Published in IET Communications
Received on 1st July 2010
Revised on 31st March 2011
doi: 10.1049/iet-com.2010.0574

In Special Issue on Distributed Intelligence and Data Fusion for Sensor Systems

ISSN 1751-8628

Modelling data-aggregation in multi-replication data centric storage systems for wireless sensor and actor networks

Á. Cuevas[1] *M. Urueña*[2] *R. Cuevas*[2] *R. Romeral*[2]

[1]*Institut Telecom, Telecom SudParis, 9 rue Charles Fourier, 91011 Evry-Cedex, France*
[2]*Universidad Carlos III de Madrid, Avenida de la Universidad, 30, 28911 Leganés, Spain*
E-mail: angel.cuevas_rumin@it-sudparis.eu

Abstract: This paper studies data-centric storage (DCS) as a suitable system to perform data aggregation on wireless sensor and actor networks (WSANs), in which sensor and actor nodes collaborate together in a fully distributed way without any central base station that manages the network or provides connectivity to the outside world. The authors compare different multi-replication DCS proposals and choose the best one to be applied when studying data aggregation. In addition, the authors provide mathematical models for the production, consumption and overall network traffic for different application profiles. Those application profiles are based on the ability of a particular application to perform data aggregation and on what type of traffic is dominant, either the consumption or the production one. Furthermore, the authors provide closed formulas for each application profile that defines the optimal number of replicas that minimise the overall network traffic. Finally, the authors validate the proposed models via simulation.

1 Introduction

Wireless sensor networks (WSNs) have focused on the interest of both the industry and academia since the last decade. A wireless sensor is a cheap node with wireless communication capabilities that is able to measure some features of its environment, but has limited storage and battery capacity. When many of these nodes collaborate together to sense some physical phenomenon of interest across a region, transmit it by relay nodes and process it in order to achieve a common goal, they create a WSN. The main characteristic of WSNs is that their nodes are battery powered, thus being limited in energy as well as in processing and storage capabilities. These limitations have differentiated WSN as a separate research area inside the wireless *ad hoc* networks one. Therefore specific protocols and algorithms have been proposed for WSN at all levels of the network stack, from physical to application layer [1]. For instance, in [2] we find the main MAC protocol proposals for WSN, whereas the most important routing proposals for WSN are presented in [3, 4]. WSN-specific transport protocols are summarised in [5]. Some examples of WSNs applications are described in [6]. Furthermore, the interest of the industry on WSN has pushed the standardisation of light-weight and low-power communication technologies that are suitable for sensor nodes. For instance, IEEE specified the 802.15.4 standard that covers physical and MAC layers. On top of 802.15.4, the Zigbee Alliance has defined a protocol stack covering routing and application layers for WSNs.

In the last few years, researchers have extended their interest on WSNs to more complex networks such as wireless sensor and actor networks (WSANs) [7, 8], in which a new player, the actor or actuator node, performs some actions based on the information retrieved from the sensors in the network. In addition, another type of network is the unattended wireless sensor network (UWSN) in which there is no connection to the external world (or it is intermittent) [9]. In this paper we focus on a network that mixes both concepts. This network works autonomously without external connection so that sensors and actors collaborate together to obtain a common goal. We name these types of networks as autonomous wireless sensor and actor networks (AWSANs) in this paper. For this type of network most of the solutions proposed for standard WSNs are no longer useful because there is not a central base station. Instead the intelligence of the system needs to be achieved via a distributed collaboration among the nodes in the network. It must be noted that these kinds of networks are not a reality yet, but we believe that autonomous networks such as AWSANs will play an important role in the research arena within the next few years.

An example would be an AWSANs in charge of maintaining the best possible conditions in a plantation. Then, based on the environmental measurements (temperature, humidity, rain intensity, light intensity etc.) and other higher-level events generated based on fusing low-level data (e.g. potential fire detected, risk of flooding etc.), it performs different actions to protect the crop. Those actions could be potentially complex, that is, in case of a

potential fire, some nodes detect the wind direction and speed to irrigate the area of the plantation more likely to be affected by the potential fire; in case of an excessive light intensity some nodes activate the engines controlling a mobile cover structure that protects the affected area etc.

One of the main issues to be solved in AWSANs is how the information is stored and delivered in a fully distributed way. Data-centric storage (DCS) [10] appears as the most suitable technology for data storage and delivery in AWSANs. The main idea under DCS is to create a distributed storage system based on the data or event types, without knowing who the producers and consumers of those events are. DCS proposes to choose one (or more) rendezvous node(s) to store the data related to a particular event or data type. The rendezvous node is chosen based on the event type. Then, a node that measures an event (producer) only needs to know the event type to store the data in the appropriate rendezvous node, and so another node that wants to retrieve the information related to that event type (consumer) can query the same rendezvous node to obtain that information. In DCS, data consumers and event producers do not need to have prior knowledge of each other, because they are able to find which is the rendezvous node of a particular event type.

Previous works in the literature [11–13] have demonstrated that using several rendezvous nodes (also called replicas or replication nodes) allocated across the network reduces the overall network traffic. This reduction implies a lower energy consumption that leads to an extension of the network lifetime. In addition, using several replicas alleviates the saturation suffered by a single rendezvous node. We refer to these systems as multi-replication DCS. In multi-replication DCS networks, consumers and producers always communicate with its closest replication node (from all replication nodes placed in the network). Later, two different approaches could be applied: (i) When consumption traffic dominates production one (for instance, when there are much more consumers than producers in the network), production events are replicated and stored in all the replicas, and thus consumers only need to access the closest replication node to retrieve the suitable information of a given event type. (ii) However, when production dominates consumption traffic, production events are solely stored in the closest replica to the producer, whereas consumer queries are routed (replicated) to reach all replication nodes, which reply with the suitable information. A key issue in multi-replication DCS systems is to find the appropriate number of replication nodes to be used to minimise the overall network traffic, which in turn means lowering the network energy consumption.

Furthermore, multi-replication DCS networks facilitate both data fusion and data aggregation in a fully distributed way, which could be very useful in an AWSAN. Data fusion is considered in this paper as the action of generating high-level events (e.g. fire) based on basic information stored in the network (i.e. temperature, humidity, smoke density, etc.). Therefore nodes that consume basic information could act as data fusion nodes, at the same time being producers of high-level events. We define data aggregation as the action of reducing (in case the application allows it) the traffic generated by producer nodes before answering consumer queries. For instance, if an application only needs consumers to know the average temperature, it is not necessary to send all individual temperature measurements to the consumer, but some data-aggregation process could be performed to compute and answer just the average value. In DCS, replicas are an ideal place to perform data aggregation since they receive data from all the producers in its surrounding area. However, to the authors knowledge, none of previous DCS proposals have considered cases in which the information can be aggregated and/or fused, even though there are previous database proposals for WSN such as TinyDB [14] and madwise [15] that could be adapted to DCS networks and would facilitate the aggregation process.

This paper also proposes a taxonomy of application profiles based on two characteristics: (i) aggregation allowance, and (ii) dominant traffic (production or consumption). This taxonomy serves us as a tool to define and validate analytical models for the consumption, production and overall network traffic for each of the application profiles. The main contribution of the paper is obtaining, for all the profiles, closed formulas for the optimal number of replication nodes to be used to minimise the overall network traffic.

The remainder of this paper is structured as follows: Section 2 describes the original DCS work that proposed to use a single rendezvous node to store the data related to a given event type. Main multi-replication DCS proposals are introduced in Section 3. In addition, they are compared via simulation to find the best multi-replication DCS proposal. In Section 4, we establish an application profile taxonomy for AWSAN networks. In Section 5, the traffic of each application profile is modelled and the optimal number of replicas to minimise the overall network traffic is found. The models are validated via simulation in Section 6. Finally, Section 7 concludes the paper.

2 Data-centric storage

Shenker *et al.* first introduced the concept of DCS in [10]. They combined the idea of distributed hash table (DHT) together with greedy perimeter stateless routing (GPSR) [16], a geographic routing protocol, to create the first DCS system called geographic hash table (GHT). Moreover, it must be highlighted that lot of research has been carried out in the last years having DCS as its main focus. For instance, there are specific routing proposals for DCS such as pathDCS [17], HVGR [18] and Double Rulings [19]. In addition, there are two works, DELiGHT [20] and Q-NiGHT [21], which consider DCS in non-uniform networks. In particular, Q-NiGHT focuses on studying QoS for DCS networks. For more detailed information about related works on DCS, we survey the above-mentioned works and other relevant ones in [12].

In this section we first describe GPSR and later introduce GHT. It must be noted that GHT assumes that sensors are able to locate themselves within the sensornet by using GPS or any other location device or system. In addition, the size and borders of the network are well known. These two assumptions in the baseline work are also considered in most of the DCS proposals that use GPSR as a routing protocol.

GPSR employs two different algorithms for routing: the first one is called greedy forwarding that in each hop moves the data as close to the destination as possible, that is, a node always chooses the closest neighbour to the destination location as the next hop. However, sometimes, no neighbour is closer to the destination than the current node because there is a routing hole. In such cases GPSR uses the second routing algorithm, called perimeter routing. This algorithm uses the right-hand rule [16] to surround the

doi: 10.1049/iet-com.2010.0574

hole. In case a node closer to the destination than the one starting the perimeter routing is found, then the routing mode is switched again to greedy forwarding.

In GHT when a producer sensor detects an event, it uses a hash function over the event name [e.g. hash('TEMP')]. The hash function provides as its output a spatial location in the sensor field. Then, when a producer detects an event, it gets the spatial location provided by the hash function and invokes a put(k, d) operation (where k is the key for the event type and d the data) that forwards the data towards that spatial location using GPSR. The node closest to that spatial location becomes the rendezvous node (also called home node) for that event type and receives the producer message, because GPSR itself is enough to find the node closest to any given position. In turn, when a consumer wants to retrieve the data related to that event type, it uses the same hash function over the event type obtaining exactly the same spatial location. Next, it performs a get(k) operation that forwards a query using GPSR to that spatial location, thus reaching the home node that replies with the stored data for that event type. Then, in GHT the reasons that no neighbour is closer to the destination location than the current node could be: (i) a routing hole or (ii) that the current node is the one closest to the destination coordinates. Therefore if when using the perimeter routing the message reaches the same node that started it, that node understands that it is the closest to the destination coordinates, and hence the responsible of storing the information related to that event. Then, that node becomes the home node and all the nodes in the perimeter that encloses the destination coordinates are called home perimeter.

The authors of GHT already warned that using a single home node could create a hot-spot problem, and they proposed to use an additional protocol, called perimeter refresh protocol (PRP), which uses GPSR to replicate the data in all the nodes of the home perimeter. In PRP, the home node periodically sends refresh messages around the home perimeter that, in addition to replicate the information, is a simple mechanism to realise if there is a new node closer to the destination coordinates than the current home node. If this happens, that node, after receiving the refresh message, will update all the nodes within the home perimeter indicating that it is the new home node and from that moment it is in charge of replicating the information in the home perimeter.

Although local replication alleviates the home node's hot-spot problem, it does not fully solve it because all the nodes surrounding the home node must still relay production events and consumption queries, thus they are also incurring a high energy consumption, especially if that event type occurs frequently in the network and/or is highly demanded by consumers. In particular, owing to the GHT fault tolerance mechanism, when a home node runs out of battery, some other node in its surrounding area is chosen as the new home node. Therefore the hot-spot problem keeps being the same over the time. Apart from this hot-spot problem, the fact of always using a single home node does not guarantee to minimise the overall network traffic. The home node could be potentially far away from consumers and producers so that their messages travel long paths until they reach the home node. In many cases, it is better to place several replication nodes along the network to reduce the producers' and consumers' communication paths with the closest replication node, while home nodes communicate among them. In the next section we discuss the main proposals in the literature defining multi-replication DCS systems (i.e. DCS systems with several replicas for a particular event type), compare them and choose the best one in terms of traffic reduction to be used by the different AWSAN application profiles introduced later in the paper.

3 Multi-replication DCS proposals

Owing to the problems explained in the previous section, some works in the literature have focused its interest on providing DCS systems with multiple rendezvous nodes (multi-replication DCS solutions).

The first proposal [22, 23] was introduced by GHT authors themselves. They propose to use a hierarchical grid replication mechanism, where the number of replicas (N_r) is defined as $N_r = 4^d$, being d the so-called network depth. Then, when $d = 0$ there is only one randomly placed home node. When $d = 1$, four replicas are allocated in a grid-fashion. That is, the network is divided in four quadrants of the same size, and a replication node is placed in the same relative position inside each of these quadrants. For instance, in a 100×100 area network, if the first home node is placed (i.e. output coordinates of the hash function) in the (10, 25) position, three more replicas will be generated at the coordinates (60, 25), (10, 75) and (60, 75). When $d = 2$, each of the previous 4 quadrants is divided into other 4, leading to 16 quadrants of the same size, placing one replica in each one in the same relative position. Therefore it is a recursive, hierarchical and grid-structured replica allocation mechanism. Once the replication nodes are allocated, the authors assume that all nodes know the value of d, and thus they are able to calculate the positions of all replicas. Then, producers store their event information in their closest replica, while consumer queries have to reach all replicas to retrieve the information. These queries follow a recursive path, starting in the original home node, and moving to other replicas level by level. Finally, the replicas' replies follow the same path back. The drawbacks of this solution are that it does not provide any insight on how to find an appropriated network depth value, and that the proposed recursive data acquisition mechanism is very costly in terms of traffic.

Tug of War (ToW) [11] uses the same replica allocation mechanism based on a 4^d grid structure, but it adds three novel contributions: (i) A more efficient routing between the replicas, the so-called combing routing, which takes advantage of the grid structure to create a shorter replication tree. Queries and events travel along one row and all the columns of the replication tree to efficiently reach all replicas. (ii) It proposes two communication modes. When production traffic dominates consumption one (also called write-one-query-all mode), each producer stores the information in its closest replica and consumer queries reach all replicas using the proposed combing routing. This mode is similar to the one described in GHT with multiple replicas. The second mode is considered for the case when consumption dominates production (also called write-all-query-one mode). In this mode, producers send event messages to the closest replica, which in turn replicates that data using the combing routing to the remaining replicas. Therefore all information related to that data type is available at all replicas, and thus, a consumer just needs to query its closest replica to retrieve the available information. (iii) They provide an analytical model that computes the optimal network depth value, d^*, to minimise the overall network traffic based on the consumption

queries and production events. Fig. 1 shows both operation modes using the proposed combing routing to create a replication tree among the replicas.

The main problem of multi-replication DCS solutions using the 4^d replication mechanism is that they are not adaptive enough. For instance, in some cases, they have to decide to use either 16 ($d = 2$) or 64 ($d = 3$) replicas and probably none of them is the optimal value, but some other value in between should be used instead.

Towards this end, the quadratic adaptive replication (QAR) [12] allows the number of replicas to follow a more adaptable quadratic evolution $N_r = d^2$. Although QAR uses the same grid-replication scheme as GHT with multiple replicas and ToW, it allows to select the optimal number of replicas from a wider set of values (i.e. 1, 4, 9, 16, 25, 36, 49, 64, 81, etc.). For instance, between 16 and 64 replicas three other QAR values could be selected: 25, 36 and 49. In addition, QAR also provides an analytical model that defines the optimal number of replicas, N_r^*, to be used to minimise the overall network traffic. QAR uses the combing routing defined by ToW and also defines two operation modes.

The authors in [13] have proposed a theoretical framework that defines scaling-laws for DCS systems. They provide an analytical model that computes the optimal number of replicas to be uniformly deployed within the sensornet. However, they do not specify or provide any example regarding how such uniform deployment should be done in a real scenario. In addition, they do not validate their model results and, as we check later in this section, Scaling-Laws present worse results than QAR and ToW.

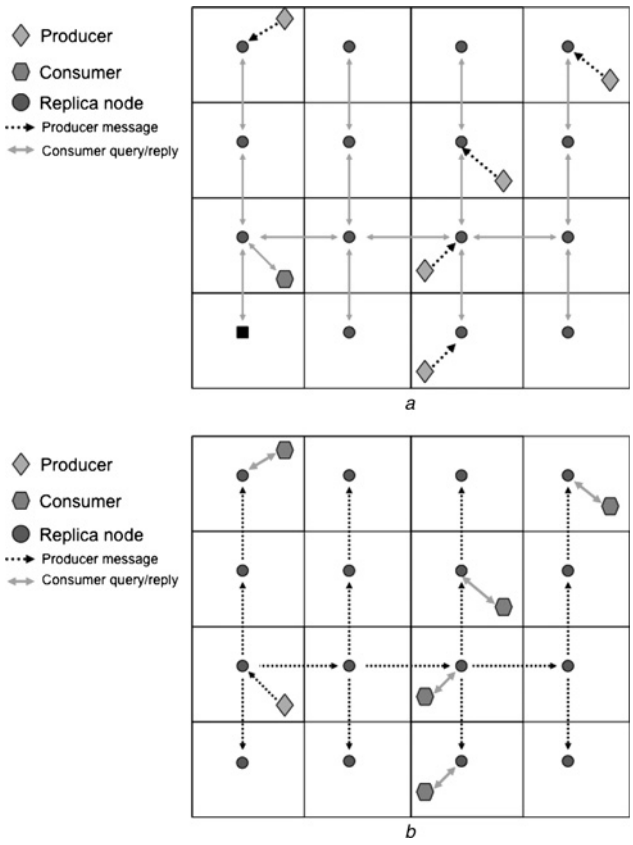

Fig. 1 *Combing routing in Tug of War (ToW) and quadratic adaptive replication (QAR)*

a Production dominates consumption mode
b Consumption dominates production mode

It must be noted that none of the analytical models that compute the optimal number of replicas in the different solutions provides an integer value that can be directly used, and thus one of the closest values allowed by each mechanism must be chosen. For instance, the QAR model could return $N_r^* = 21.23$ replicas in a particular case, being 16 and 25 the closest allowed values. Then, we evaluate the traffic model with both values and select the one that returns the lowest network traffic.

3.1 Comparison of multi-replication DCS solutions

In this section we compare the different multi-replication DCS solutions in the literature (QAR, ToW, Scaling-Laws, GHT with multiple replicas and the original GHT with a single replica) to decide which is the one that minimises the overall network traffic. We analyse the case of a producer sending the event data to the closest replica, which replicates the information in the remaining replicas, while consumers only need to access the closest replica to retrieve the information. For this mode all DCS solutions provide a model to compute the optimal number of replicas except for multi-replication GHT. Multi-replication GHT and ToW follows a 4^d grid replication schema, thus we use for both the optimal number of replicas provided by the ToW model. The results of the other operation mode are similar, but they are not shown owing to lack of space.

We have run simulations using a custom simulator that avoids MAC and physical layer issues. The simulated AWSAN has the following characteristics: an area of $A = 1000 \times 1000\ m^2$, $N = 5000$ nodes, and a node transmission range of $Tx = 50$ m. In all the described models, the optimal number of replicas depends on the ratio between the consumption and production traffic. We refer to this ratio as $(\varepsilon_c/\varepsilon_p)$. We range it from 1 to 40 and simulate 50 scenarios for each ratio value to estimate the average network traffic for each approach. In order to obtain meaningful results, we use the total number of hops traversed by events, queries and replication messages as the measure of the overall traffic. In order to present the actual differences among all the approaches, we have selected the best one (QAR) as the baseline, and account for the overhead generated by all the other solutions for each particular $(\varepsilon_c/\varepsilon_p)$ ratio.

Fig. 2*a* shows the extra traffic generated by the different approaches compared to QAR, and Fig. 2*b* shows the number of replicas used by each solution for a particular ratio (multi-replication GHT uses the same number than ToW and standard GHT always uses a single home node).

It is clear that QAR outperforms all other approaches. ToW generates in average 4% extra traffic and in certain ratio values this overhead goes up to a 17% when compared with QAR. Scaling-Laws generate an average extra traffic of a 34% with peaks close to a 50%. GHT with multiple replicas increases QAR traffic in average a 20% reaching for some ratios peaks of a 50%. Finally, GHT with a single replica is the worst solution generating an extra traffic cost of a 127% compared to QAR. Therefore QAR is the best multi-replication DCS mechanism, which minimises the overall network traffic owing to a higher adaptivity compared to ToW and GHT with multiple replicas. Moreover, Scaling-Laws' proposal does not seem to be a very accurate model for multi-replication DCS. Therefore based on these results, we propose QAR as the multi-replication DCS system for AWSANs in the rest of the paper.

doi: 10.1049/iet-com.2010.0574

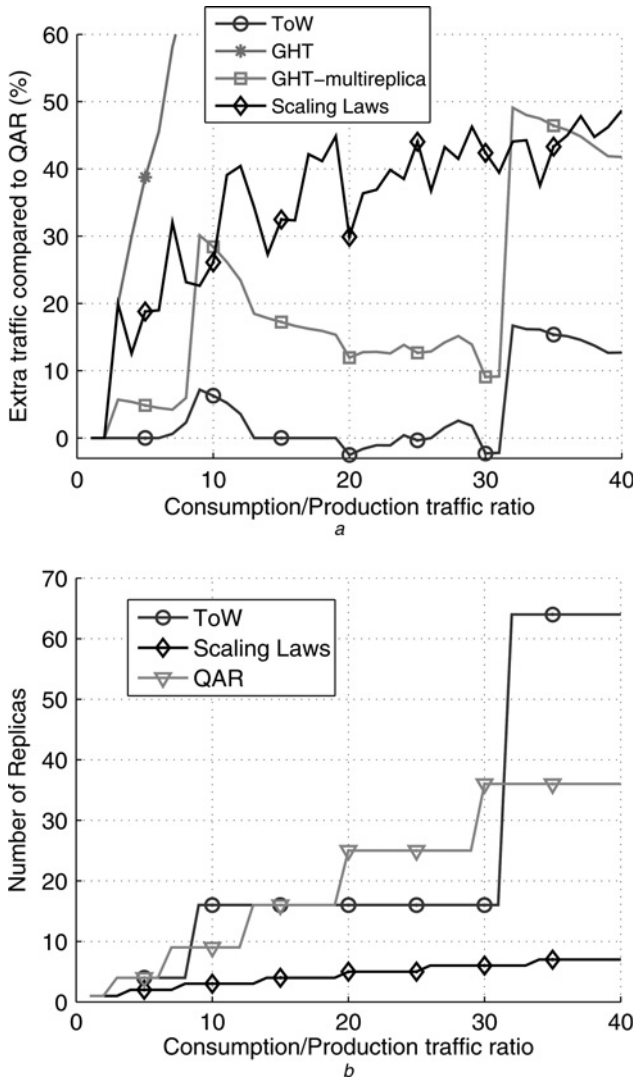

Fig. 2 *Multi-replication proposals comparison*
a QAR traffic improvement
b Optimal number of replication nodes

4 AWSAN application taxonomy

This section introduces an application profile taxonomy that covers a large number of potential AWSAN network applications that rely on DCS. Those application profiles serve us as a tool to develop mathematical models to find the optimal number of replicas that minimise the overall network traffic and thus reduces the energy consumption within the network. First of all, these profiles focus on applications where consumers potentially need to access all data from any part of the network (i.e. locality is not taken into consideration). The taxonomy is established based on two different characteristics: (i) the first one differentiates applications in which the information can be aggregated in replication nodes from those applications that do not allow data aggregation. (ii) The second characteristic differentiates which operation mode should be used depending on whether consumption traffic dominates production one or the other way around.

Following, based on the previous characteristics, we briefly describe the four proposed applications profiles, which are summarised in Table 1:

Table 1 Taxonomy of AWSAN-DCS application profiles

	Cons > Prod	Prod > Cons
non-aggregation	ConsNonAggr	ProdNonAggr
aggregation	ConsAggr	ProdAggr

• *ConsNonAggr:* In this application profile, consumption traffic dominates production one. That could happen when: (i) consumer queries and producer events rates are similar, but the number of consumers is much larger than the number of producers; (ii) the number of consumers and producers is similar, but the consumer queries rate is higher than the production events one; (iii) rates and the number of consumers and producers are similar, but each query is replied with a large amount of data that generates a lot of messages. In addition, this profile does not allow data aggregation because consumer nodes need to retrieve all the data stored within the network for that application.

An example of this application profile is a service discovery application, in which few nodes provide a service that is demanded by many other nodes in the network. In this case, service provider nodes are producers whereas nodes requiring the service are consumers, and thus the number of consumer queries is larger than that of production events.

• *ProdNonAggr:* In this profile the production traffic dominates the consumption one. In addition, all the data stored within the network for a particular application are demanded by the applications consumers.

Some monitoring applications (e.g. animal detection) cannot aggregate the events they generate (e.g. an application that tracks animals cannot aggregate an event that locates an animal A in a position X and animal B in another position Y, both individual values should be available). Furthermore, in monitoring applications usually the number of events will be much higher than the number of consumer queries (e.g. a user with a PDA obtaining the animal readings once per week).

• *ConsAggr:* This profile contains applications in which consumer queries traffic dominates the production one. Those applications could take advantage of aggregating data provided by different events in order to reduce the overall network traffic. Therefore individual events produced in a given area can be aggregated by the replication node responsible of that area. For instance, basic environmental data such as temperature, humidity etc. are clear examples of information that can be aggregated to calculate maximum, minimum or average values.

Applications matching this profile are those where producers generate information suitable to be aggregated of a very popular event type and it is consumed by many nodes in the network. For instance, in an intrusion detection system several sensors could detect the presence of an intruder and send the same detection event to the closest replica for all those sensors. That replica aggregates all these detection events containing the same information into a single message and replicates it in all the remaining replicas. There could be many actor nodes consuming intrusion information to perform different actions (e.g. raise an alarm, lock doors and/or gates etc.).

• *ProdAggr:* In this application profile, the production traffic dominates consumption one. In addition, replicas can aggregate those events received from producers to achieve an effective overall traffic reduction.

An example of this profile is an application in which a bunch of nodes produce environmental information (e.g. temperature, humidity etc.) that can be aggregated to obtain statistical parameters such as average, standard deviation, minimum and maximum values. Then, some actor nodes deployed across the network consume those values to perform some actions in the field (e.g. open a water valve).

The last two application profiles refer to applications that allow data aggregation. We define in this paper an aggregation factor, f, that is useful to understand with a single metric how much aggregation is being performed in the network. In addition, we discuss how this parameter is related to the optimal number or replicas.

5 Analytical models for the application profiles

In this section we present the consumption, production and overall traffic models for all the introduced application profiles. In addition, we obtain a closed formula for each profile that determines the optimal number of replicas that should be used to minimise the overall network traffic.

Before describing the models, let us define some basic assumptions for all of them. We assume that all nodes know which are the network limits as well as its own position, which is a common assumption in the DCS baseline work and all the multi-replication works described in Section 3. The production events and consumption queries are homogeneously distributed across the whole network. This means that there is no locality neither in the events nor in the queries generation. We propose a distance-based model that does not only measure the traffic in terms of messages/second, but also accounts the distance travelled by the messages, since more distance implies longer paths, which have a direct impacton the energy consumption. Therefore the model measures the traffic in (messages*meter/second). Finally, it must be noted that our models are thought for dense networks in which messages can roughly follow straight paths from source to destination nodes.

5.1 Analytical model of ConsNonAggr application profile

This model covers the case in which consumption traffic dominates production one. Producers first send their events to the closest replica. In turn, this replica copies every single event in the remaining replication nodes. Consumers send their queries to the closest replica that replies with the suitable information. Therefore in this application profile the replication traffic (messages exchanged among replicas) is a large portion of the overall network traffic.

We first introduce which are the parameters for the ConsNonAggr analytical model.

• *e:* Production rate. Average number of production events per second per producer. It is measured in events/second.
• *q:* Consumption rate. Average number of consumption queries per second per consumer. It is measured in queries/second.
• β: Number of messages generated per production event (i.e. usually one). It is measured in messages/event.
• α: Number of messages generated per consumer query. It should be two at least, since each consumer query should have at least one reply message. However, if it is requesting lot of information, more than one reply message could be sent back from the closest replica to the consumer. It is measured in messages/query.
• N_p: Number of producer nodes.
• N_c: Number of consumer nodes.
• N_r: Number of replicas.

The production traffic has two different elements: (i) the traffic generated from the producers to the closest replica, and (ii) the traffic owing to replication. The first one is modelled as the distance between two random points in a square cell (we remember that QAR divides the network in as many cells as replicas being used). Therefore the first part of the production traffic is

$$T_{p_1}(N_r) = \beta N_p e \frac{\delta\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

where δ defines the average distance between two random points within a unit square area and it is equal to 0.52 distance units [11] (It must be noted that in case the sensornet shape is not square, there will be some areas after the grid division that do not have a square shape. In those cases using $\delta = 0.52$ would lead to slightly inaccurate results. However, if a better value for δ is found for a particular network it can be applied to the described model), and $\sqrt{A}$ maps that distance into the actual area of the network.

The second component of the production traffic is owing to replication of the events in all the remaining replicas. For modelling it we count the number of branches in the replication tree and the distance of each branch. The distance of a branch, which depends on the number of replicas, is $\sqrt{A}/\sqrt{N_r}$. In addition, the number of messages per second is βN_p. Therefore the second part of the production traffic is

$$T_{p_2}(N_r) = \beta N_p e(N_r - 1)\frac{\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

Hence, the overall production traffic within the network is

$$T_p(N_r) = T_{p_1}(N_r) + T_{p_2}(N_r) = \beta N_p e \frac{\delta\sqrt{A}}{\sqrt{N_r}} + \beta N_p e(N_r - 1)\frac{\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

The consumption traffic is modelled like T_{p_1} since it also needs to measure the distance between a consumer and its closest replica. Thus, the overall consumption traffic in the network is

$$T_c(N_r) = \alpha N_c q \frac{\delta\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

doi: 10.1049/iet-com.2010.0574

Therefore the overall network traffic is

$$T(N_r) = T_p(N_r) + T_c(N_r) = \beta N_p e \frac{\delta\sqrt{A}}{\sqrt{N_r}} + \beta N_p e(N_r - 1)\frac{\sqrt{A}}{\sqrt{N_r}} + \alpha N_c q \frac{\delta\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

Then if we optimise the number of replicas that minimise the overall network traffic, the final result is

$$N_r^* = \delta\frac{\alpha N_c q}{\beta N_p e} - (1 - \delta)$$

Therefore in this application profile, the number of replicas that minimise the overall traffic depends on the ratio between the number of messages owing to consumption queries and the number of messages owing to production events, which is roughly the ratio between consumption and production traffic (without including the replication traffic).

5.2 Analytical model of ProdNonAggr application profile

The functionality of this profile is opposite to the previous case. In this case, producers store their events only in the closest replica (that does not replicate them). Now, consumers send their queries to its closest replica, that in turn forwards (replicates) the queries to the remaining replicas using the replication tree. To forward the query it is worth using the replication tree (it is nothing but a multicast tree) since this is a 1-to-N communication. However, the remaining replicas do not use the replication tree for answering back to the closest replica to the consumer, since it is an N-to-1 communication and the replication tree leads to longer paths than using the direct path from each of the remaining replicas to the first one. Therefore the other replicas send directly its local information to the closest replica to the querying consumer. When the closest replica has received all the information, it sends the information back to the consumer. Then, we define a new parameter necessary in this model:

- ε: Average number of messages per consumer query from each replica to the closest replica to the consumer. It is measured in messages/query.

The production traffic is because of production events being stored in the closest replica to the producer

$$T_p(N_r) = \beta N_p e \frac{\delta\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

However, in this case, the consumption traffic has three terms: (i) the traffic between the consumer and its closest replica, which generates α messages per query. (ii) The traffic as a result of forwarding the query from the closest consumer's replica to the rest of the replicas using the replication tree. In this case, one message is generated in each tree branch. (iii) Finally, the traffic caused as a result of the replies from each replica to the consumer's closest replica using shortest path routing. Each replica will generate on average ε reply messages per consumer query. Therefore the overall consumption traffic in the network is

$$T_c(N_r) = \alpha N_c q \frac{\delta\sqrt{A}}{\sqrt{N_r}} + N_c q(Nr - 1)\frac{\sqrt{A}}{\sqrt{N_r}} + \varepsilon N_c q(N_r - 1)\frac{\sqrt{A}}{2} \quad \text{msg} * \text{m/s}$$

Thus, the overall network traffic is

$$T(N_r) = T_p(N_r) + T_c(N_r) = \beta N_p e \frac{\delta\sqrt{A}}{\sqrt{N_r}} + \alpha N_c q \frac{\delta\sqrt{A}}{\sqrt{N_r}} + N_c q(Nr - 1)\frac{\sqrt{A}}{\sqrt{N_r}} + \varepsilon N_c q(N_r - 1)\frac{\sqrt{A}}{2} \quad \text{msg} * \text{m/s}$$

In this case finding the optimal number of replicas to minimise the overall traffic leads to a more complex process. The equation to be solved is

$$\varepsilon y^3 + y^2 + 1 + \delta\left(\alpha + \frac{\beta N_p e}{N_c q}\right) = 0$$

where $y = \sqrt{N_r}$. Then, by solving this cubic equation, we find that the optimal number of replicas that minimises the overall network traffic is

$$N_r^* = \left[\frac{1}{3\varepsilon}\left(-1 + \frac{2^{(2/3)} + A^{(2/3)}}{2^{(1/3)}A^{(1/3)}}\right)\right]^2$$

where A is

$$A = -2 - 27\varepsilon^2\left(1 + \delta\left(\alpha + \frac{\beta N_p e}{N_c q}\right)\right) + \sqrt{-4 + \left[-2 - 27\varepsilon^2\left(1 + \delta\left(\alpha + \frac{\beta N_p e}{N_c q}\right)\right)\right]^2}$$

Although this is a much more complex expression than the one obtained for the other profiles, it is still a closed formula that can be easily computed.

5.3 Analytical model of ConsAggr application profile

In this case, similar to ConsNonAggr application profile, consumers access their closest replica to retrieve the suitable information. However, in this case all the production events are not copied into the remaining replicas. Now each replica receives data events from producers in its surrounding area, and after some predefined time (which is a parameter of each specific application), it generates a replication event that aggregates all the available local information. Therefore this model introduces two new parameters:

- r: Replication process rate. Average number of replication-events per second and per replica. It is measured in replication-events/second.
- γ: Number of messages generated per replication-event. It is measured in messages/replication-event.

Table 2 Simulation parameters to evaluate the ConsNonAggr profile

Scenario	Figures	N_c	N_p	$\alpha\left(\frac{\text{msg}}{\text{query}}\right)$	$\beta\left(\frac{\text{msg}}{\text{event}}\right)$	$q\left(\frac{\text{query}}{s}\right)$	$e\left(\frac{\text{event}}{s}\right)$
1	Fig. 3*a*	800	200	2	1	1	1
2	Fig. 3*b*	500	500	10	1	1	1
3	Fig. 3*c*	500	200	10	1	5	1

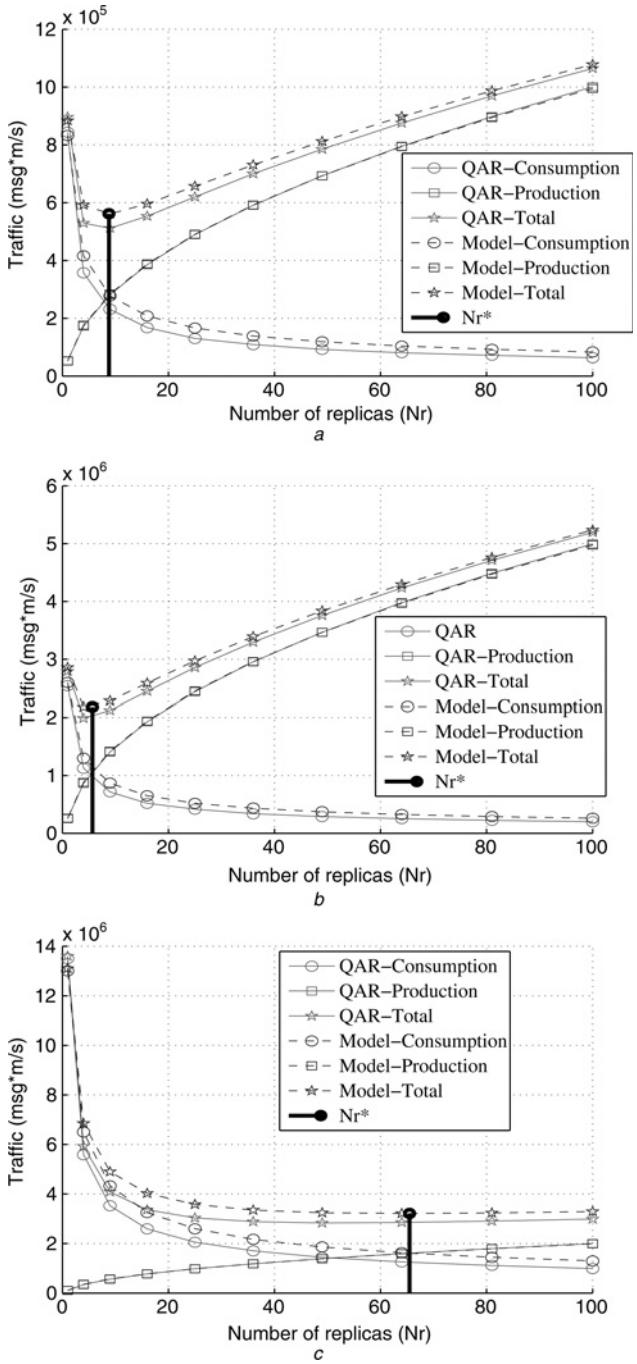

Fig. 3 *Validation of the ConsNonAggr profile model*
a $N_c = 800$, $N_p = 200$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$
b $N_c = 500$, $N_p = 500$, $\alpha = 10$, $\beta = 1$, $q = 1$, $e = 1$
c $N_c = 500$, $N_p = 200$, $\alpha = 10$, $\beta = 1$, $q = 5$, $e = 1$

The only difference between ConsNonAggr and ConsAggr profiles takes place in the replication traffic, since in ConsAggr the replicas do not replicate all the events received from producers, but they aggregate them, and thus, out of a large number of production events they produce just γ messages (γ could be 1).

Therefore the replication traffic (which is a part of the production traffic, T_{p_2}) is

$$T_{p_2}(N_r) = \gamma N_r r(N_r - 1)\frac{\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

Then, the production, consumption and overall traffic in the network for this application profile are

$$T_p(N_r) = \beta N_p e\frac{\delta\sqrt{A}}{\sqrt{N_r}} + \gamma N_r r(N_r - 1)\frac{\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

$$T_c(N_r) = \alpha N_c q\frac{\delta\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

$$T(N_r) = T_p(N_r) + T_c(N_r) = \beta N_p e\frac{\delta\sqrt{A}}{\sqrt{N_r}} + \gamma N_r r(N_r - 1)\frac{\sqrt{A}}{\sqrt{N_r}} + \alpha N_c q\frac{\delta\sqrt{A}}{\sqrt{N_r}} \quad \text{msg} * \text{m/s}$$

Then, the optimal number of replicas that minimise the overall traffic is

$$N_r^* = \frac{1}{6}\left(1 + \sqrt{1 + 12\delta\frac{\alpha N_c q + \beta N_p e}{\gamma r}}\right)$$

We obtain a very different expression than the one generated by the ConsNonAggr profile. In this case, the optimal number of replicas is roughly the square root of the ratio between the sum of consumer and production traffics and the replication traffic.

5.3.1 Aggregation factor (f): Applications that belong to this profile can be characterised by an aggregation factor, f. This factor measures the ratio between all the production traffic received by the replication nodes and the traffic generated by the replicas. Therefore the aggregation factor is defined as

$$f = \frac{\gamma N_r r}{\beta N_p e}$$

It goes from 0 to 1. Then, when f is closer to 0, it indicates that the replication traffic is much lower than the production traffic, so there is an important data aggregation performed by the replicas. However, if $f = 1$, it means that the replication traffic is the same than the traffic generated by producers and thus no aggregation is

doi: 10.1049/iet-com.2010.0574

Table 3 Simulation parameters to evaluate the ProdNonAggr profile

Scenario	Figures	N_c	N_p	$\alpha\left(\frac{\text{msg}}{\text{query}}\right)$	$\beta\left(\frac{\text{msg}}{\text{event}}\right)$	$q\left(\frac{\text{query}}{s}\right)$	$e\left(\frac{\text{event}}{s}\right)$	$\varepsilon\left(\frac{\text{msg}}{\text{query}}\right)$
1	Fig. 4*a*	100	1000	2	1	1	1	20
2	Fig. 4*b*	100	1000	2	1	1	1	1
3	Fig. 4*c*	100	1000	2	1	1	10	1

being performed by the replicas. In Section 6, we will discuss how this factor is related to the optimal number of replicas.

5.4 Analytical model of ProdAggr application profile

In this last application profile (similarly to ProdNonAggr), producer events are stored in the closest replica but not further replicated, whereas the closest replica to a consumer forwards (replicates) its queries to the remaining replicas. Then, the replication tree is used to forward the query (communication 1 to N). However, in this case the replication tree is also employed to forward the query replies. Thus, a replication node first aggregates its local information to calculate one or more parameters related to a given event type (e.g. maximum value). Therefore replicas that are leaves in the replication tree aggregate their local information and send one (or more) messages back using the replication tree. Intermediate replicas aggregate the information received from downstream replicas with its own local information and send the aggregated information upstream. Hence, the replica closest to the consumer receives aggregated information from all the other replicas, then it aggregates its own local information with the received information and sends the information to the consumer. Therefore it is necessary to introduce a new parameter for this model:

- θ: Average number of messages per consumer query in a replication tree branch. It is measured in messages/query.

Then, a replica could send a single message ($\theta = 1$) such as the maximum value of an event type in the network, or several messages, one per each statistical parameter of that event type.

In the ProdAggr profile the production traffic is the one generated from producers to the closest replica

$$T_{\text{p}}(N_{\text{r}}) = \beta N_{\text{p}} e \frac{\delta\sqrt{A}}{\sqrt{N_{\text{r}}}} \quad \text{msg} * \text{m/s}$$

In this case, the replication traffic is part of the consumption traffic, since what is sent through the replication nodes tree is the consumer query. Thus, the consumption traffic has several elements. First, the traffic generated between the consumer and its closest replica, which has already been modelled in the previous cases. Later, the traffic created in the replication tree is the sum of the traffic generated by the forwarded query, plus the traffic generated by the replies. As we already mentioned, intermediate replicas aggregate its own data with data from other replicas towards the replica that introduced the query in the tree. In average, θ messages traverse each branch of the tree for each query.

Therefore the overall consumption traffic for this application profile is

$$T_{\text{c}}(N_{\text{r}}) = \alpha N_{\text{c}} q \frac{\delta\sqrt{A}}{\sqrt{N_{\text{r}}}} + \theta N_{\text{c}} q (Nr - 1) \frac{\sqrt{A}}{\sqrt{N_{\text{r}}}} \quad \text{msg} * \text{m/s}$$

Then, the overall traffic becomes

$$T(N_{\text{r}}) = T_{\text{p}}(N_{\text{r}}) + T_{\text{c}}(N_{\text{r}}) = \beta N_{\text{p}} e \frac{\delta\sqrt{A}}{\sqrt{N_{\text{r}}}} + \alpha N_{\text{c}} q \frac{\delta\sqrt{A}}{\sqrt{N_{\text{r}}}} + \theta N_{\text{c}} q (Nr - 1) \frac{\sqrt{A}}{\sqrt{N_{\text{r}}}} \quad \text{msg} * \text{m/s}$$

Finally, the value of N_r that minimises the overall traffic for this application profile is

$$N_{\text{r}}^{*} = \delta \frac{\beta N_{\text{p}} e}{\theta N_{\text{c}} q} - \left(1 - \delta \frac{\alpha}{\theta}\right)$$

In this case, the optimal number of replicas depends on the ratio of production events to consumption queries. It is opposite to the ratio that modulates the optimal number of replicas in the ConsNonAggr profile.

5.4.1 Aggregation factor (f): An aggregation factor can also be proposed for this application profile. In this case, it is the number of messages that traverse each branch of the tree divided by the total number of production messages. Then, the aggregation factor is

$$f = \frac{\theta Ncq}{\beta Npe}$$

Again f goes from 0, the highest aggregation degree, to 1, the lowest aggregation degree.

Then, in this case, the optimal number of replicas can also be formulated as

$$N_{\text{r}}^{*} = \frac{\delta}{f} - \left(1 - \delta \frac{\alpha}{\theta}\right)$$

Therefore by knowing the aggregation factor of an application we can roughly estimate the optimal number of replicas. As it was expected, an aggregation factor close to 0 (high aggregation degree) increases the number of replication nodes deployed in the field, whereas an aggregation factor close to 1 (low aggregation degree) leads to use fewer replication nodes.

6 Model validation and discussion

In this section, we validate the models described for each application profile via simulation, and we discuss the main aspects of each model. We use a custom simulator that

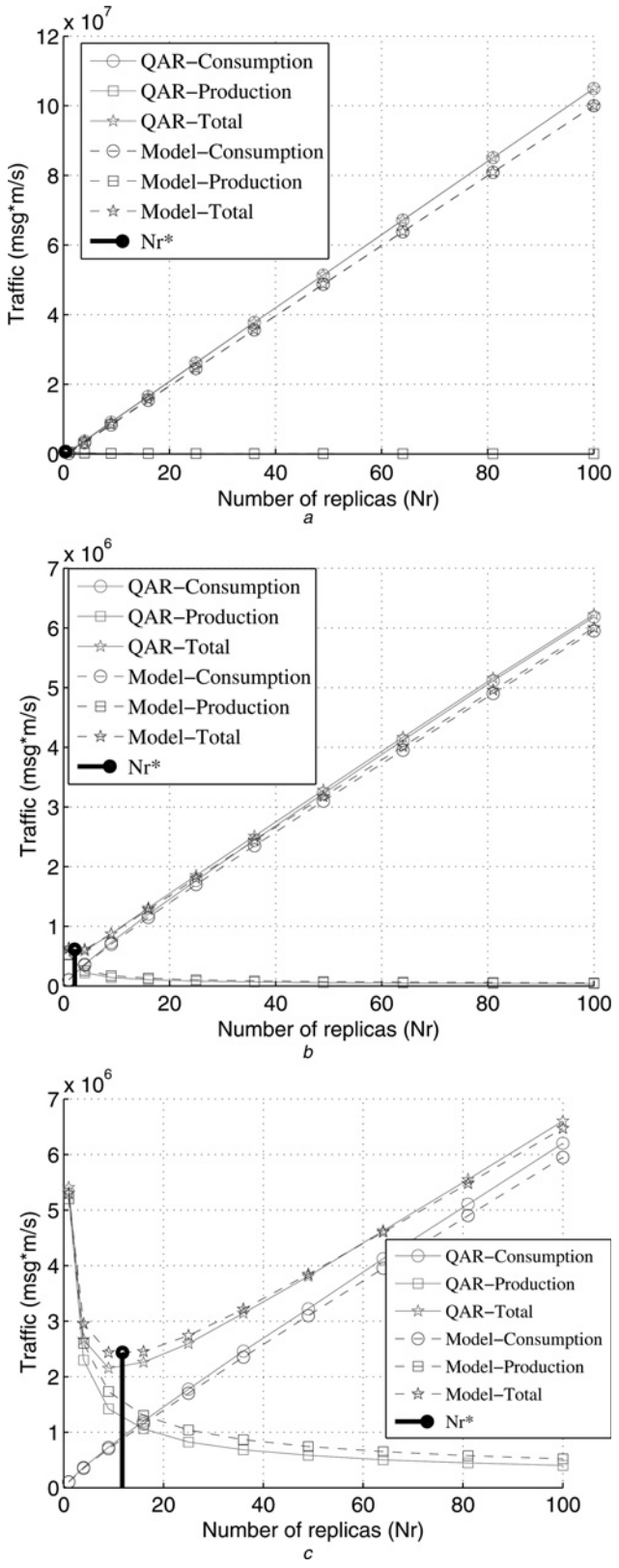

Fig. 4 *Validation of the ProdNonAggr profile model*
a $N_c = 100$, $N_p = 1000$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\varepsilon = 20$
b $N_c = 100$, $N_p = 1000$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\varepsilon = 1$
c $N_c = 100$, $N_p = 1000$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\varepsilon = 1$

accounts for the distances that the messages travel from consumers to replicas, from producers to replicas and inside the replication tree, and provides traffic results in terms of (messages*meter/second), which is the same metric used by our models.

In order to validate the models we use a network of area $1000 \times 1000\ m^2$, where 2000 nodes are deployed at random. We evaluate the traffic for different numbers of replicas following the QAR mechanism, that is, $N_r = 1$, 4, 9, 16, 25, 36, 49, 64, 81 and 100. For each number of replicas we run 50 different realisations and obtain the average traffic value (confidence intervals are too small to be seen).

6.1 ConsNonAggr model validation

We evaluate three different scenarios to validate the ConsNonAggr profile. Table 2 shows the simulation parameters used in each scenario as well as the index of the figure where the results are shown.

The first case is a scenario where consumption queries dominate production events owing to a larger number of consumers in the network. We observe that simulation results match the model and, what is more important, the optimal number of replicas provided by the model is also valid for the simulations. The second case demonstrates that consumption queries and their replies can dominate production events even though the number of producers is similar to that of consumers, since each consumption query could be replied with several data messages. Again, in this case the model is very accurate and using the optimal number of replicas leads to a very good result. Finally, the third scenario shows an application in which consumption queries and replies are much larger than production events. This implies an important increment of the consumption traffic, which leads to increase the optimal number of replicas to be used. Looking at the graphs is interesting to note that, in scenarios that require many replicas, selecting a slightly different number of replicas around the optimal value does not have a significant impact on the overall traffic. Then, in the third scenario the optimal number of replicas provided by the model is 65.48, but the traffic difference when selecting 36, 49, 64 or 81 replicas is less than 2.5%. Therefore in this kind of scenario where the ratio between consumption queries + replies and production events is high, misunderstanding the number of replicas by a small factor have a low impact on the traffic overhead.

6.2 ProdNonAggr validation

In order to validate this model we again consider different scenarios. Table 3 summarises the simulation parameters utilised for each scenario and indicates the figures where the simulation results for each scenario are compared to the ProdNonAggr model. In this case the parameter ε is a key factor, since the unicast communication from all the replicas to the closest one to the consumer using a direct path generates a major portion of the traffic.

The results show that the proposed model for the application profile ProdNonAggr is very accurate. In addition, to use more than one replica, the traffic because of production events has to be very high and the traffic because of query answers from all the replicas to the closest one to the consumer has to be lower. Owing to this reason, the first scenario provides an optimal number of replicas below 1, which leads to use a single home node in a real QAR system. The second scenario, which reduces 20 times the query answer from all the replicas to the one closest to the querying consumer, leads to $N_r^* = 2.13$ replicas that in a real QAR implementation would be 4. Finally, in a

doi: 10.1049/iet-com.2010.0574

Table 4 Simulation parameters to evaluate the ConsAggr profile

Scenario	Figures	N_c	N_p	$\alpha\left(\frac{\text{msg}}{\text{query}}\right)$	$\beta\left(\frac{\text{msg}}{\text{event}}\right)$	$q\left(\frac{\text{query}}{s}\right)$	$e\left(\frac{\text{event}}{s}\right)$	$r\left(\frac{\text{rep.event}}{s}\right)$	$\gamma\left(\frac{\text{msg}}{\text{rep.event}}\right)$
1	Fig. 5*a*	500	500	2	1	1	1	1	1
2	Fig. 5*b*	500	500	2	1	1	1	1	5
3	Fig. 5*c*	500	500	2	1	1	1	0.01	1

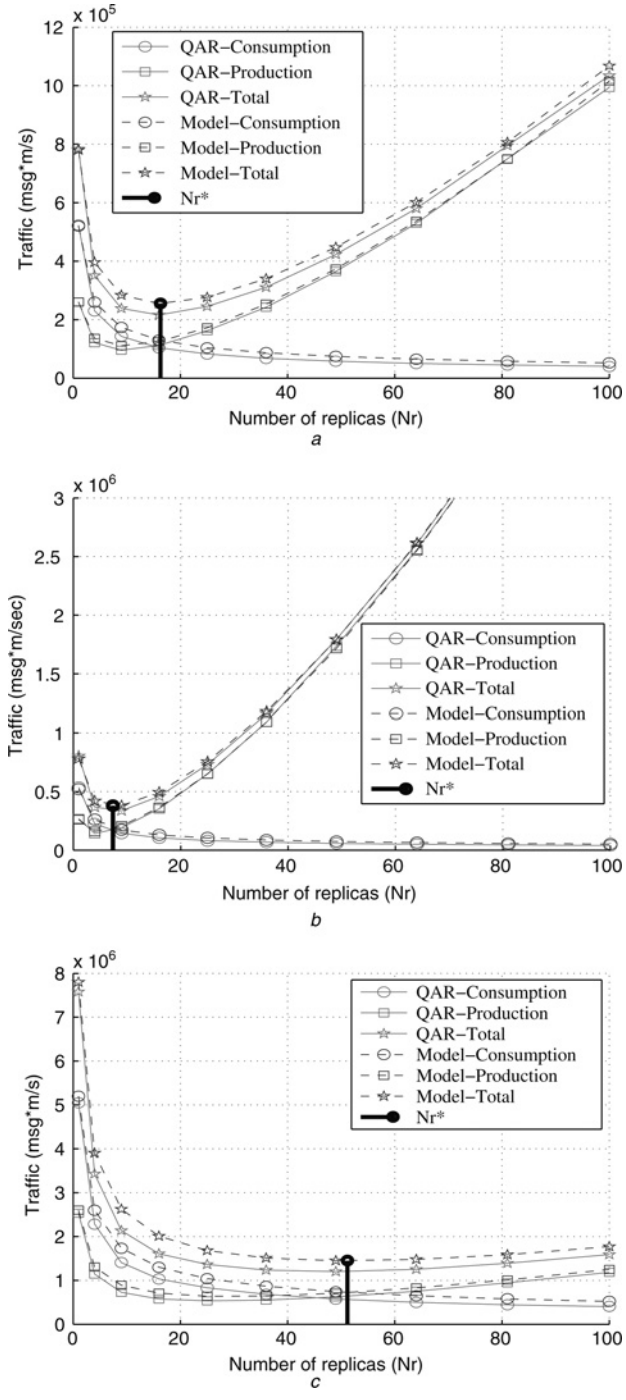

Fig. 5 *Validation of the ConsAggr profile model*

a $N_c = 500$, $N_p = 500$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\gamma = 1$, $r = 1$
b $N_c = 500$, $N_p = 500$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\gamma = 5$, $r = 1$
c $N_c = 500$, $N_p = 500$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\gamma = 1$, $r = 0.01$

scenario with a very high production traffic, the optimal number of replicas only increases up to $N_r^* = 11.75$, 9 being the practical value for QAR.

6.3 ConsAggr model validation

In this case the number of replicas depends on the sum of the traffic generated from producers and consumers to the closest replica in front of the replication traffic. This means that the larger the data aggregation, the higher the number of replicas to be used. Then, while validating the model accuracy we also check this fact. Hence, we evaluate three different scenarios whose parameters are defined in Table 4, which also indicates the figures associated with each scenario.

First of all, it must be highlighted that the model is very accurate to the simulations results, and thus the optimal number of replicas provided by the model is the right one for applications using the ConsAggr profile.

If we compare the aggregation factor for the optimal number of replicas in each of the simulated applications we find that: (i) in the first application the optimal number of replicas provided by the model is $N_r^* = 16.29$, which leads to use 16 replicas for QAR, and the aggregation factor in this case is, $f = 0.032$. (ii) In the second application we increment the number of messages generated per replication process up to 5. The optimal number of replicas is reduced to $N_r^* = 7.38$, which leads to use 9 replicas in a real deployment. In this case $f = 0.09$, three times higher that in the previous case. (iii) Finally, in the third case we use a ten times lower replication rate and just one message is sent per replication event. Thus, the optimal number of replicas increases up to $N_r^* = 51.16$, which is transformed to 49 replicas in a real scenario implementing QAR, with an aggregation factor of $f = 0.0098$. A clear conclusion is that when the aggregation factor decreases a higher number of replicas is obtained. This means that replicas are closer to producers and consumers, which directly implies a traffic reduction. Of course, since more replicas are deployed, the replication traffic increases because the replication tree length grows.

Finally, in all cases the number of producer events and consumer queries are the same, and so, the more the data aggregation, the lower overall network traffic. Then comparing the minimum traffic value in the different applications, the first one with a higher aggregation factor, presents three times more traffic than the third application; while the minimum traffic in the second application is the double that of the third application.

6.4 ProdAggr model validation

In this application profile the optimal number of replicas depends on the ratio between production events and consumption queries. In addition, the ratio is modulated by θ. Then we evaluate three different traffic scenarios to validate the model accuracy. Table 5 shows the simulation parameters used in the different scenarios and the index of the figures where each simulation is compared to the model.

Table 5 Simulation parameters to evaluate the ProdAggr profile

Scenario	Figures	N_c	N_p	$\alpha\left(\frac{\text{msg}}{\text{query}}\right)$	$\beta\left(\frac{\text{msg}}{\text{event}}\right)$	$q\left(\frac{\text{query}}{s}\right)$	$e\left(\frac{\text{event}}{s}\right)$	$\theta\left(\frac{\text{msg tree branch}}{\text{query}}\right)$
1	Fig. 6*a*	100	500	2	1	1	1	1
2	Fig. 6*b*	100	1000	2	1	1	3	1
3	Fig. 6*c*	100	500	2	1	1	1	3

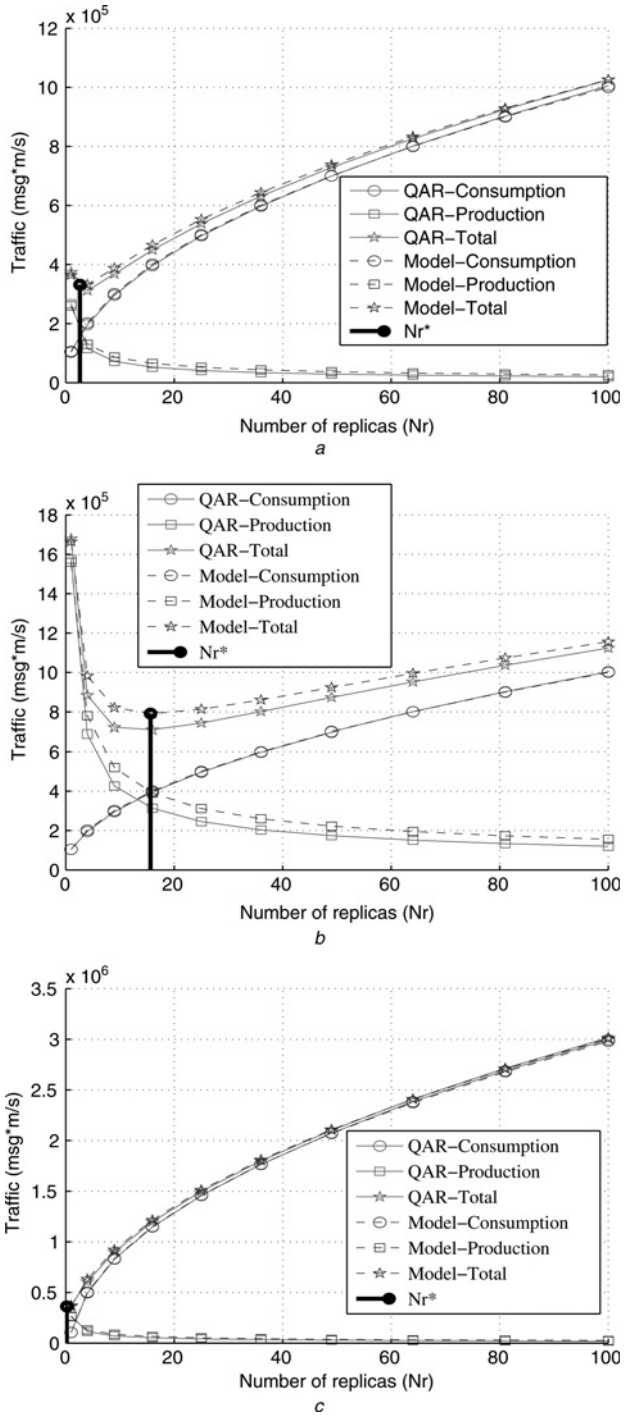

Fig. 6 *Validation of the ProdAggr profile model*

a $N_c = 100$, $N_p = 500$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\theta = 1$
b $N_c = 100$, $N_p = 1000$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 3$, $\theta = 1$
c $N_c = 100$, $N_p = 500$, $\alpha = 2$, $\beta = 1$, $q = 1$, $e = 1$, $\theta = 3$

First of all, it must be noted that the model is very accurate in all the different cases, and, what is more important, it provides the right number of replicas to be used.

The graphs show the expected results since when we increase the $N_p e/N_c q$ ratio the optimal number of replicas increases; however, if we increase θ, the number of replicas is reduced. Increasing θ means a lower aggregation degree ($f \rightarrow 1$), since all the information cannot be aggregated in a single message but several of them need to be generated.

7 Conclusions

We have introduced the analytical traffic models for data aggregation in Wireless Sensor and Actor Networks that use Data Centric Storage as storage and delivery mechanism. First of all, it must be highlighted that using multi-replication DCS mechanism is a must, since it performs much better than those works proposing to use a single rendezvous node. Furthermore, the results demonstrate that the larger the aggregation degree, the more replicas should be placed in the field to reduce the overall network traffic. The reason is that aggregation reduces the replication traffic, which is the largest one in multi-replication DCS systems.

8 Acknowledgments

This work was partially supported by the Spanish government through the T2C2 Project (TIN2008-06739-C04-01) and the regional government of Madrid through the MEDIANET (S-2009/TIC-1468) project.

9 References

1 Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: 'Wireless sensor networks: a survey', *Comput. Netw.*, 2002, **38**, (4), pp. 393–422
2 Demirkol, I., Ersoy, C., Alagoz, F.: 'MAC protocols for wireless sensor networks: a survey', *IEEE Commun. Mag.*, 2006, **44**, (4), pp. 115–121
3 Akkaya, K., Younis, M.: 'A survey on routing protocols for wireless sensor networks', *Ad Hoc Netw.*, 2005, **3**, (3), pp. 325–349
4 Al-Karaki, J.N., Kamal, A.E.: 'Routing techniques in wireless sensor networks: a survey', *IEEE Wirel. Commun.*, 2004, **11**, (6), pp. 6–28
5 Wang, C., Sohraby, K., Li, B., Daneshmand, M., Hu, Y.: 'A survey of transport protocols for wireless sensor networks', *IEEE Netw.*, 2006, **20**, (3), pp. 34–40
6 García, C., Ibargengoytia, P., García, J., Pérez, J.A.: 'Wireless sensor networks and applications: a survey', *Int. J. Comput. Sci. Netw. Secur.*, 2007, **7**, (3), pp. 264–273
7 Verdone, R., Dardari, D., Mazzini, G., Conti, A.: 'Wireless sensor and actuator networks' (Elsevier, 2008)
8 Akyildiz, I.F., Kasimoglu, I.H.: 'Wireless sensor and actor networks: research challenges', *Ad Hoc Netw.*, 2004, **2**, (4), pp. 351–367
9 Di Pietro, R., Mancini, L.V., Soriente, C., Spognardi, A., Tsudik, G.: 'Catch me (If You Can): data survival in unattended sensor networks'. Proc. 2008 Sixth Annual IEEE Int. Conf. on Pervasive Computing and Communications, PERCOM'08, Washington, DC, USA, 2008, pp. 185–194
10 Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., Estrin, D.: 'Data-centric storage in sensornets', *SIGCOMM Comput. Commun. Rev.*, 2003, **33**, (1), pp. 137–142
11 Joung, Y.J., Huang, S.H.: 'Tug-of-war: an adaptive and cost-optimal data storage and query mechanism in wireless sensor networks'. Proc.

Fourth IEEE Int. Conf. on Distributed Computing in Sensor Systems, DCOSS'08, Berlin, Heidelberg, 2008, pp. 237–251
12 Cuevas, A., Urueña, M., Romeral, R., Larrabeiti, D.: 'Data centric storage technologies: analysis and enhancement', *Sensors*, 2010, **10**, (4), pp. 3023–3056
13 Ahn, J., Krishnamachari, B.: 'Fundamental scaling laws for energy-efficient storage and querying in wireless sensor networks'. Proc. Seventh ACM Int. Symp. on Mobile Ad Hoc Networking and Computing, MobiHoc'06, New York, USA, 2006, pp. 334–343
14 Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W., Tiny, D.B.: 'An acquisitional query processing system for sensor networks', *ACM Trans. Database Syst.*, 2005, **30**, pp. 122–173
15 Amato, G., Chessa, S., Vairo, C.: 'MaD-WiSe: a distributed stream management system for wireless sensor networks', *Softw. Pract. Exp.*, 2010, **40**, pp. 431–451
16 Karp, B., Kung, H.T.: 'GPSR: greedy perimeter stateless routing for wireless networks'. Proc. Sixth Annual Int. Conf. on Mobile Computing and Networking, MobiCom'00, New York, USA, 2000, pp. 243–254
17 Ee, C.T., Ratnasamy, S., Shenker, S.: 'Practical data-centric storage'. Proc. Third Conf. on Networked Systems Design & Implementation, NSDI'06, USENIX Association, Berkeley, CA, USA, 2006
18 Zhao, Y., Chen, Y., Ratnasamy, S.: 'Load balanced and efficient hierarchical data-centric storage in sensor networks'. Fifth Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks, SECON'08, New York, USA, 2008, IEEE, pp. 560–568
19 Sarkar, R., Zhu, X., Gao, J.: 'Double rulings for information brokerage in sensor networks'. Proc. 12th Annual Int. Conf. on Mobile Computing and Networking, MobiCom'06, New York, USA, 2006, pp. 286–297
20 Albano, M., Chessa, S., Nidito, F., Pelagatti, S.: 'Data centric storage in non-uniform sensor networks'. Proc. Second Int. Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID 2007), Santa Margherita Ligure (Genova), Italy, 2007
21 Albano, M., Chessa, S., Nidito, F., Pelagatti, S.: 'Q-NiGHT: adding QoS to data centric storage in non-uniform sensor networks'. IEEE Int. Conf. on Mobile Data Management, 2007, pp. 166–173
22 Ratnasamy, S., Karp, B., Yin, L., *et al.*: 'GHT: a geographic hash table for data-centric storage'. Proc. First ACM Int. Workshop on Wireless Sensor Networks and Applications, WSNA'02, New York, USA, 2002, pp. 78–87
23 Ratnasamy, S., Karp, B., Shenker, S., *et al.*: 'Data-centric storage in sensornets with GHT, a geographic hash table', *Mob. Netw. Appl.*, 2003, **8**, (4), pp. 427–442