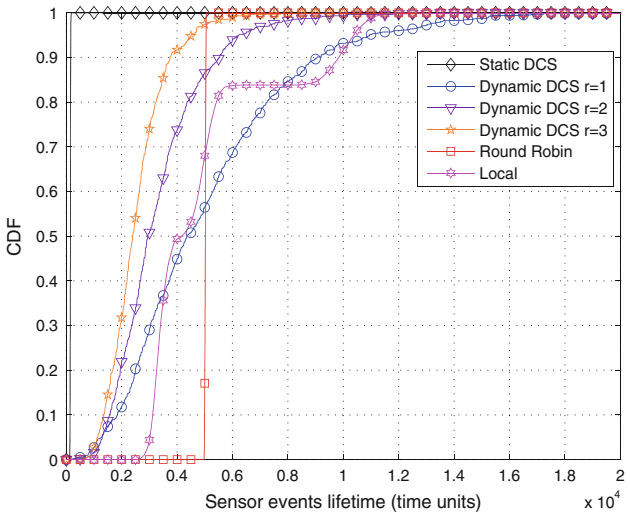


Fig. 5 CDF of event lifetime for Local Storage, Round Robin Storage, Static DCS and Dynamic DCS in a heterogeneous scenario

representing the lifetime of events stored in nodes with different traffic profiles, demonstrating that events lifetime for Local Storage is no longer homogeneous across network nodes.

5.3 Real WSN deployment scenario

In this subsection we rely on parameters extracted from real WSNs deployments to evaluate the performance of our solution as compared to Round Robin and Local Storage.

Our evaluation is based on the scenario presented in [14] where a WSN has been deployed to measure climate conditions inside The Mogao Grottoes. This WSN is formed by 241 nodes that monitor three climate parameters: temperature, relative humidity and CO2 density. The event generation rate is 1 event per minute for each parameter. Unfortunately, this paper does not provide the memory size for the sensors. In order to obtain sensors' memory size we use the data reported in [12] that refers to a real WSN deployment to monitor climate conditions for viticulture purposes. This paper uses sensors that include a 64 kb non-volatile memory for data storage, which is the value we use in our experiment. Furthermore, none of the two previous works provides information regarding the memory size use by each event. For that, we use the information reported in a third paper [11], which refers to a real WSN deployment for habitat monitoring, that reports a 25 bytes data payload (i.e. event size) in the generated packets. Hence, we consider that each events consumes 25 bytes of storage.

Figure 6 shows the CDF event lifetime for Local Storage, Round Robin, standard DCS, and Dynamic DCS with $r = 1$, $r = 2$ and $r = 3$. In the case of Dynamic DCS we establish an epoch duration of 30 s for all event types. First of all we notice that this scenario corresponds to an homogeneous one since all nodes generate events at the same rate for each application. Therefore, results in the graph are similar to those obtained in the homogeneous scenario, and hence the explanation we provided for homogeneous scenarios is also valid here. Using the equation that provides event lifetime in the case of Round Robin, we compute a median event lifetime of 6,400 s, which is very close to the median obtained from the simulation, 6,372 s (106.2 min or 1 h and 46 min). As it happened in the homogeneous case, Local Sotrage shows a very similar behaviour in all the nodes, having a very close median (6,354 s) to the benchmark established by Round Robin. Again, standard DCS appears as a bad solution to provide long-term storage. Finally, for the case of Dynamic DCS, we obtain 5,628 s as median event life time, which is 88 % of the Round Robin median result. In a nutshell, we extract the same conclusions discussed in the homogeneous scenario, thus we refer the reader to them in order to avoid redundancy.

5.4 Epoch selection analysis

It has been demonstrated that using the proposed Dynamic DCS can be utilized as a long-term storage system. However, there is a key parameter that has not yet been studied, which can tune the proposed Dynamic DCS system towards the Round Robin Storage mechanism or the poor static DCS results. That parameter is the epoch duration. On one hand, if we use short epochs we expect to have a deterministic behaviour close to Round Robin Storage. Short epochs mean frequent changes, and thus a balanced utilization of storage resources of the network because a

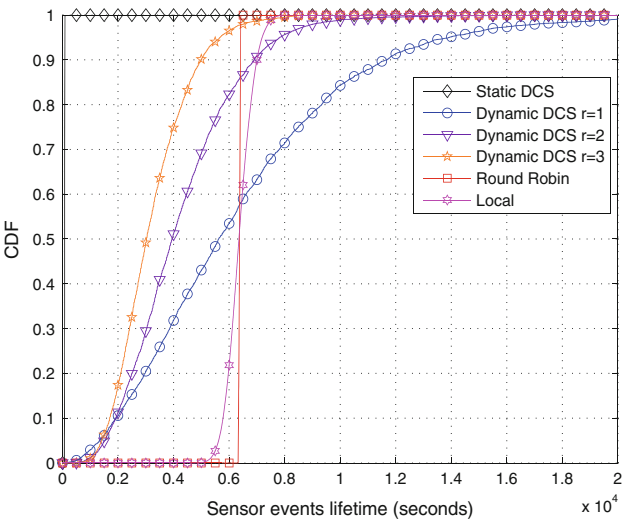


Fig. 6 CDF of event lifetime for Local Storage, Round Robin Storage, Static DCS and Dynamic DCS in a scenario using real deployment parameters

node stores few events and quickly passes the home node responsibility to another node. On the other hand, if we use very long epochs we take the risk of saturating the nodes memory before the end of the epoch. Therefore, events happening in an epoch *i* would overwrite events generated in the same epoch. Thus, the system performance would be approximating the performance offered by a static DCS solution, which has been demonstrated to be quite useless in case the applications require long-term event availability.

We evaluate the Dynamic DCS solution with a single replica for the following epoch durations measured in time units: 1, 5, 10, 50, 100, 500 and 1,000. Figure 7 shows the event lifetime CDF for a network with the same parameters as those described in the homogeneous experiment. In addition, Table 1 shows the average median event lifetime taken from 50 different simulation rounds for all epoch durations. As expected, the shorter the epoch the closer is the result to the Round Robin Solution. Then, for epochs of 1, 5 and 10 time units the average median event lifetime is 4991, 4938 and 4861 time units respectively. All of them are over a 97 % of the ideal Round Robin median (5,000 time units). However, when we select longer epochs, i.e. 500 or 1,000 time units, the median is reduced to 141 and 118 time units respectively. If we look at the graph, for those long epochs most events (70 and 85 % respectively) experience a short lifetime below 350 time units. However, those few events that are lucky of not being overwritten within a single epoch experience a very long availability. The reason is that epochs are very long and thus the nodes storing those "lucky" events will only be selected again as home nodes after a long time. Finally, intermediate epoch durations (e.g. 50 time units) in which the nodes are not saturated in a single epoch shows median lifetimes lower than those of the shorter epochs, but, as stated in the homogeneous scenario, they still show an important number of events reaching a long lifetime that cannot be achieved in case of selecting short epoch durations.

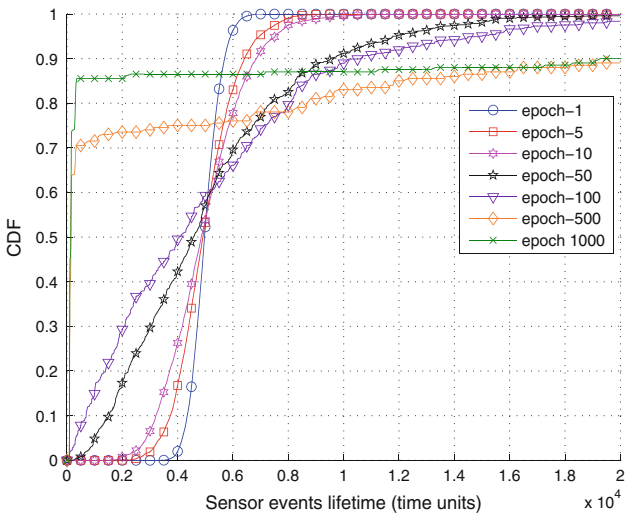


Fig. 7 Epoch duration impact on event lifetime for Dynamic DCS

Table 1 Average median event lifetime for different epoch duration in Dynamic DCS (D-DCS), and its comparison to the Round Robin (RR) bench mark 5,000 time units (t.u.)

Epoch duration (t.u.)	1	5	10	50	100	500	1,000
Avg. median D-DCS lifetime(t.u.)	4,991	4,938	4,861	4,378	3,726	141	118
\% of RR median (5,000 t.u.)	99.82	98.76	97.22	87.56	74.52	2.82	2.36

Therefore, short or medium epoch durations should be selected depending on the application requirements. It is clear that using long epochs is always a wrong decision because it would bring the network closer to the imbalanced and weak performance experienced by Static DCS proposals. In addition, Dynamic DCS could approximate the Round Robin Storage scheme in those cases where the application requires to extend the median event lifetime. This demonstrates that our Dynamic DCS proposal is a very flexible system that can be adapted to application requirements in terms of data availability by just tuning the epoch duration.

6 Conclusions and future works

We demonstrated in a previous work [4] that changing the home nodes over the time is essential to extend the network lifetime in DCS systems. This paper further enhances that benefit demonstrating that such dynamism also enables a long-term storage system. Therefore, this paper completes our previous investigations and now we can clearly state that it does not make sense to propose static DCS solutions, but rather it is necessary to share the load associated with being a home node among all network nodes. In this paper, we have demonstrated that a Dynamic DCS system provides great performance results not only in terms of energy consumption, but also in terms of long-term sensor events storage.

This research has assumed that nodes are homogeneously distributed across the network. However, in many real WSNs and WSANs deployment this is not the case, and this might directly impact the performance of Dynamic DCS. Therefore, as next step, we plan to investigate the level of such impact and, in case it is necessary, to adapt our solution so that it can be also used as a long-term storage system in non-uniform DCS scenarios.

Acknowledgments The research leading to these results has been partially funded by the Spanish MEC under the CRAMNET project (TEC2012-38362-C03-01) and eeCONTENT Project (TEC2011-29688-C02-02), by the General Directorate of Universities and Research of the Regional Government of Madrid under the MEDIANET Project (S2009/TIC-1468), and by the the INDECT project (Ref 218086) of the 7th EU Framework Programme. In addition, the work of G. de Veciana was supported by the National Science Foundation under Award CNS-0915928.

References

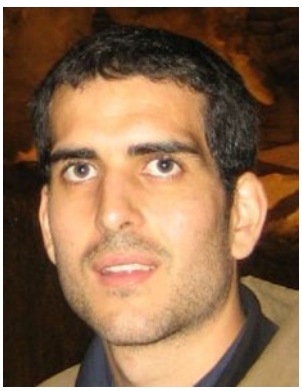
1. Ahn, J., & Krishnamachari, B. (2006). Fundamental scaling laws for energy-efficient storage and querying in wireless sensor networks. In *Proceedings of ACM international Symposium on mobile ad hoc networking and computing, MobiHoc '06* (pp. 334–343). New York, NY, USA: ACM.
2. Akyildiz, I. F., & Kasimoglu, I. H. (2004). Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks, 2(4)*: 351–367.
3. Albano, M., & Chessa, S. (2009). Distributed erasure coding in data centric storage for wireless sensor networks. In *IEEE Symposium on. computers and communications, ISCC '09* (pp. 22–27). Sousse, Tunisia, 5–8 July, IEEE.
4. Cuevas, Á., Urueña, M., & de Veciana, G. (2010). Dynamic random replication for data centric storage. In *Proceedings of the 13th ACM international conference on modeling, analysis, and simulation of wireless and mobile systems, MSWIM '10* (pp. 393–402). New York, NY, USA: ACM.
5. Dimakis, A. G., Prabhakaran, V., Ramchandran, K. (2006). Decentralized erasure codes for distributed networked storage. *In IEEE/ACM transactions on networking 14(1)*, 2809–2816.
6. Ghose, A., Grossklags, J., & Chuang, J. (2003). Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings of the 4th international conference on mobile data management, MDM '03* (pp. 45–62) London, UK: Springer.
7. Joung, Y.-J., & Huang, S.-H. (2008). Tug-of-war: An adaptive and cost-optimal data storage and query mechanism in wireless sensor networks. In *Proceedings of the 4th IEEE international conference on distributed computing in sensor systems, DCOSS '08* (pp. 237–251). Berlin, Heidelberg: Springer.
8. Karp, B., & Kung, H. T. (2000). GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking, Mobicom '00* (pp. 243–254). New York, NY, USA: ACM.
9. Le, T. N., Yu, W., Bai, X., & Xuan, D. (2006). A dynamic geographic hash table for data-centric storage in sensor networks. In *IEEE wireless communications and networking conference, WCNC '06* (pp. 2168–2174). New York, NY, USA: IEEE.
10. Liao, W.-H., Shih, K.-P., & Wu, W.-C. (2010). A grid-based dynamic load balancing approach for data-centric storage in wireless sensor networks. *Computer and Electrical Engineering 36(1)*, 19–30.
11. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., & Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on wireless sensor networks and applications, WSNA '02* (pp. 78–87). New York, NY, USA: ACM.
12. Matese, A., Di Gennaro, S. F., Zaldei, A., Genesio, L., & Vaccari, F. P. (2009). A wireless sensor network for precision viticulture: The NAV system. *Computers and Electronicsin Agriculture, 69(1)*, 51–58.
13. Mazzini, G., Conti, A., Verdone, R., & Dardari, D. (2008). *Wireless sensor and actuator networks*. Amsterdam: Elsevier.
14. Ming, X., Yabo, D., Dongming, L., Ping, X., & Gang, L. (2008). A wireless sensor system for long-term microclimate monitoring in wildland cultural heritage sites. In *IEEE international Symposium on parallel and distributed processing with applications, ISPA '08* (pp. 207–214). IEEE.
15. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., et al. (2002). GHT: A geographic hash table for data-centric storage. In *Proceedings of the 1st ACM international workshop on wireless sensor networks and applications, WSNA '02* (pp. 78–87). New York, NY, USA: ACM.
16. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., & Estrin, D. (2003). Data-centric storage in sensornets. *SIGCOMM Computer Communication Review 33(1)*, 137–142.

Author Biographies

Ángel Cuevas got his Ph.D. in Telematics engineering (2011), M.Sc. in Telematics engineering (2007), and M.Sc. in Telecommunications engineering (2006) at University Carlos III of Madrid. Since Jan. 2013 he is Visiting Professor at University Carlos III de Madrid, after holding a post-doc position at Institute Mines-Telecom from Mar. 2011. He is co-author of more than 25 papers in top venues like ACM CoNEXT, WWW, IEEE/ACM Transactions on Networking, ACM Transactions on Sensor Networks, IEEE Communications Magazine, etc. He is recipient of the best paper award at ACM MSWiM' 10.

Manuel Urueña received his M.Sc. degree in Computer Science from Universidad Politécnica de Madrid in 2001 and his Ph.D. degree in Telecommunications from Universidad Carlos III de Madrid in 2005. At present, he is an assistant professor in Telematics Engineering at Universidad Carlos III de Madrid. His research activities range from P2P systems, through load balancing and service discovery protocols, to Optical networks. He has been involved in several national and international research projects related with these topics, including the EU IST GCAP and the EU SEC INDECT projects.

Gustavo de Veciana (S'88-M'94-SM'01-F'09) received his B.S., M.S, and Ph.D. in electrical engineering from the University of California at Berkeley in 1987, 1990, and 1993 respectively. He is currently the Joe. J. King Professor at the Department of Electrical and Computer Engineering. He served as the Director and Associate Director of the Wireless Networking and Communications Group (WNCG) at the University of Texas at Austin, from 2003–2007. His research focuses on the analysis and design of wireless and wireline telecommunication networks; architectures and protocols to support sensing and pervasive computing; applied probability and queueing theory. Dr. de Veciana has served as editor for the IEEE/ACM Transactions on Networking. He was the recipient of a National Science Foundation CAREER Award 1996, co-recipient of the IEEE William McCalla Best ICCAD Paper Award for 2000, co-recipient of the Best Paper in ACM Transactions on Design Automation of Electronic Systems, Jan 2002–2004, co-recipient of the Best Paper in the International Teletraffic Congress (ITC-22) 2010, and of the Best Paper in ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems 2010. In 2009 he was designated IEEE Fellow for his contributions to the analysis and design of communication networks. He is on the technical advisory board of IMDEA Networks.



Rubén Cuevas obtained his Ph.D. (2010) and M.Sc. (2007) in Telematics Engineering at University Carlos III of Madrid (Spain). Since Sep. 2010 he is Assistant Professor at at University Carlos III of Madrid. He has been research intern at Telefonica Research Lab and Courtesy Assistant Professor at University of Oregon in 2008 and 2012, respectively. He is co-author of more than 30 papers in prestigious international journals and conferences such as IEEE/ACM Transactions on Networking, IEEE Networks, ACM CoNEXT, or IEEE Infocom.

Noël Crespi professor, holds a Master's from the Universities of Orsay and Kent, a diplome d'ingénieur from Telecom ParisTech, a Ph.D. and a Habilitation from Paris VI University. From 1993 he worked at CLIP, Bouygues Telecom and then France Telecom R\&D in 1995. In 1999, he joined Nortel Networks as Telephony Program manager. He joined Institut Telecom in 2002 and is currently professor and Program Director, leading the Service Architecture Lab. He coordinates the standardisation activities for Institut Telecom at ITU-T, ETSI and 3GPP.