UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

DISEÑO, ANÁLISIS E IMPLEMENTACIÓN DE UN MARCO DE TRABAJO DE ALMACENAMIENTO CENTRADO EN LOS DATOS CON ADAPTACIÓN ESPACIO-TEMPORAL Y MULTI-REPLICACIÓN PARA REDES DE SENSORES Y ACTUADORES INALÁMBRICAS

Autor: **Ángel Cuevas Rumín**
Ingeniero en Telecomunicación, Máster en Ingeniería Telemática

Director: **Manuel Urueña Pascual**
Doctor en Tecnología en de las Comunicaciones

Leganés, Febrero de 2010

UNIVERSIDAD CARLOS III DE MADRID

DEPARTMENT OF TELEMATICS ENGINEERING

Ph.D. THESIS

DESIGN, ANALYSIS AND IMPLEMENTATION OF A SPATIAL-TEMPORAL ADAPTIVE MULTI-REPLICATION DATA CENTRIC STORAGE FRAMEWORK FOR WIRELESS SENSOR AND ACTOR NETWORKS

Author: **Ángel Cuevas Rumín, M.Sc.**

Supervisor: **Manuel Urueña Pascual, Ph.D.**

Leganés, February 2010

DESIGN, ANALYSIS AND IMPLEMENTATION OF A SPATIAL-TEMPORAL ADAPTIVE MULTI-REPLICATION DATA CENTRIC STORAGE FRAMEWORK FOR WIRELESS SENSOR AND ACTOR NETWORKS

Autor: Ángel Cuevas Rumín

Director: Dr. Manuel Urueña Pascual

Firma del tribunal calificador:

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Calificación:

Leganés, ___ de _____ de _____.

A mis padres Loli y Ángel,
a mis hermanos Arantxa y Rubèn,
a mi pareja Ana,
a mi yaya Chon,
y a mi sobrino Celso.

Que con cuidados, la flor florece.

– Manolo García, Niña Candela (2004).

Un pueblo que no sabe leer ni escribir
es un pueblo fácil de engañar .

– Ernesto Che Guevara (1928-1967)

Agradecimientos

La culminación de esta Tesis Doctoral es el final de un proceso largo de formación con algunos momentos difíciles, pero sin duda una gran cantidad de satisfacciones. Este proceso no habría sido posible sin todas esas personas que me rodean y me quieren y me dan un apoyo incondicional cada día. Por tanto esta Tesis pertenece también a todos y cada uno de ellos.

En primer lugar quiero acordarme de mis padres, Loli y Ángel. Ellos son los responsables de quién soy hoy en día, de los valores que poseo, y del profundo orgullo que siento por ellos. Mi madre, Loli, ha sabido mostrarme el amor incondicional, la fortaleza de la honestidad y me ha enseñado que el mayor de los triunfos es irme cada noche a la cama con la conciencia tranquila. Ha sido, es y será un apoyo del cual no podría prescindir. Mi padre me ha enseñado como el esfuerzo del día a día merece la pena y para mí se ha convertido por méritos propios en ese oráculo en el cual verter todas tus dudas y al que ves como una fuente de sabiduría inagotable, siendo sus consejos de un valor incalculable. Pero más allá de todo eso él ha estado siempre ahí y sé que siempre estará.

A mi hermano Rubén decirle que es la persona que mejor me conoce y que más sabe de mí. Lo hemos compartido todo, y sé que aún nos queda mucho por compartir. Yo fui una persona muy afortunada, porque pocas personas tienen la suerte de nacer con su mejor amigo al lado, y además que ese mejor amigo se convierta en un pilar fundamental para tí porque sabes que nunca te fallará, y siempre será una referencia en el camino a seguir. Hemos pasado mucho juntos, pero estoy seguro que lo mejor está por llegar.

Mi hermana Arantxa siempre ha estado ahí, con una sonrisa cuando hacía falta, y siendo, sin ella saberlo, una referencia para mí, porque es un orgullo tener una hermana como ella, y sentir su cariño día a día durante muchos años. Además, nos ha regalado a todos esa joyita que es Celso, mi sobrino, que ha llenado de alegría e ilusión nuestra famillia, y verle crecer día a día es el mejor de los regalos.

A Ana, mi pareja, mi chica, sólo puedo decirla gracias. A su lado he vivido un año maravilloso, ha conseguido darme mucha felicidad, y enseñarme a amar de una manera serena y relajada, en la cual el día a día se convierte en un auténtico placer, cada pequeña cosa es preciosa si la comparto con ella. Y además gracias a ella esa compañera llamada

Doris ha entrado en mi vida.

A María y Celso, quiero agradecerles lo felices que hacen a mis hermanos, y decirles que hoy son una parte más de mi familia que la completa y la hace más valiosa.

Mi yaya Chon es la mejor persona que he conocido, y una de las personas que más quiero. Su bondad infinita te hace entender como deberíamos ser, y que este mundo sería un lugar mucho mejor si hubiese más personas como ella.

Quiero acordarme también de mis abuelos, Lumi que ha sido siempre alguien especial para mi, de mi abuela Pilar y de todos mis tíos, Fali, Javi, Mari, Loli, Pili y Carlos.

Y por supuesto no puedo olvidar a mis amigos, Jorge, Rafa, Toñín, Victor, Diego, Alfonso, Fernando, Javitxu, que son parte de lo que soy y de lo que quiero seguir siendo.

Agradezco también haber conocido a esas personas que están en tu vida o aparecen en tu camino y te dejan una huella, por su gran fortaleza vital, o porque te enseñan a mirar el mundo de otra manera. Me acuerdo de vosotros Soco, Smriti, Paul, Vicky, Maribel.

En lo profesional quiero agradecer especialmente a David la confianza que depositó en mí hace más de 4 años. Él me ha allanado el camino muchas veces y me ha permitido desarrollarme como investigador en un entorno agradable y lleno de confianza. Sin duda me gustaría que todos mis jefes en el futuro fueran como él y me pusiesen las cosas tan fáciles.

Manolo ha sido la persona que me ha guiado y la que más me ha ayudado en este proceso supervisando mi trabajo, entendiéndome y apoyádome en todo momento. A parte de un gran tutor de Tesis, Manolo ha sido un mejor compañero, y gracias a él hoy todo ésto es posible.

No quiero olvidarme de una persona que quizá sin él saberlo ha jugado un papel fundamental en mi motivación y en eseñarme la belleza que se esconde tras la carrera del investigador. Esa persona es Gustavo. Él me ha servido de inspiración, y sin duda es el referente más claro que tengo en cuanto a que tipo de investigador quiero ser, no sólo por su notable éxito, sino por su humildad y la gran persona que me ha demostrado ser.

Quiero recordar también a mis compañeros de grupo, Jose, Richi, Isaac, Gerson, Carlos, y a los de departamento, Mika, Issaías, Carmen, Albert, Marco, Andrés, Chema, Roberto, Paul, Alex, Iljitsch, Alberto, Arturo, María, Goyo, Pablo, Carlos Jesús, Ignacio, Marcelo y Antonio.

Finalmente, quiero agradecer a todas las personas de las que me olvido y que en algún momento de mi vida me han arrancado una sonrisa y han hecho que pase momentos agradables.

Abstract

This PhD Thesis presents a novel framework for Data-Centric Storage (DCS) in a Wireless Sensor and Actor Network (WSAN) that enables the use of a multiple set of data replication nodes, which also *change over the time*. This allows reducing the average network traffic and energy consumption by *adapting* the number of replicas to applications' traffic, while balancing energy burdens by varying their location. To that end we propose and validate a simple model to determine the optimal number of replicas, in terms of minimizing average traffic/energy consumption, from the measured applications' production and consumption traffic. Simple mechanisms are proposed to decide when the current set of replication nodes should be changed, to enable new applications and sensor nodes to efficiently bootstrap into a working sensor network, to recover from failing nodes, and to adapt to changing conditions. Extensive simulations demonstrate that our approach can extend a sensor network's lifetime by at least a 60%, and up to a factor of 10x depending on the lifetime criterion being considered. Furthermore, we have implemented our framework in a real testbed with 20 motes that validates in a small scenario those results obtained via simulation for large WSANs. Finally, we present a heuristic that adapts our framework to scenarios with spatially heterogeneous consumption and/or production traffic distributions providing an effective reduction in the overall traffic, as well as reducing the number of nodes that die over the time.

Keywords: Wireless Sensor and Actor Network (WSAN), Data-Centric Storage (DCS), Quadratic Adaptive Replication (QAR), Random Multi-Replication, spatio-temporal adaptation.

Resumen

Esta Tesis se enmarca en el campo de la redes de sensores y actuadores inalámbricas. Para este tipo de redes existe un sistema de almacenamiento y entrega de información totalmente distribuido denominado D*ata-Centric Storage* (DCS). En dicho sistema se selecciona un nodo en la red para almacenar toda la información relativa a una aplicación o tipo de evento. Dicha elección se realiza mediante el uso de una función de *hash* que, usando como argumento el propio nombre de la aplicación (o tipo de evento), devuelve el identificador (e.g. coordenadas geográficas, identificador de nodo, etc) del nodo responsable de almacenar toda la información que deesa aplicación (o tipo de evento).

El uso de un único nodo para almacenar todos los datos de un mismo tipo generados en la red tiende a generar un punto de saturación en la red (especialmente en términos energéticos) ya que una gran cantidad de tráfico es encaminada hacia un único punto. De hecho, no sólo el nodo seleccionado como nodo de almacenamiento, sino también todos aquellos que le rodean, experimentan un mayor gasto de recursos ya que son los encargados de rutar los mensajes hacia el nodo de almacenamiento.

Este problema ha dado lugar a sistemas que utilizan multiples replicas para aliviar la generación de un punto de congestión y elevado consumo energético en la red. Situando varios puntos de almacenamiento para un tipo de evento dado, es posible aliviar la congestión de un único punto. Sin embargo la generación de nuevas réplicas tiene un coste asociado, y por tanto existe un número de replicas óptimo que minimiza el tráfico total en la red, que a su vez tiene un impacto directo en la reducción del consumo energético y la extensión del tiempo de vida de la red. En esta Tesis se proponen dos esquemas de replicación para redes de sensores que usan DCS como sistema de almacenamiento distribuido. Para ambos casos se han desarrollado modelos matemáticos que permiten conocer el número óptimo de réplicas que deben ser utilizadas (para minimizar el tráfico total en la red) en función de la intensidad de producción y consumo de un tipo de evento. El primer mecanismo, denominado *Quadratic Adaptive Replication* (QAR), propone el uso de una estructura mallada para la colocación de las réplicas. QAR mejora trabajos previos que ya proponían un esquema de replicación en *grid*, ya que es más adaptativo a las condiciones de tráfico en la red. El segundo mecanismo simplemente genera localizaciones aleatorias dónde situar las replicas. Sorprendentemente, esta Tesis demuestra que es el mejor sistema de replicación, incluso por delante de QAR, ya

que es el más adaptativo a las condiciones de tráfico. Además, tiene la gran ventaja de que es extremadamente simple y puede aplicarse en redes irregulares o que utlizan diferentes protocolos de enrutamiento.

Los sistemas de replicación alivian el problema del punto único de congestión, pero no lo solucionan completamente, ya que siguen apareciendo puntos de congestión menores, tantos como replicas sean usadas. Por tanto, la red sigue presentando una gran desigualdad en el consumo energético, ya que aquellos puntos seleccionados como réplicas (y sus vecinos) usan una mayor energía para desarrollar su actividad. Frente a este problema, se propone como solución el cambio de las réplicas a lo largo del tiempo. Esecialmente, se limita el tiempo que un nodo puede permanecer desempeñando el papel de réplica, de tal forma que, una vez pasado ese tiempo, otro nodo tomará esa responsabilidad. Aplicando esta propuesta se consigue un equilibrio en el consumo energético de los nodos de la red, lo que tiene un gran impacto en la extensión del tiempo de vida de la red. En los experimentos realizados, dicha extensión tiene un valor mínimo de un 60%, llegándose a extender el tiempo de la vida hasta 10 veces bajo ciertas definiciones de tiempo de vida de la red.

La principal contribución de esta Tesis es la presentación de un marco de trabajo adaptativo tanto espacial como temporalmente que, basado en un modelo teórico, indica cuál es el número óptimo de replicas que deben ser usadas en un determinado periodo. En esta Tesis se propone un protocolo completo que cubre todas las funcionalidades para que dicho sistema pueda ser implementado y desplegado en el mundo real.

Para demostrar que el sistema propuesto puede ser implementado en ndoos de sensores comerciales, esta Tesis presenta la implementación realizada en 20 motas del fabricante Jennic. Asimismo, se ha empleado un pequeño test de pruebas para confirmar la validez de los modelos matemáticos para la obtención del número óptimo de réplicas, así como para demostrar que el cambio de las replicas a lo largo del tiempo genera una mejor distribución del consumo energético en la red.

Contents

List of Figures

List of Tables

Part I

Motivation and Introduction

Chapter 1

Motivation and Introduction

1.1 Motivation

Wireless Sensor Networks (WSNs) have focused the interest of both the industry and academia in the last decade. A wireless sensor is a low cost device 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, to transmit it by relay nodes and to process it in order to get a common goal, a Wireless Sensor Network is obtained. The main characteristic of WSNs is that their nodes are battery-powered, thus being limited in energy as well as in processing and storage (e.g., memory limited). These limitations have differentiated WSN as an entire research area inside the wireless ad hoc network one. Therefore, new protocols and algorithms have been proposed for WSN at all levels of the network stack, from the physical to the application layer [ASSC02]. In particular, in [DEA06] we can find an overview of the main medium access control (MAC) protocols proposals for WSNs, whereas the most important routing proposals for WSNs are presented in [AY05, AKK04]. WSN-specific transport protocols are summarized in [WSL+06]. Finally, some examples of WSNs applications are described in [GhIgGhPd]. Furthermore, the interest of the industry on WSN has pushed the standardization of lightweight and low power communications technologies that are suitable for sensor nodes. For instance, the Institute of Electrical and Electronic Engineers (IEEE) has specified the 802.15.4 [oEE06] standard that covers the physical and MAC layers. On top of 802.15.4, the Zigbee Alliance [All05] has defined a protocol stack covering routing and application layers for WSNs. Also, the Internet Engineering Task Force (IETF) has defined in RFC 4944 [MKC07] to allow IPv6 datagram transmission over an IEEE 802.15.4 network.

Wireless Sensor Networks (WSNs) have been told to be one of the key technologies in the next decade, which is starting nowadays. In particular, some people one decade ago envisioned sensor nodes deployed everywhere, communicating together and generating a huge amount of valuable data for nowadays information society.

Indeed, quite recently, at the beginning of 2010, the WSN section of a company called Crossbow has been sold for 18 million dollars to another company called MSMIC. This reflects that although WSN have not been deployed everywhere yet, IT companies are looking at them as a profitable technology within the next few years.

Moreover, a lot of research funds have been provided in Europe during the last decade (FP5, FP6 and FP7 programmes) to further develop WSN technology and push it towards its integration with the Information Society. Some examples of integrated projects with a consortium formed by relevant European universities and companies which are very related to WSNs are: SENSOR, RUNES, CoBIs, WASP, GINSENG, etc.

Therefore, all these reasons have made us to focus our interest in WSN technologies. However, WSNs is a quite wide research field so we have focused our attention to a particular research area in order to contribute with novel solutions.

One of the most interesting WSNs research lines we found is the so called Data-Centric paradigm. The main idea underlying Data-Centric proposals is that for many WSN applications what is really important is not who generates the data, but the nature of the data the WSN gathers (albeit where and when the data was obtained could be also important). For instance, a node in a sensornet could be interested in establishing communication paths only with nodes that measure a given type of data (e.g., temperature).

Under this definition, some works like Direct Diffusion [IGE00,IGE$^+$03] defined a Data-Centric Routing, where routing tables are created in function of nodes interest in particular data types. Therefore, a node forwards a particular message to a neighbour only if it is interested in the data contained in the message or it is the next hop towards an interested node.

Besides routing, storage is another widely studied topic in WSN research because the obtained data need to be stored in some way before processing it. There are two well-known approaches, plus a third one that has been recently proposed:

- Local Storage, that is, the same node measuring the physical phenomenon stores the data. Then, some protocol should be defined to allow potential consumers of those data to find and access the nodes where such data is stored. The easiest way to reach all nodes of interest would be to use flooding within the network, but broadcasting is very energy demanding for a WSN formed by battery powered nodes

- External Storage, which usually assumes that WSNs always have a central node (called *sink* or *base station*) that manages the whole network and acts as an access point for external users. Therefore, when a sensor measures a physical event it could push it to the sink that then stores it for the external WSN users to access it.

- In-network Storage proposes to keep the generated information in some node inside the network, although not necessarily in the one that produced the information. Therefore, a very interesting solution could be integrating together in-network storage with the Data-Centric paradigm. With this approach, a *rendezvous node* within the sensornet is chosen in order to store the information. That rendezvous node is selected

based on the data (or event) type, instead of being a predefined one as in the previous sink case, thus, being considered a Data-Centric mechanism. This alternative storage mechanism is named Data Centric Storage (DCS). In DCS, when a sensor measures a physical event for a particular data type, it pushes the information towards the rendezvous node in charge of that data type. In turn, consumers interested on that data type query that particular rendezvous node to retrieve the information.

Traditionally, many WSN works assume that the only node querying the network in order to gather the stored information is the base station node, sometimes on behalf of external users. In such a case, it seems that the best option could be to use external storage, so that the base station is able to process all the suitable information locally. However, in many cases there could be lot of useless data or events that are sent to the base station, which later discards them because they are never used. Therefore, there could be some resource consumption and energy waste on sending such information. Under these conditions local storage or in-network (i.e DCS) storage could outperform external storage.

In addition, novel WSNs paradigms have been proposed in the last few years that led us to further investigate DCS as an efficient alternative to local and external storage.

- Unattended Wireless Sensor Networks are deployed in areas with difficult access and in many cases without any possibility of external communication. Therefore, data is collected after long time periods, thus the network should be able to store the information until certain collection time. Under this conditions, using a base station as single storage point leads to a non-fault tolerance solution. For instance, (i) in case the base station fails all the information is lost; (ii) if during some periods the paths towards base station from some network nodes are not available (e.g. battery of relay nodes is exhausted), the associated information of that region for that period is also lost, etc. Therefore, in such case it seems interesting to exploit the combined storage capacity that sensor nodes offer, either through local storage or using some kind of in-network storage such as DCS.

- Wireless Sensor and Actor Networks (WSANs) present a new type of node, the *actuator* or *actorf*. An actor node consumes the data generated within the WSN in order to perform some action. Therefore, in this kind of networks the base station is not the only data consumer anymore, but also the actor nodes themselves. Hence, external storage is not optimal at all for WSANs because: (i) using a single storage node will create a routing hot spot, (ii) depending on the base station position within the network, forcing all consumer nodes to access the base station to get some information could lead to use very long paths, which consume a lot of resources and energy. Hence, we strongly believe that in networks where there are quite a lot of potential consumer nodes, such us WSANs, DCS could be the best solution in order to provide an effective storage mechanism.

Therefore, all the previous arguments motivated us to further investigate the Data Centric Storage paradigm. However, from the very first moment we were concerned on not only to focus on a theoretical analysis, but to also have in mind that our proposals should take into

account practical issues, so that our proposals could be implemented in commercial sensor motes.

1.2 Introduction

In this Thesis we consider a simple framework to build a distributed information delivery service for one or more applications running over a Wireless Sensor and Actor Network (WSAN). Each application is modelled as a distributed set of *producer* and *consumer* nodes, e.g., sensors or actuators that exchange information by relaying packets across neighbouring nodes. We assume that producer and consumer nodes do not have explicit knowledge of each other, but are aware of the name(s) of the application(s) in which they are participating. This makes possible to build a highly scalable distributed information service involving large numbers of producers and consumers.

Data-Centric Storage (DCS) [SRK+03, RKY+02, RKS+03] is an elegant solution to this problem. The key idea is to identify one node in the network which serves as a rendezvous point between producers and consumers associated with an application. This node is determined by generating a spatial location based on applying a hash function to the application's name, and then finding the node in the network which is the closest to it. Thus producers and consumers, which have knowledge of the hash function and application's name, are able to determine and route to the common rendezvous point without any additional information. Then, a producer pushes new information to the *rendezvous node*, which, in turn, is responsible for storing (and possibly aging) data. Consumers are able to subsequently pull information from the same rendezvous node.

However, using a single rendezvous node could lead to create a hot spot in the network, especially if it is serving as the rendezvous node of a popular application. In addition, the problem is that, not only the selected rendezvous node spends its battery rapidly, but also its surrounding area will be overloaded since lot of routing paths are established through that zone in order to reach the rendezvous node.

Therefore, we propose to use several rendezvous nodes (also called *replication nodes* or *replicas*) in order to store the data of a given event type. In that way, we mitigate the hot spot problem, although it is not fully eliminated. For that purpose, we propose two novel multi-replication DCS mechanisms to allocate several rendezvous nodes within a WSAN:

- **Quadratic Adaptive Replication (QAR)**: Some of the most relevant multi-replication DCS proposals in the literature focus on placing replication nodes creating a grid structure. However, we realize that those approaches are not adaptive enough since they are bounded to a reduced set of values to be used as the number of rendezvous nodes (i.e. 1, 4, 16, 64, 256, etc). QAR also proposes to use a grid structure but provides a more adaptive solution, increasing the set of replication nodes values that can be used within the network (i.e. 1, 4, 9, 16, 25, 36, 49, 64, etc). This thesis further demonstrates that QAR outperforms all previous proposals in the literature in terms of traffic cost. When utilizing QAR the location of the first replica is generated

at random by using a hash function over the application's name $hash(app)$ and then the location of the other replication nodes is computed to generate a grid structure.

- **Random Multi-Replication**: We also propose a very simple mechanism that allocates rendezvous nodes at random within the network. We have chosen random placement because it is the simplest way of computing the location for any given number of replication nodes. Furthermore, it allows to use whatever number of replication nodes since it is not bounded to any fixed structure. In fact, we demonstrate in this Thesis that random replication outperforms previous multi-replication DCS proposals (including QAR) since it generate the lowest overall network traffic under the same consumption and production traffic conditions. In this case, the location of the nodes is generated by using several times the same hash function that uses as input parameter the application's name and a replica id number that goes from 1 to N_r, $hash(app \oplus i) \ \ \forall i \in [1, N_r]$. Therefore, the cost of computing the random locations is very low.

Hence, for both proposals QAR and Random Multi-Replication, we consider the case where nodes can determine the current set of N_r replicas associated with a given application just by generating N_r spatial locations with a hash function. However, those spatial locations unlikely match the actual location of any of the network nodes. Therefore, the network nodes that are the closest ones to these hashed spatial locations serve as rendezvous (or replication) nodes for that application.

In this setting any producer or consumer, which is also aware of the application's name and N_r, can independently determine the location of the 'nearest' replication point by determining the minimum distance between itself and all the spatial locations generated by the hash function. Then, it will route its messages towards the closest hashed location, and the routing layer will forward the message to the nearest node to these coordinates, that is, the closest rendezvous node.

In cases where consumption queries intensity dominates production events generation, we consider a Data-Centric Storage framework where application's information is pushed, stored and/or replicated across all the rendezvous points. This permits consumers to pull information from rendezvous points that are closer, thus reducing network traffic, energy overheads and reducing response time, while also improving fault-tolerance in the case where some nodes fail or run out of energy.

However, in the opposite case, when production dominates consumption, replicating all the events across the multiple replication nodes when there are just few queries in the network is a bad idea. In this case, we propose to store the data *solely* at the closest rendezvous node, and thus consumers need to query all the rendezvous nodes for possible data.

Then, if we think in the case when consumption dominates production (a similar reasoning could be done with the opposite case), closeness between consumers and replication nodes is beneficial from the point of view of reducing traffic to consumers, energy expenditures and delay to access the data. However, if a large number of replication nodes is employed, the production costs, including the cost to transport and store information across

multiple replication nodes can be high. Thus a key trade-off in our framework is to decide how many replication nodes should be used.

Then for both, QAR and Random multi-replication, we show precisely how this trade-off can, and should, be optimized so as to minimize the total network traffic, in bits-meter/second, and thus, to first order, also minimize the overall energy consumption of a given application. The optimal number of rendezvous nodes depends on the ratio of the production intensity to that of consumption, i.e., is critically dependent on the traffic associated with the application.

Although mitigated by the use of multiple rendezvous nodes, a node that serves as a replication point, will experience a higher traffic load associated with processing consumption and production messages, and thus its energy reserves will be depleted at a higher rate. This is also the case for relay nodes that serve to transport information among replication points. Thus, it is desirable to balance such roles among all network's nodes. In order to solve this issue, we propose to change the replication nodes set over the time. To this end, application's timeline is subdivided into *epochs*. During each epoch a new set of replication points is randomly selected. Therefore, the hash function needs a new parameter that identifies the current epoch in order to allow consumers and producers to compute the right rendezvous locations at any time. Therefore, in the Random Multi-Replication framework that function would be $hash(app \oplus epoch \oplus i) \ \ \forall i \in [1, N_r]$. Moreover, in each epoch one can, not only to choose a new set of replication points, but also adapt the number of replicas to match changes in an application's production and consumption traffic.

The main objective of this Thesis is to design an adaptive multi-replication DCS framework that efficiently creates a distributed information delivery service. The proposed framework is highly flexible, yet also presents challenges towards optimizing its adaptation to application's traffic.

In addition, since we are concerned about providing truly practical solutions, we have specified all the necessary operations in order to turn our design in a practical solution. Therefore, in order to demonstrate that, we have implemented the proposed framework for the case when consumption dominates production. In addition, this implementation validates in a real deployment most of the outcomes from the theoretical work.

Finally, all previous related works, as well as our initial framework proposal, assume that consumption queries and production events intensities are roughly homogeneously distributed within the network. However, that assumption is not true for many WSAN applications where consumption and/or production could be mainly localized in some region of the network. Therefore, in this Thesis we propose a heuristic that adapts our framework to those scenarios in which the production events and consumption queries traffic are heterogeneously distributed across the network.

To finalize this introductory chapter, we briefly summarize which are the main contributions presented in this Thesis:

- We propose two novel multi-replication DCS proposals: Quadratic Adaptive Replication (QAR) and Random Multi-Replication, that are compared to previous proposals

in the literature, and outperforms all of them in terms of traffic reduction. In particular, Random Multi-Replication is the one that presents better results.

- We propose and validate a simple model to determine the optimum number of randomly-placed replicas for Random Multi-Replication, as well as the optimal number of replication nodes placed in a grid structure for QAR, in terms of minimizing the overall traffic and associated energy consumption, given the measured intensities for production and consumption traffic of an application
- For Random Multi-Replication, and based on the same model, we assess the utilization of replication resources such as storage, which allows us to evaluate if memory requirements are sufficient when multiple applications share the same network resources, or if the amount of replication should be constrained due to the limited memory capacity of nodes.
- We propose a simple mechanism to equalize the energy burdens across the network and to adapt the degree of replication to an application's (possibly changing) traffic. We achieve this by changing the set of replication nodes over the time. A deep analysis of the implications of changing the replication nodes is also presented in this paper. Moreover, we demonstrate via simulation that changing the set of randomly located rendezvous nodes in large WSAN extends the lifetime at least by 60% when compared to previous proposals in the literature. This enhancement is shown to reach factors of 10x under some conditions.
- We propose various mechanisms to implement the above information delivery framework. In particular we propose the use of a *Meta-Information Service* in a WSAN supporting multiple applications. This service enables efficient bootstrapping of new sensor nodes and new applications, while addressing key fault-tolerance requisites for such networks.
- We have developed the full framework functionality in a 20 motes testbed, whose evaluation results show that the main analytical model and simulation results are also applicable to a small network. In addition, we demonstrate that our framework is feasible and could be easily implemented in commercial motes.
- Finally, we extend our baseline work (as well as previous ones in the literature), which assumes a spatially homogeneous traffic distribution, with an heuristic that adapts our framework and improve the network performance when the consumption and/or production traffic distributions are heterogeneous. To the best of our knowledge, this is the first work that adapts a DCS multi-replication network to spatially heterogeneous traffic conditions.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 briefly present the state of the art for Wireless Sensor Networks. Main works on Data-Centric Storage are introduced in Chapter 3, which in addition describes the most relevant related works for this Thesis. Next we

briefly describe in Chapter 4 the problem statement of this Thesis and the main assumptions realized in this work. Our novel multi-replication DCS proposals: Quadrative Adaptive Replication (QAR) and Random Replication, are introduced in Chapters 5 and 6 respectively. Chapter 7 describes the Dynamic Adaptive Random Multi-Replication Data-Centric Storage framework, whose implementation appears in Chapter 8. The adaptive solution of our framework to spatially non-homogeneous traffic distributions is introduced in Chapter 9. Conclusions and future works for this Thesis are described in Chapter 10. Finally, Chapter 11 closes the Thesis with those works published out of this Thesis.

Part II

State of the Art and Related Work

Chapter 2

State of the Art of Wireless Sensor Networks (WSN)

A wireless sensor is a small and low cost 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 via relay nodes and process it in order to get a common goal, a Wireless Sensor Network (WSN) is obtained.

WSN has been mentioned as a key technology in a very close future. A lot of research, standardization, and commercial efforts have been made in the last decade, and quite a few industrial companies have open product lines on WSN.

This chapter briefly summarizes main aspects related to WSN, mainly focusing on research and standardization efforts.

2.1 WSNs characteristics

There are several characteristics that define a WSN as a novel concept of communication network on its own. Next, the main characteristics are introduced.

- **Wireless**: The first characteristic of a WSN is the wireless nature of the communications taking place. This enables the possibility of a much simpler and low-cost deployment since sensor nodes can be placed almost anywhere without the need of installing cables to join them. This characteristic has been highlighted in many papers, which even propose that in the future the deployment of a WSN could be done by throwing hundreds or even thousands of sensors from a plane in order to monitor remote areas with difficult access for human beings.
- **Battery-Powered**: This is the main characteristic that differentiates WSNs from other

ACTIVE PROCESSING		
Mode	Typ	Unit
Cpu processing	4250+310/MHz	µA
Radio transmit	44	mA
Radio receive	49	mA
The following current figures should be added to those above if the feature is being used		
ADC	580	µA
DAC	240/290	µA
Comparator	72	µA
UART	270	µA
Timer	70	µA
2-wire serial interface	50	µA
SLEEP MODE		
Mode	Typ	Unit
Sleep mode	3.5	µA
The following current figures should be added to those above if the feature is being used		
RAM retention	25	µA
Comparator (low-power mode)	1.7	µA

Figure 2.1: *Jennic JN-5121 current consumption map (from Jennic JN-5121 data sheet document).*

wireless ad-hoc networks. Sensor nodes are assumed to be battery powered and, independently of what are the research goals (e.g. routing, security, data-acquisition, etc), most of the proposed WSN solutions are energy-aware. Then, one of the most important metrics to evaluate any WSN research proposal is the sensornet lifetime, which is directly related to the node's energy consumption. Therefore, generally speaking if a proposal reduces a lot the delay and at the same time uses very efficiently the available bandwidth but it is very energy demanding, then it is quite useless in the field of WSNs.

Due to this limitation, the energy demands of the main functionalities realized by a sensor node have been deeply studied. It has been demonstrated that the most energy demanding task is the use of the wireless transceiver either for transmission or reception. That task is some orders of magnitude (depending on each particular mote)

larger than processing tasks (timers, sensor reads, etc). Figure 2.1 shows the current consumption of different tasks for a JN-5121 mote manufactured by Jennic vendor (this mote is used in the testbed developed this thesis). Clearly, wireless transceiver utilization is consuming most of the energy. Therefore, many solutions in the WSN field try to avoid as much as possible the usage of the wireless transceiver. In order to save data transmissions there are a lot of research works on: in-network processing [PSW04], data aggregation [RV06, KEW02, IEGH02, WGA06, HCM05, BdVS04], data fusion [DDHV03, GMPK10, YKT03, LLLD06], data compression [KL05] etc.

Furthermore, most of the commercial sensor nodes present sleep modes that leave the device in a very low-energy state. From the networking point of view this state can be managed by synchronizing and/or coordinating the nodes in the network so that they are able to enter in sleep mode when they do not have to participate in the network operation. Also a lot of research has been carried out, especially at the MAC layer [YHE02, PHC04, vDL03, LKR04, oEE06], in order to permit a sleep period as longer as possible to reduce the sensor motes energy consumption

- **Low Bandwidth**: A sensor usually produces a low amount of data, thus the bandwidth requirements are also low. Therefore, the data rateof this kind of networks is usually low when compared with other wireless ad-hoc networks. For instance the IEEE 802.15.4 standard [oEE06] defines two different operation band frequencies: 900 MHz and 2.4 GHz. In the first one the maximum data rate is 40Kbps, whereas in the second case it goes up to 250Kbps.

- **Low-Cost**: Sensor nodes are thought to be extremely cheap in a near future, in the range of few dollars. Although it is still soon to find sensor nodes at that price, they are quickly reducing their cost. For instance, we used two different development kits (including 5 motes[1] per kit) bought from the Jennic company. We bought one kit (JN-5121) at the end of 2006 and the price was around 500e, thus 100e per node. At the beginning of 2009 we bought 3 additional kits (JN-5139) an the price was around 250e. Therefore, in 2 years the prices were reduced by a half. In addition, the nodes in last kits are better, since they include new hardware and software features that facilitate their use. Therefore, our experience says that the price of sensors nodes is rapidly decreasing and at the same time the feature set of the nodes is increasing. It must be highlighted that low-cost is a crucial requirement for WSN in order to fulfill the huge hopes that this technology generated during its inception about a decade ago.

2.2 Sensor hardware

In this section we briefly summarize the common elements present in most sensor boards.

- Processing Unit: There are different processor utilized by different manufacturers.

[1] In fact a common name for sensor nodes is *mote*, which shows the intended low cost and ubiquitous deployment of WSNs

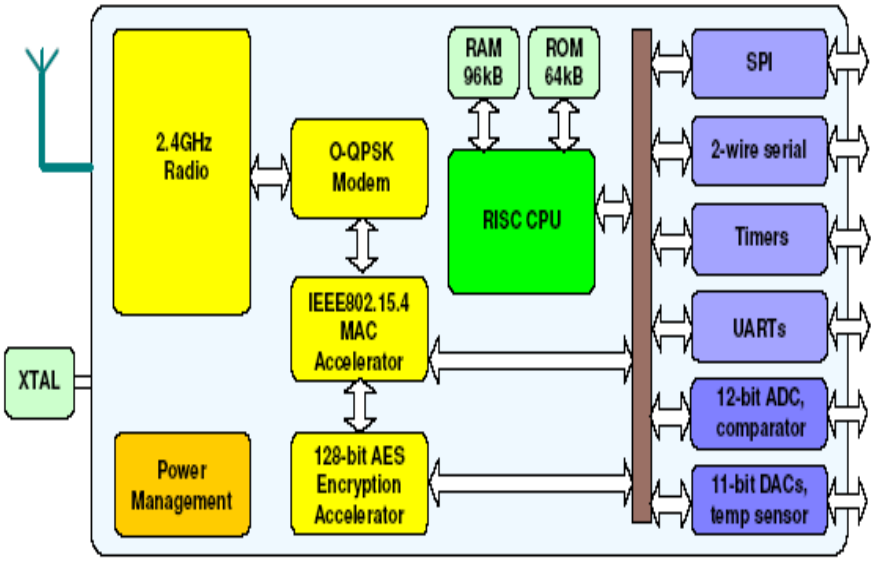


Figure 2.2: *Jennic JN-5121 mote block diagram (from Jennic JN-5121 data sheet document).*

Most of them can only be considered micro-controllers while other products include general purpose CPU.

- Communication Unit: It is usually a radio transceiver. However, there are some special sensor networks whose communications are based on ultrasound, such as in the underwater sensor networks (UWSNs), or infrared devices such as in the optical sensor networks

- Sensing Unit: It includes the actual sensors that measure physical parameters. Commercial sensor boards usually are integrated with different type of sensor devices like temperature, humidity or light-intensity sensors. In addition, these boards are provided with peripheral ports in which external sensors can be also connected.

- Connectivity Unit: This element is used to connect the nodes to computers (e.g. send WSN data to the external word, debug sensor node behaviour, etc). Most of the commercial boards use USB or Serial ports for this purpose.

- Storage unit: This element is used to provide with storage capacity for all the operations realized by the mote, i.e. data storage, routing tables, MAC information, etc. A combination of ROM and RAM memories is usually provided.

The system units presented above are the fundamental components that can be found in most of the commercial motes (used for development) in the market. In addition, some other elements appear in many sensor boards: ports to connect external sensors, buttons, LEDs, etc. For instance, Figure 2.2 shows the blueprints of a JN-5121 mote from Jennic.

2.3 MAC and Routing layers

There have been a lot of research in WSN at all protocol stack levels, from the physical to the application layer. However, due to the focus of this Thesis we just put emphasis and briefly classify MAC and routing proposals.

2.4 MAC Layer

There are three big families of MAC protocols in WSNs: TDMA-based, CSMA-based and hybrid.

Time Dimension Multiple Access (TDMA) protocols schedule the nodes' access to the medium in order to avoid contention and collisions. Therefore, TDMA-based protocols assure collision avoidance. Hence, when a node transmits it is sure that the medium is free for its transmission. However, in many cases this is not the best option. For instance in network with real-time data a TDMA-based MAC protocol a node could not access the medium a soon it has available data to be transmitted, but it has to wait until the slot time it has been assigned for transmission, even when nobody else is transmitting.

Carrier Sense Multiple Access (CSMA) protocols can be classified in centralized (IEEE 802.15.4 Beacon enabled [oEE06]) and distributed (AI-LMAC [CvH04], Crankshaft [HL07], Y-MAC [KSC08]) protocols. The former rely on a central node (i.e. IEEE 802.15.4 coordinator) that schedules the remaining nodes for transmission, whereas in the distributed protocols the nodes organize themselves by exchanging information to create a cordinated transmission scheduling.

CSMA-based protocols do not pre-define any transmission scheduling. Nodes could access the network at any moment, thus there exists some probability of collision. Some CSMA-based protocols are: S-MAC [YHE02], B-MAC [PHC04], T-MAC [vDL03], X-MAC [BYAH06], IEEE 802.15.4 Non-Becon Enabled [oEE06]. For instance, the last one implements a backoff mechanism similar to the one used by the IEEE 802.11 standard. Therefore, although there is still a certain collision probability, in low-loaded networks a node could use most of the bandwidth without necessity of scheduling its transmission.

Finally, Hybrid access protocols, such as Z-MAC [RWAM05], propose a mix of previous techniques. Basically, they propose to schedule the access in saturated regions (e.g. close to the sink), whereas using CSMA-based protocols in low-traffic areas.

2.5 Routing Layer

There have been many WSN routing proposals in the last decade. Some of them are direct adaptation of routing protocols for wireless ad-hoc networks (e.g. AODV [PBRD03]), whereas there are many others that have been specifically designed for WSNs. Many taxonomies have been proposed for classifying routing protocols, we have chosen the one pro-

posed by Govan and Kurose at Infocom 2009 tutorial session. They define the following categories of routing approaches and the main proposals for each type:

- Data-Centric: This routing approach proposes to create routes based on the nodes interest. For instance, if there are some nodes that are interested on temperature readings or they generate them, they spread this interest across the network. Therefore, routes between nodes that have similar interest are established. The most relevant work in this field is Direct Diffusion [IGE00, IGE+03].

- Coordinate-Space: In some cases, nodes have some knowledge of its position within the network (e.g. GPS coordinates, virtual coordinates, etc). This information can be used to route messages towards some given coordinates in the space by sending the message towards that location. For this kind of routing we can find an entire research line called, Geographic Routing, with plenty of works in the literature. The most relevant proposal is GPSR [KK00].

  Furthermore, there are some other proposals that do not forward the message to a particular coordinates, but towards a certain region or beacon node. Some works on this area are: BVR [FRZ+05], HVGR [ZCR08], Rendezvous regions [SH04], etc.

- Ad-hoc: These type of proposals use some protocol of existing ad-hoc routing protocols. In particular, there are some adaptations of the AODV [PBRD03] protocol such as TinyAODV or the AODV version proposed in the Zigbee specification [All05]. There are some other proposals in this category such us TYMO [scr07].

- Collection-Tree: These proposals uses a custom routing for centralized WSNs in which all the data is transmitted to a sink. Therefore, they propose to create a routing tree (i.e. parent-children structure) and route the queries downstream in the tree, whereas the data flows upstream. Two of the most important proposals in this category are MultiHopLQI [Mul09] and CTP [GFJ+09].

2.6 Storage in WSNs

We have introduced storage as a separate section due to its relevance in this PhD Thesis. There are three main ways of storing the information within a WSN:

- Sink storage or External storage: The first and most used storage mechanism is to save all the information in the sink node. Everytime a measurement is obtained or an event is detected, they are forwarded to the sink. This technique is the most adequate for the common vision of a WSN with centralized management. However, there are scenarios in which it is not the best option. For instance, if only some part of the information generated by the network is really worthy, lot of resources (e.g. energy) are wasted for forwarding useless information to the sink. In addition, in novel networks such as Wireless Sensor and Actor Networks (WSANs), it is not true anymore that the unique querying node is the sink, but there could be additional information consumers inside

the network, e.g. actors, that need that information to perform some actions. Then that information does not need to be always sent towards the sink, but it could be retrieved instead by other nodes within the network. Therefore, this new network paradigm has pushed the investigation of different methods to store the information.

- Local storage: The second option in order to avoid the transmission of useless data is that each sensor node stores locally the measured data or detected event. Later other nodes (e.g sink, actors, etc) interested on some information are able to query the suitable nodes in order to retrieve the desired information. The easiest way of being sure of retrieving all the information is using flooding in the network, however flooding is a very energy-consuming operation. Otherwise, more sophisticated service discovery algorithms are required in order to allow information consumers knowing which nodes have to be queried.

- In-network storage: This is a compromise solution between the previous two mechanisms. The information is stored inside the network like in the local storage case, but not in the nodes that originated it. Generally, some algorithm is used to define one or more rendezvous nodes so that the nodes producing the information push the data towards those nodes, and nodes consuming that information (e.g. sink, actors, other sensors, etc) query those rendezvous nodes. This storage mechanism is the one consuming the less energy under certain network conditions [RKY+02], since useless information does not need to travel towards the sink (that could be potentially far away), and at the same time if the rendezvous nodes are well defined, flooding queries across the network is completely avoided.

2.7 Standarization Efforts

Industry is looking to WSNs as a very profitable market in the close future. Therefore, the incipient market behind WSN technology has influenced some standardization bodies like the Institute of Electrical and Electronic Engineers (IEEE) or the Internet Engineering Task Force (IETF) to put some effort on standardizing suitable protocols for WSNs, and even sparked the creation of an association of companies, called Zigbee Alliance, to define a common specification for Wireless Sensor Nodes and Networks. Next, we briefly describe these three standardization efforts:

- IEEE 802.15.4 standard [oEE06]: It was initially defined for Wireless Personal Area Networks (WPANs), that are nothing but a particular type of WSN. However, it has been adopted by manufacturers to also implement WSNs solutions. Therefore, many commercial motes implement this standard. It has been defined for the physical and MAC layers. Its main characteristic is that it offers two communications modes. The first one is beacon-enabled, in which a coordinator node sends beacon frames that defines a superframe structure, which among other operations organizes the transmissions. The second operation mode is non-beacon-enabled, in which there is not such superframe structure and the upper levels are the ones in charge of scheduling the

transmissions. There are lot of features included in this standard including: type of devices, frame format, security, CSMA algorithm to access the medium, etc.

- IETF 6LowPan [MKC07]: 6LowPan stands for IPv6 over Low power Personal Area Network, but nowadays, as in the previous case, it no longer refers to WPANs but WSNs. The base specification document is RFC 4944 [MKC07]. This standard defines encapsulation and header compression mechanism to allow IPv6 packets being sent and received over an IEEE 802.15.4 network. This working group faces lot of challenges in this standard: adaptation of IPv6 to IEEE 802.15.4 MTU (from 1280 bytes to 127 bytes), address resolution (128 bits for IPv6 and 16 bits for IEEE 802.15.4), dealing with transport protocols such as TCP, etc. We believe that 6LowPan is a key step in order to enable the so-called Internet of Things, in which sensors, cameras, or other ubiquitous computing devices are willing to be integrated within the Internet. However, for particular WSN applications, it could introduce an important overhead that is not always necessary nor desirable.

- Zigbee Specification [All05]: This specification has been proposed by the Zigbee Alliance. This forum is composed by more than 100 companies including major players in the electronic components market like: Philips, Texas Instrument, Sony, Siemens, etc. Zigbee specification defines a network and a application layer that run over the IEEE 802.15.4 standard. They propose different multi-hop network structures (star, mesh and tree-based) and the routing to be used in each one of them. They also propose an application profile architecture in order to facilitate the application definition in Zigbee networks. The alliance is in charge of evaluating if a given device is compliant with the Zigbee protocol stack. Those products that are Zigbee compliant are attached with a Zigbee stamp indicating this fact.

2.8 Applications Requirements

There is a wide set of WSN applications: monitoring the environment, monitoring building and other structures, object tracking, industrial monitoring, healthcare, monitoring non accessible areas such us a volcano, surveillance, battlefield etc.

Different applications have different requirements, but the most important ones are:

- Latency: There will be applications that require low-latency in the data delivery, whereas some others would not face problems if the data adquisition is delayed a bit. For instance battlefield and surveillance applications are usually very sensitive to data latency, whereas monitoring a plantation conditions is not affected from a communication high latency.

- Reliable delivery: It is always a desirable feature for any WSN application. However, there are applications that are more sensitive to a data loss. For instance, for networks that do not allow an easy physical access in case of failure (e.g. volcano, jungle, forest, dessert, etc) reliable data delivery is a must, whereas if the application is monitoring

the temperature in a building it may be acceptable to loose one out of 5 temperature readings.

- Lifetime: There are some WSNs whose operation time should be very large because the high-cost of replacing node's battery due to its physical location. Especially, environmental monitoring applications focuses on using protocols that extend as much as possible the sensornet lifetime.
- Data Rate: Usually WSN are designed for low-rate applications. However, there could some applications that during particular moments demand a high bandwidth due to a heavy load. Therefore, data rate is an important requirement when deciding what kind of solution is going to be applied in a particular application.

Usually when designing a WSN for a particular application a trade-off between the previous requirements is found. Among all research works in the literature there are solutions that focuses more in one of the requirements and try to optimize the network operation considering just that requirement.

2.9 Novel network concepts derived from WSNs

The great success of the WSN field in the research arena has lead to the creation of new WSN subfields that are hot-topics nowadays. Next, we describe some of these research lines.

- Wirelesss Sensor and Actor Networks (WSANs) [AK04]: This new research area introduces a new type of node in the network, namely an actuator or actor. An actor is a node that, based on the information measured by the sensors, performs some action. An actor could be a node of different nature, from a simple LED that is activated by a node based on some local data, up to a very complex robot that performs high-level actions based on the information obtained from all the network. This is a very active field that presents particular challenges different to those presented in standard WSNs. In some cases, WSANs change the traditional understanding that all the network information flows towards a single sink, because now there are actor nodes that are also information consumers different than the sink. This affects principally to data storage, which traditionally was performed by the sink. Therefore, external storage could not be the best option in a WSAN, but use some kind of in-network storage.
- Multimedia Wireless Sensor Networks (MWSNs) [AMC07]: Standard WSN usually are low-rate. MWSNs open a new research line in which the exchanged data is composed by images, audio or video. These networks consider devices such as cameras, microphones, etc as sensors, thus the network load is much higher than in standard WSNs. Then, in order to reduce the data traffic as much as possible, MWSNs focus on studying new codecs suitable for WSNs and new in-network processing mechanisms that allows reducing the network transmissions.
- Underwater Sensor Networks (USNs) [APM05, APM06, PKL07]: It is an special type of WSNs since the transmission media is not air anymore, but water. Thus, in most

cases the RF signals used in standard WSNs are changed by ultrasound communications. Then, due to the use of a very different physical technology, USNs are defining new channel and communication models.

- Unattended Wireless Sensor Networks (UWSNs) [EGHK99, ZWR+09, YCL+10]: This kind of networks is deployed on areas where human access is very difficult. Thus, it is very costly to physically access the network to configure nodes or replace batteries. Therefore, one of the main issues that need to be solved is these networks is self-organization. A second particular issue is the network resilience and fault tolerance. Moreover, a new network paradigm is extending a bit the concept of UWSNs, and envisions networks that will operate autonomously performing very complex operations without human intervention. That kind of sensor and actor nodes will need to self-organize, and cooperate together in order to deal with heavy and complex processing tasks. This work is based on the peer-to-peer (p2p) network paradigm so that nodes will have to exchange information among them [CCUnL07], define an efficient in-network storage system, perform in-network data processing, utilize protocols that provide a long-term network operation lifetime, etc

2.10 Distributed Hash Table for WSN

To finalize this state of the art section, we include a brief section about Distributed Hash Tables (DHTs) for WSN since DCS utilizes the principles of DHTs.

There have been lot of research in the field of p2p networks utilizing a DHT for indexing the content in order to facilitate content search [SMK+01, MM02, RD01]. In addition, we have some experience even though in peer to peer networks [CCAC+09, CCCA+10, CUnB09] that has helped us to easily understand and utilize the DHT concept over a WSN. In particular, a DCS system can be seen as a DHT that store the information related to a given data type.

Chapter 3

Related Work

This chapter describes those works that are close related to the contributions of this PhD Thesis.

Data Centric Storage (DCS) is an in-network repository mechanism that defines a storage rendezvous node based on the name assigned to an application or event type. For that, DCS relies on a hash operation over the application's name that provides the locationof the rendezvous node for that application within the network.

Therefore, in this chapter we review the most relevant works about Data-Centric Storage (DCS), starting from the seminal work in [SRK+03], analysing what are the suitable scenarios where DCS can be applied, going through those works proposing multi-replication for DCS system, describing other works in the literature related to storage optimization, and analysing some other relevant contributions to this field. Finally, we discuss some open issues and research challenges to be covered in DCS.

3.1 Terminology

This section defines some common terminology that is used in the rest of the chapter, since different works use different names to refer to the same concept.

- **Event type**: It refers to a physical phenomenon or to some combination of them, providing semantical meaning to the data it refers to. Therefore, an even type could refer to a single physical phenomenon (e.g., temperature, humidity, *etc.*), or to more complex combinations (e.g., an event type 'FIRE' is generated when a sensor measures a temperature higher than 100 Celsius degrees, the smoke concentration is over 12 parts per million (ppm) and the humidity has been reduced by a 5% in the last 10 measurements).

- **Rendezvous node or home node**: It is the node responsible of storing the data related

to a given event type.

- **Producer node**: When a node detects an event, it forwards the information related to that event towards the suitable home node. Then, we name that node as a producer. An application usually has several producer nodes
- **Consumer node**: It is a node that queries a rendezvous node to retrieve the data related to a given event type or application. An application has one (the sink) or more consumer nodes.
- **Replica or replication node**: When several nodes play the role of home node for the same data type, we refer to them as replicas or replication nodes, because the data is usually copied (replicated) in all these nodes.
- **Relay node**: It is a node in the network that does not have any of the previous roles (replica, consumer or producer) so that it is only used to relay messages.

3.2 Geographic Hash Table (GHT), the first DCS Proposal

Ratnasamy *et al.*, first defined the concept of Data Centric Storage (DCS) in [SRK+03]. They combined the idea of a Distributed Hash Table (DHT) [SMK+01, MM02, RD01] together with the Greedy Perimeter Stateless Routing (GPSR) [KK00], a geographic routing protocol, to create a DCS system called Geographic Hash Table (GHT).

In order to employ geographic routing, this work 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 by all nodes. In fact, most DCS works assume that the sensornet is enclosed in a square.

In GHT when a producer sensor detects an event, it uses a hash function over the application or event name (e.g., *hash('TEMP')*). The hash function provides as the output some spatial coordinates inside 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 closest node to that spatial location becomes the home node for that event type and receives the producer message, because GPSR itself is enough to find the closest node to a 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, and thus it obtains exactly the same spatial location. Next, it uses 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. Figure 3.1 shows a simple example to let the reader better understand how GHT works.

GPSR uses two different algorithms for routing: the first one is called *greedy forwarding*, which in each hop moves the message as closest to the destination as possible, *i.e.*, 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. This could be

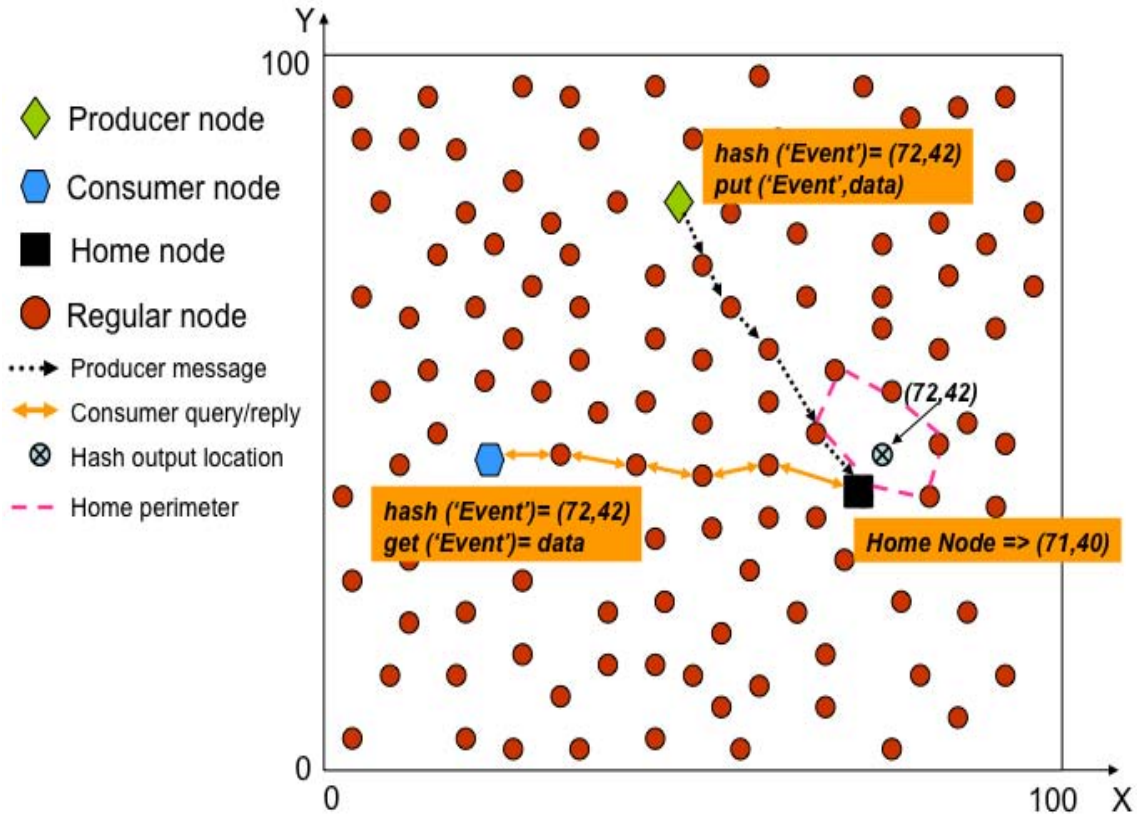


Figure 3.1: *Geographic Hash Table (GHT) example.*

either because the node is already the closest one to the destination, or because it has found a routing hole and some detour is needed. In such cases GPSR uses the second routing algorithm, called *perimeter forwarding*. This algorithm uses the right-hand rule [KK00] to surround the hole, always following the same direction. If during the perimeter forwarding phase a node closer to the destination that the one starting the perimeter forwarding is found, then the routing is switched again to greedy forwarding. However, if by using perimeter routing the message reaches the same node who started the perimeter routing, that node understands that it is the closest one to the destination coordinates, and therefore the responsible of storing the information related to that event. Then, this node becomes home node, and all the nodes in the perimeter that encloses the destination coordinates are called *home perimeter* nodes.

The authors warned that using a single home node could create a hot-spot problem. Therefore, they proposed to use an additional protocol, called Perimeter Refresh Protocol (PRP), in order to replicate the data in all the nodes of the home perimeter. Thus, reducing the hot-spot of having a single rendezvous node serving all producers and consumers of a given event type. For that purpose, the home node sets up a timer and periodically sends refresh packets along the home perimeter. These refresh packets are just addressed with the GHT location provided by the hash function over the event type stored by the home

node. In addition, refresh packets contain the current data stored for that event type, and they are routed as any other packet in GPSR. Then, when a refresh packet reaches a node, there are three options: (i) If that node is closer to the destination coordinates than the home node, then it becomes the new home node. It consumes the refresh packet storing the data it contains and generates a new one. (ii) The node receiving the refresh packet is not closer to the destination coordinates than the home node, then, it just stores the event data and keeps forwarding the packet. (iii) If the same refresh packet reaches twice a node that was not the home node, then it becomes the new home node. Thus it consumes the packet and establishes its own timer to start generating refresh packets. Therefore, PRP provides fault-tolerance and is able to deal with mobile scenarios. However, it must be noted that PRP only provides local replication because all the replicas are placed nearby, in the same routing area.

In addition, [SRK+03] compares three different storage mechanisms: local storage (each producer stores the measured data), DCS and external (sink) storage. A simple mathematical model, as well as a complete performance evaluation, are presented indicating when DCS outperforms the other two storage models. GHT focuses on a standard WSN, therefore it assumes that consumption queries are always generated from the sink.

3.2.1 GHT Discussion

One of the main drawbacks of GHT is the complexity added by GPSR when it applies the perimeter routing algorithm, since it requires (among other things) to planarize the connection graph of the sensornet. Some other geographic routing proposals (e.g., [BMSU99, BMSU01]) can be found in the literature and could be also applied to DCS. In addition, there are some proposals that introduce DCS-specific routing protocols (Section 3.4 covers these proposals).

Moreover, using a single home node could create a hot-spot problem because all traffic targets a single node (the home node), which expends its battery quicker than other nodes. GHT authors propose to use several local replicas, by means of PRP, in order to reduce the potentially high load suffered by the single home node. Then, if several replicas are answering consumer queries the load is balanced among them. However, local replication does not completely solve the routing hot-spot problem. For a very popular content many queries will reach the home perimeter, thus the area surrounding the home perimeter becomes a routing hot spot. Therefore, local replication could help balancing the home node's extra-energy expenditure, but it does not fully solve the routing hot-spot problem. As it will be shown later in this chapter, it is necessary to replicate the data in different places along the sensornet in order to reduce that effect (see Section 3.6).

3.3 Scenarios for DCS

There are two kinds of sensor networks where DCS could be applied. On one hand, standard WSNs have a central node, usually named sink, base station or gateway, that connects the WSN with the external world. In this kind of networks, sensor nodes gather information

from the environment and can store it either locally, in a rendezvous node (if DCS is being applied) or push it to the sink that stores all data. In such a network, usually the consumers of the WSN data are external users that query the network through the sink.

On the other hand, nowadays not only sensors but actuators are becoming more and more relevant in the WSN research community, and Wireless Sensor Networks are becoming Wireless Sensor and Actuator Networks (WSANs) [AK04, RV08]. An actuator is a node that, based on the information retrieved from one or more sensors, performs some action. For instance a node controlling a water valve in a plantation opens it depending on the humidity and temperature conditions obtained from the sensors in the field. It must be noted that a node could perform both roles in the network, being a sensor and an actuator at the same time.

The usage of WSANs enables the possibility of autonomous WSANs that, in a fully distributed way, measure the environment, process the collected data and perform some actions to achieve a goal without any external intervention. Therefore, in this case, DCS seems to be a suitable storage and information retrieval system because in these scenarios there is no a clear central, powerful node as in the sink case.

3.4 Routing for Data Centric Storage

There are some routing protocols [CCNR06, NS03] proposed for WSNs that could be applied to DCS, although they were not specifically designed for this paradigm. Also, there are other proposals in the literature that define novel routing protocols specifically designed for DCS such as: Rendezvous Regions [SH04], pathDCS [ERS06] and HVGR [ZCR08].

These authors mainly argue that unicast routing using GPSR has several problems for DCS, and they focus on other type of approaches. Rendezvous Regions and HVGR are focused on region-oriented routing that does not route messages to a particular spatial location in the network, but just to a particular area. In that area there is local knowledge to decide which node becomes the home node. Besides, pathDCS does not map an event type into a spatial location, but to a routing path that ends at the home node.

3.4.1 Rendezvous Regions

This work [SH04] uses geographic regions as rendezvous areas for producers and consumers. The authors claim that sending messages towards geographic regions instead of geographic points, relaxes the requirements for location accuracy. In this case an event type is mapped into a region identifier.

The network is divided into rectangular regions where each region could be responsible of one or more event types. The side of a region is several hops long, although the exact value is not specified. Each region is assigned with an identifier. Then, each event type is mapped to a region. Finally, each region locally chooses the home node as well as the local replicas.

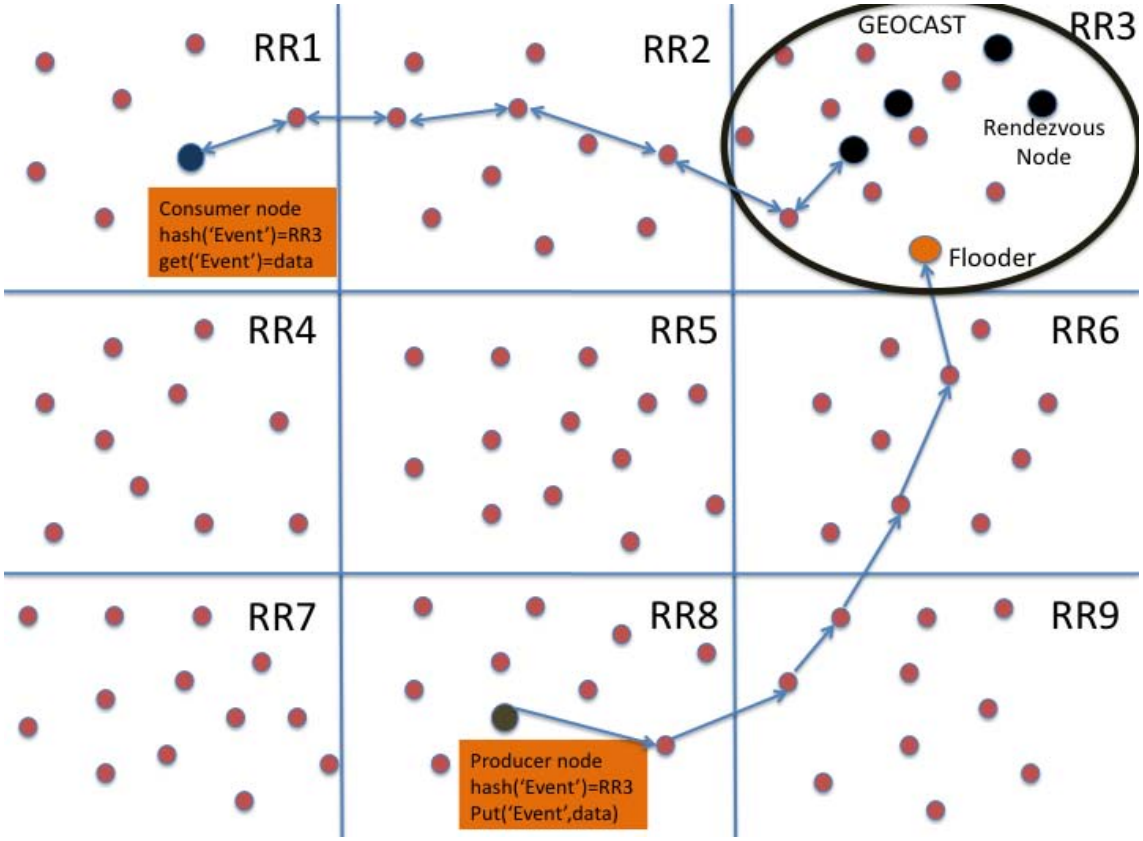


Figure 3.2: *Example of Rendezvous Regions routing protocol.*

The authors define 3 different forwarding mechanisms:

- *Unicast*: Direct messages to a given node. This could be used when the location of the rendezvous node is well-known.
- *Geocast*: Sending the message to all nodes in a geographical region. It is useful to send a message to several nodes inside a region (in this case the local replication nodes within that region), however it incurs in an important overhead. The authors propose to use it just for production events.
- *Anycast*: It is enough with reaching one of a set of nodes (one of the local replication nodes). The authors propose to use this forwarding mechanisms for consumption queries.

It is assumed that nodes are able to detect in which region they are located. In addition, they know how the regions are distributed within the sensornet, that is, each region location and its identifier.

Local replicas are elected on-demand. When a production event message reaches the first node in the targeted rendezvous region, this node (known as flooder) geocasts the message inside the region. Each local replica receiving the message sends back an ACK to

the flooder. If the number of ACKs is not enough (*i.e.*, the number of acknowledged local replicas is lower than the minimum threshold established) the flooder geocasts again the message including a self-election probability, p. Then, each node that receives the geocast message elects itself as replication node with p probability. This process keeps going until the minimum number of replicas is reached or p becomes 1.

When a producer generates an event, it computes the target region and includes its ID within the message. The nodes in the path forward the message towards that region. Once the message reaches the flooder, it geocasts it and receives one (or more) ACKs, that are notified to the producer. If after a given time the producer did not receive any ACK it sends the message again.

This mechanism works similarly for the consumer queries. However, when the query reaches the flooder it uses anycast since it has previously cached all the replication nodes' locations. The selected replica replies to the flooder, and then the flooder to the consumer. Figure 3.2 shows a graphical example of the Rendezvous Regions functionality.

The authors claim that this work performs better in mobile scenarios than the original GHT. Moreover, the main advantage of this protocol compared to GPSR is that nodes only need to know in which region they are located but not its exact position, thus avoiding the necessity of using GPS or other location device, or any other costly protocol in order to provide each node with its coordinates within the sensor field. On the other hand, the paper neither discuss the size of the area nor how the identifiers should be assigned to each area or how a particular sensor knows which its area identifier is. Sadly, all these assumptions are key issues in order to practically deploy this routing proposal.

3.4.2 Hierarchical Voronoi Graph Routing (HVGR)

The authors of HVGR [ZCR08] follow a similar approach than Rendezvous Regions since they propose to use a region-oriented routing, although, they implement it in a different way. The main contribution of this paper is that it employs logical coordinates, thus it avoids the necessity of each node to use GPS or other location device to know its own location.

The main idea underlaying HVGR is the use of hierarchical coordinates associated with landmarks (nodes whose location is well-known). There is a number of first level landmarks that divide the network into the same number of first level subregions. These landmarks are named L_i, with i=1,2,3,...,m_1. Later, inside each first level subregion there are several second level landmarks, each one with a different second level identifier. Therefore, inside the subregion belonging to the first level landmark, we find several 2nd level landmarks named as $L_{1,j}$, with j=1,2,3,...,m_2. This division continues until some *n-th* subregion in which all the nodes are within 1-hop distance. Finally, a node is identified as the sequence of landmarks it belongs to, e.g., L_1,$L_{1,1}$,$L_{1,1,1}$, etc. The number of landmarks in the *k-th* level is defined as m_k.

Every node in the sensornet learn where all 1st level landmarks are. In addition, a node located in a particular 1st level subregion knows the location of all the 2nd level landmarks within that 1st level subregion. In turn, any node in a given 2nd level subregion knows the

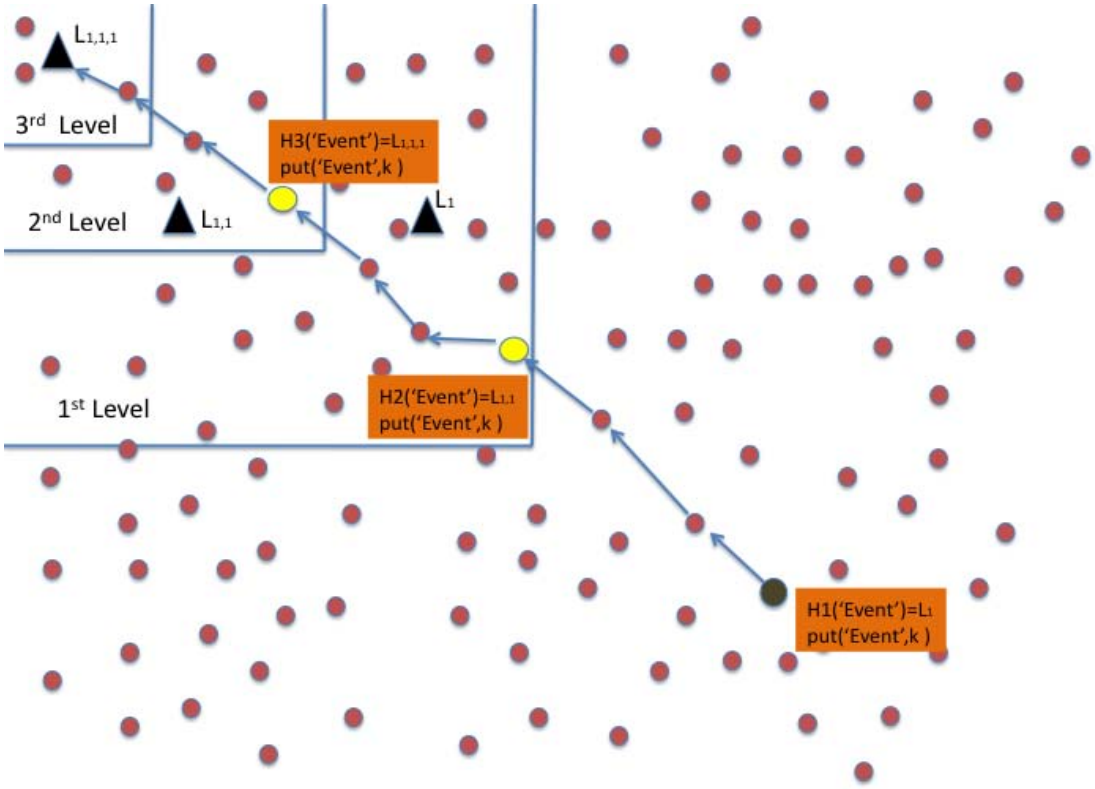


Figure 3.3: *Example of HVGR routing protocol.*

path to all 3rd level landmarks within that 2nd level subregion, and so on. This knowledge is required to implement the proposed recursive region-oriented routing.

There is a set of hash functions ($H_1, H_2, H_3, ..., H_n$) employed for routing, each one assigned to a different level. Then, when a producer generates an event, it uses $H_1()$ with the event type and obtains the 1st level landmark to send the message to. However, the message does not need to reach that landmark, but just the first node within that 1st level subregion. By receiving the message it is able to route the message towards the suitable 2nd level landmark. In order to do this, that node uses $H_2()$ over the event type and obtains the 2nd level landmark. Following this routing mechanism the message finally reaches the rendezvous node in charge of that event type. Figure 3.3 shows an example of HVGR routing.

HVGR requires a first phase where the hierarchical infrastructure is built, which means creating the regions and selecting the landmarks. There is a first node, called $\mathcal{C}$, which starts the landmark selection process, and it is defined as the master among all the first-level landmarks. HVGR uses [ZCR08] to select all m_1 1st level landmarks. Each of these 1st level landmarks is the master landmark within its 2nd level subregion, and they repeat the operation to find the rest of $m_2 - 1$ landmarks in each 2nd level subregion. The process stops when a master landmark detects that all the nodes within its region are its direct neighbours. It must be noted that, by using this algorithm, a 1st level landmark is also the landmark for

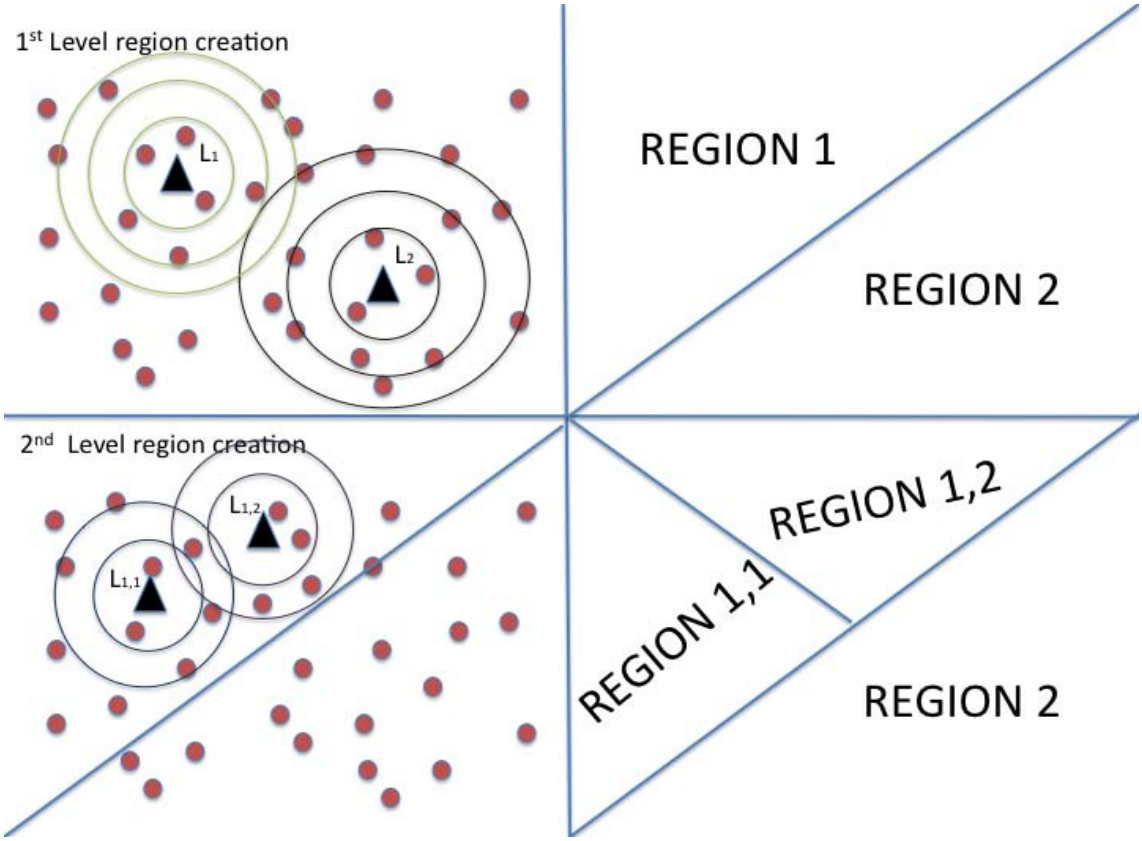


Figure 3.4: *Generation of HVGR routing protocol regions.*

the rest of the levels. Figure 3.4 shows a simple example of the region generation.

Once all the landmarks have been selected, there is an initial process in which every node in the sensornet learns the landmark coordinates. All the 1st level landmark nodes flood a LANDMARK message, so that each node in the network learns the path to all the 1st level landmarks, and then, each node selects its closest landmark. In such way, the network is divided into m_1 first level subregions. Next, the 2nd level landmarks make a scooped flood within its 1st level region. Again, a node knows all the 2nd level landmarks within its 2nd level subregion, and the 2nd level subregions are generated when each node selects its closest 2nd level landmark. By running this recursive algorithm the network is ready to employ the region oriented routing.

This solution relies on a hierarchical infrastructure that avoids the need of each node to know its own spatial location. Instead of this, each node knows the path to a set of landmarks and route the messages towards them. This routing is demonstrated to be more efficient that GPSR. The disadvantage is the cost related to the creation of the hierarchical infrastructure. Choosing the landmarks as well as notifying their position to the nodes requires quite a lot of messages, in particular for large WSNs.

3.4.3 PathDCS

PathDCS [ERS06] also relies on landmarks, here called *beacons*, to route messages and queries, thus avoiding the necessity of point-to-point routing (such as geographic routing) that requires precise location knowledge.

PathDCS authors also agree that point-to-point routing presents problems in WSNs, so that it is better to define tree-structures in order to route messages. In particular, pathDCS creates one tree per beacon node, rooted at the beacon node itself. They use existing algorithms for tree creation [DCABM05, WTC03] and use [FJL+97] to elect the beacon nodes. Also each node is assigned with a logical identifier.

The authors remark that in DCS, based on a key (event type), *k*, the home node should be consistent for any production event or consumption query. Then, the novelty of this proposal is that an event type is not mapped to a spatial location, but to a path. Roughly speaking, based on the event type there are a set of routing indications: "go to beacon 1, later travel two hops towards beacon 2 and finally one hop towards beacon 3". Therefore a path consists in a sequence of *p* beacons b_i, and lengths l_i, with i=1,2,...,*p*. Then, any event (production) or query (consumption) is first sent to beacon b_1 (using the tree created by b_1), next it is forwarded l_2 hops towards beacon b_2, and so on, until it is sent from the last node l_p hops towards b_p, where l_i is the distance in hops between the source node and the beacon b_i. Figure 3.5 shows a simple example of two production events in pathDCS.

A beacon node b_i is the node whose identifier is the closest one to *hash(k,i)*, whereas the length of the *i-th* segment is obtained as: $l_i = h(k, i)\ mod\ hops(n_i, b_i)$, where n_i is the node that starts the *i-th* segment and $hops(n, b)$ the distance, in number of hops, between the node *n* and the beacon *b*. Therefore, in order to apply this proposal, only two parameters are required: the number of beacon nodes within the network and the number of segments used in a path, *p*.

Moreover, pathDCS authors also propose to use local replication in the k-hops home node's neighbours to reduce the load of the home node, which is just a local replication mechanism.

PathDCS is a very interesting approach, which has been implemented and tested in real sensor nodes. As HVGR, it relies on a logical location infrastructure that provides references to all the nodes within the sensornet to perform the routing without knowing its own precise physical spatial location, and this seems simpler than the HVGR one. The main disadvantage is that pathDCS needs to create one routing tree per beacon node, which implies additional energy consumption. The energy for creating the beacon infrastructure depends on the size of the WSN and the number of beacon nodes used, which is left unspecified. In addition, as shown in Figure 3.5 the length of the paths is unbounded, and long paths could be used even in the case in which the rendezvous node is just one hop away from the producer or consumer node.

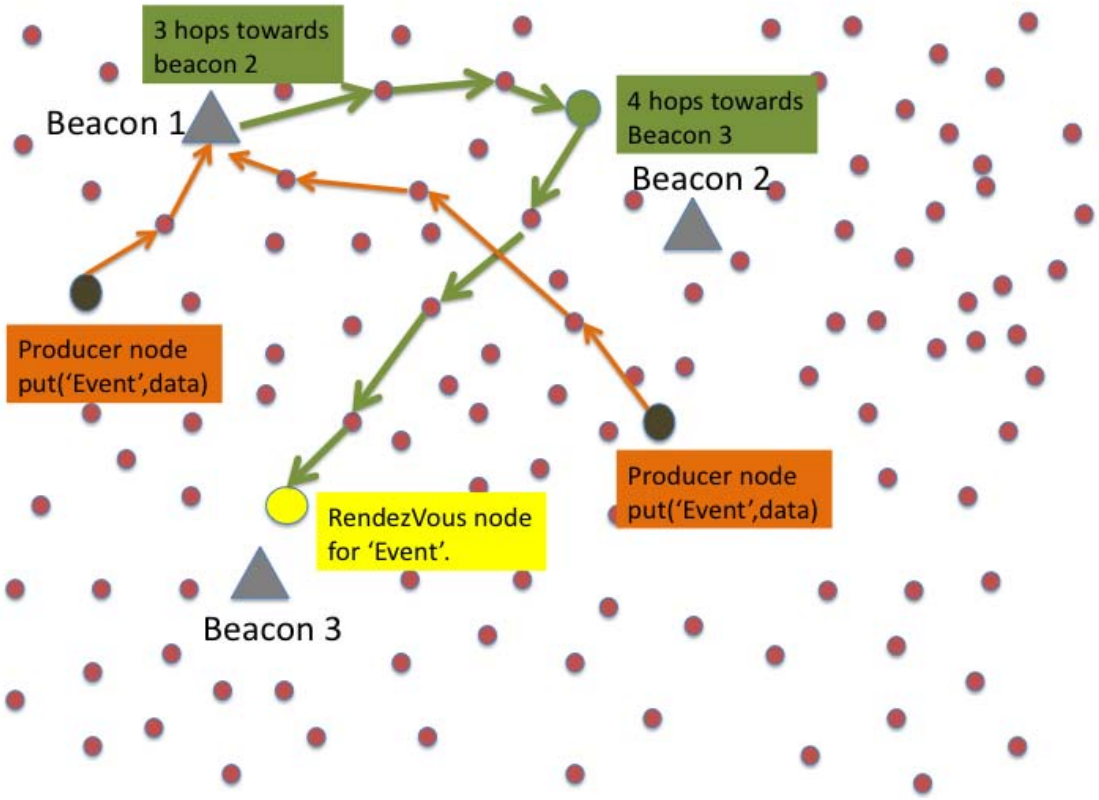


Figure 3.5: *Example of pathDCS routing protocol.*

3.5 Balancing Storage in DCS Systems

The home node (in DCS) or the replication nodes (in multi-replication DCS) are the nodes that store the data events on behalf of producers. In many applications data need to be accessible during certain time, thus home nodes could suffer a high memory utilization because they need to store many data records. Therefore, at some point the memory of a home node could be saturated, and some mechanisms are required either to avoid this situation or to select a new home node when the precious one is overloaded. In addition, in heterogeneous WSNs where some nodes have more memory than others, it would be interesting to select those ones as home nodes.

This section presents two relevant works [TWXD06, LSW10] that are focused on balancing storage in DCS systems.

3.5.1 Dynamic DCS

The authors of [TWXD06] propose to use a dynamic GHT that is based on two main features: (i) A temporal-based GHT, that changes the home node over the time. (ii) A

	t = 0	t = 1	t = 5	t = 10	t = 15	t = 20	t = 30	t = 40	t = 50
e_0	0	0	0	99	99	98	97	96	95
e_1	1	1	1	0	0	99	98	97	96
e_2	2	2	2	1	1	0	99	98	97
e_3	3	3	3	2	2	1	0	99	98
e_4	4	4	4	3	3	2	1	0	99
e_5	5	5	5	4	4	3	2	1	0
e_6	6	6	6	5	5	4	3	2	1
e_7	7	7	7	6	6	5	4	3	2
e_8	8	8	8	7	7	6	5	4	3
e_9	9	9	9	8	8	7	6	5	4
e_{10}	10	10	10	9	9	8	7	6	5

Table 3.1: *Cell selected by Dynamic DCS in function of the event type and time slot.*

location selection scheme that chooses as potential home nodes those with more memory and battery resources. They affirm that with these two improvements the sensornet lifetime will be significantly extended.

The authors focus the problem in node storage capacity, indicating that the home node of an event type could run out of memory, so that it could not be able to store new events. For that purpose, they divide the network into grid cells. First of all, they assume that there is one home node per event type that is allocated in the center of a cell. Then, they propose a hash function that allows allocating the home nodes of different event types in different cells. In addition, this hash function has into account a time slot Δt, to allow the home node of a given event type to change over the time. Both the event types and the cells are assigned with a numerical identifier. Then, the hash function uses as input the time slot number (t) and the event type identifier (e_i), and gives as output the cell number identifier where the home node will be placed for that event type. Some other required values are: the total number of cells (n) and the total number of event types (e), because in the formula the least common multiple (M) is used of n and e. In particular a $\vartheta = M/e$ parameter appears in the hash formula. The proposed hash function is :

$$h(e_i, t) = (te + i - (t \; div \; \vartheta)) \; mod \; n$$

The best way of understanding how it works is with an example. Let us think in a network with 100 cells (n) and 10 event types (e). Thus, being $M = 100$, $\vartheta = M/e = 10$. Table 3.1 shows the cell number associated with a particular event type for different time slot numbers. It can be appreciated that the home node for a particular event type remains in the same cell during ϑ time slots. After that time, it moves to the cell with the immediately lower identifier.

In addition, the authors try to solve the problem of selecting the node with more resources in a particular cell as the home node. They use the Node Contribution Potential [LXY05] as a measurement of the node's capacity. In particular, this value is a com-

bination of the storage capacity of the node and its remaining battery. Then, all the nodes inside a cell exchange their contribution potentials in order to calculate the cell potential, which is sent to the sink together with the location of the node with higher contribution potential within the cell. In turn, the sink selects a set of cells above a given cell potential threshold to be the location set for an arbitrarily long period, and broadcasts this set, together with the coordinates associated with it. During that period all nodes use that location set. Therefore, if n_L is the number of cells in the set, the new hash function is:

$$h(e_i, t) = (te + i - (t \; div \; \vartheta)) \; mod \; n_L$$

This paper enables changing the home node for a given event type over the time. Thus, the authors claim that this alleviates the hot spot storage problem. However, they do not explain what happens with the data already stored during the transition from the old home node to the new one, thus it seems that the data stored in the old home node is just lost. Furthermore, practical issues like the number of cells to be used, how they are assigned with an identifier, or how the sensors are able to know the identifiers of each cell, are not discussed.

3.5.2 A grid-based dynamic load balancing approach for DCS

This recent work [LSW10] also applies the idea that when a home node becomes saturated, a new one needs to be elected. The main difference between this work and the previous one [TWXD06], is that it selects a new home node to store the new production events, but still keeps the previous one since now consumer queries will get data from both, the old and the new home node, whereas [TWXD06] just assumes that after a given period a new home node is selected, but it does not discuss what happens with the data stored in the old home node. The authors in [LSW10] divide the network in a grid with cells of the same size in such a way that they assure that all the nodes inside a cell are within one hop distance. Then, each cell is assigned with positive coordinates (X_{grid_i}, Y_{grid_i}) called *grid ID*, which are based on its spatial location. Each sensor, given its own coordinates (x_i, y_i), is able to calculate its own grid ID. Therefore, each sensor knows its grid ID, its own coordinates and its grid neighbours coordinates. In addition, each node also has and share with its neighbours a virtual grid ID and virtual coordinates, which initially are equal to the actual grid ID and actual coordinates respectively.

Then when a producer detects an event of a given event type, it calculates the home grid ID (instead of the spatial coordinates whose closest node is the home node) by means of a hash function. Then, once the production message reaches the first node in the home grid, it sends the message to the other grid neighbours, which are only 1 hop far away. Since all nodes know their neighbours' coordinates, they compute who is the closest one to the center of the grid, which becomes the home node for that event type. The remaining nodes in the grid just discard the message. They use the virtual coordinates for these computations.

Next, the authors define several storage thresholds, *i.e.*, the first storage level is S_1 events (e.g., 30 events), the second, S_2 events (e.g., 60 events), until a maximum threshold that is the

maximum storage capacity. Once the home node reaches the first storage level, it changes its virtual coordinates to (∞,∞) and sends this update to its grid neighbours. Therefore, following the operational mode presented in the paper, the second closest node to the center of the grid becomes the new home node from that moment. It is possible, that at some point all the nodes reach the 1st level threshold. The last node reaching this threshold (the furthest one from the center of the grid) notifies this fact. Then the 2nd storage threshold is established in the grid, and all the nodes within the grid reset their virtual coordinates being again the same than their actual coordinates. However, following this mechanism at some point all the nodes within a grid could be saturated (reaching the last storage threshold). In such a case the authors propose something called extended grid, that is, using all the adjacent grids to the saturated one to select a new home node.

Finally, when a consumption query reaches the grid, it is received by all the nodes in the grid. Therefore, all the nodes with some stored data for the queried event type replies back to the consumer, which in this paper is always the sink.

This proposal solves in a smart way the storage problem by selecting a new home node when the previous ones reach an established storage threshold. This solution assumes that the network is divided in areas of the same size so that all nodes in an area are in range of each other. However, with this definition there would be many overlapped areas or nodes that could belong to several areas. Therefore, the definition of how the sensornet is partitioned should be more precise in order to be deployed in a real WSN.

3.6 Multi-Replication in Data Centric Storage

It has been already mentioned that one of the main drawbacks of the original Data Centric Storage (DCS) proposal is the hot-spot creation when a single node is in charge of a popular event. Local replication (using nearby nodes as replicas) does not fully solve the problem, just mitigate it, since still all traffic is directed toward a well defined area, which becomes a hot-spot zone.

Therefore several works in the literature have proposed to use several replication nodes, allocated along the network, in order to avoid the hot-spot area problem. These works proposes different replication structures: grid [RKY$^+$02, RKS$^+$03, JH08, GGC03, AK06], unstructured [TNK04], circumferences [SZG06] and mesh [LHZ04, SEP99, LSJ06, LSS07, LSS08, SLJ08]. Next, this section covers the most important works proposing multi-replication in DCS systems.

3.6.1 GHT with multiple replicas

Ratnasamy *et al.*, the original DCS authors, were already aware of this problem and proposed a grid-structured multi-replication system [RKY$^+$02, RKS$^+$03]. They assume a square sensornet field that is divided into a 4^d grid, being d the so-called *network depth*. The original home node is established as defined by GHT, that is, by applying a hash function

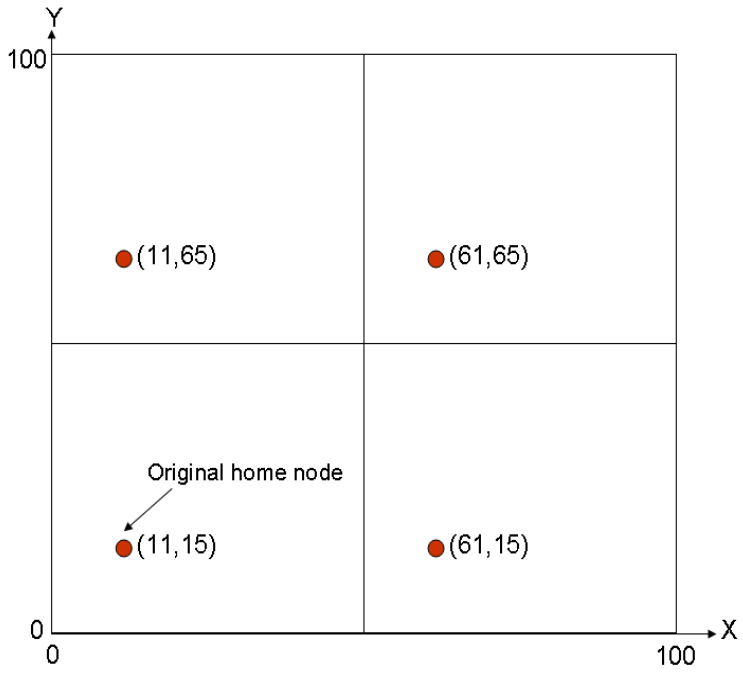


Figure 3.6: *Multi-replication grid in GHT network with depth* $d = 1$.

over the event type. After that, $4^d - 1$ mirror replicas are allocated in the same relative position inside each cell of the grid. Figure 3.6 shows a multi-replication scenario with d=1 and Figure 3.7 with d=2. Again, the coordinates shown in the figures refer just to space locations, thus the closest node to each location is the home replica node in that grid cell. This mechanism is recursive, because replicas of level d are also replicas of the $d + 1$ level.

With this system, a producer generating an event stores it in the closest replication node. Then, in order to retrieve all the data related to a particular event, a consumer queries the top home node ($d = 0$). This one in turn forwards the query to the remaining first level replicas ($d = 1$), that in turn do the same with their second level replicas ($d = 2$), and so on, in a recursive manner. The replies to the query travel the same path back to the original home node, which finally sends the information back to the consumer. Therefore, the information retrieval is based on a hierarchical routing, which also relies on GPSR. Figure 3.8 depicts an example scenario including both production and consumption communications.

There are no further discussion of multi-replication in [RKY$^+$02, RKS$^+$03], but just a final remark: the proposed muti-replication environment reduces the storage cost in front of using a single home node, but increases the query cost, since now consumer packets travel longer distances to get the information.

This work is the first one introducing the possibility of using several home nodes instead of only one. However, the authors do not provide any way to determine a suitable d in order to minimize the query cost. Furthermore, this work does not present any performance evaluation when using the multi-replication scheme.

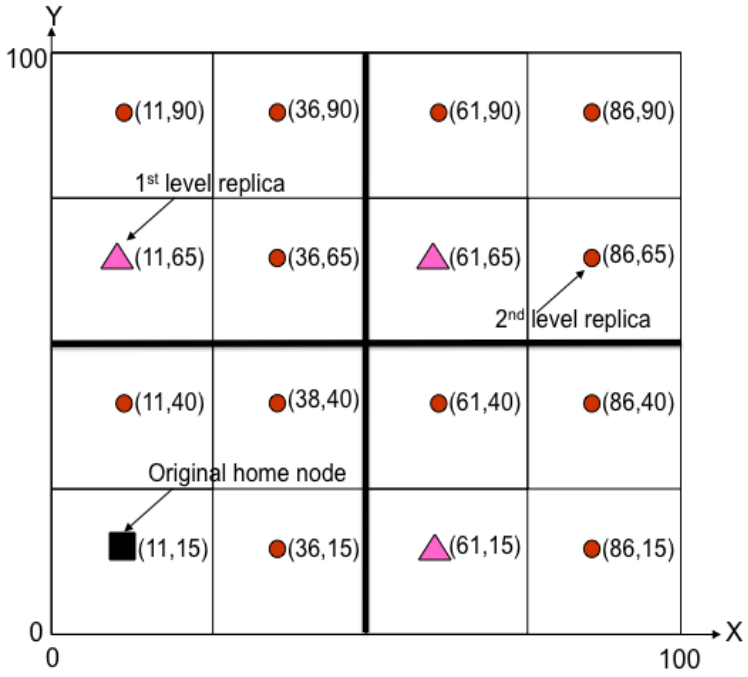


Figure 3.7: *Multi-replication grid in GHT network with depth* $d = 2$.

3.6.2 Resilient Data-Centric Storage

In this work [GGC03], a two-level replication strategy is proposed for control and data. The sensornet is divided into Z zones. Each zone could contain 3 types of nodes:

- **Monitor Node**: Each zone has one monitor node for each event type. The monitor nodes communicate among them by exchanging monitor maps that include control and summary information. The information contained in this monitor maps are: list of zones containing replica nodes, list of zones containing monitor nodes, event summaries and Bloom filters that are used for attribute-based queries (i.e. get temperatures over 30 Celsius degrees).

- **Replica Node**: Each zone has, at most, one replica node for each event type. In addition, if the replica node is present, it is the same node than the monitor one. Then, the extra function of a replica-monitor node is to store event data of the given event type.

- **Normal Node**: These are nodes that are neither monitor nor replica nodes.

The storage model is quite similar to the one proposed in GHT with multiple replicas: a node that generates an event forwards it to the monitor node of its zone. If the monitor node is a replica of that event, it stores the data, otherwise, since it has the list of zones containing replica nodes, it forwards the data to the closest replica.

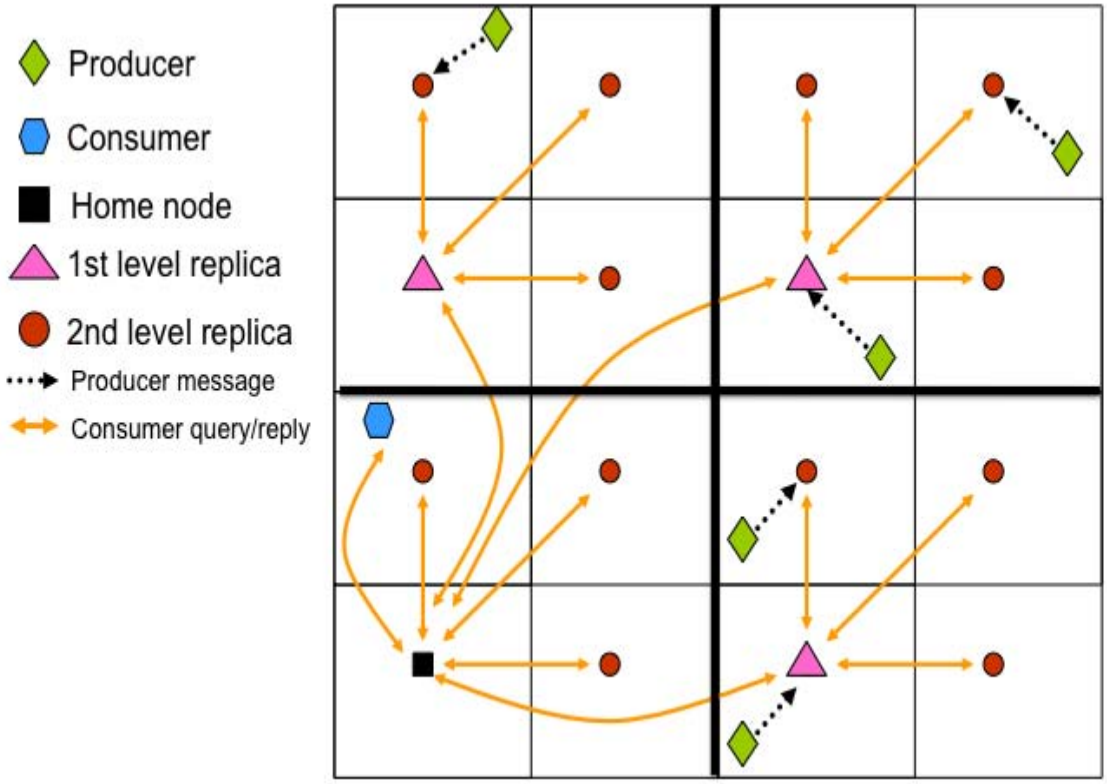


Figure 3.8: *Multi-replication GHT scenario using hierarchical routing.*

The main contribution of this paper is the different query types that it supports, which are more sophisticated than the one proposed in [RKY+02, RKS+03]. There are three types of queries:

- **List**: This is the one described in GHT [RKY+02, RKS+03]. A consumer sends a query requesting the data of a given event type to the monitor node in its area. This one forwards the query to all the replication nodes for that event type. Then, they reply directly to the consumer node with all the requested data.

- **Summary**: The consumer requests a summary of the event type information. It contacts its local monitor node that replies directly with the summary for that event type (e.g., average, max and min values of that event type).

- **Attribute-based**: It is a query requesting data for different events which match certain constraints in the attribute values. For that purpose Bloom filters are used since they provide a compact summary of data without false negatives. A consumer sends a query to its local monitor node for a given event type. Then the monitor node duplicates the query and sends it to all the replication nodes with Bloom filters' matches. All the replicas reply directly to the querying node.

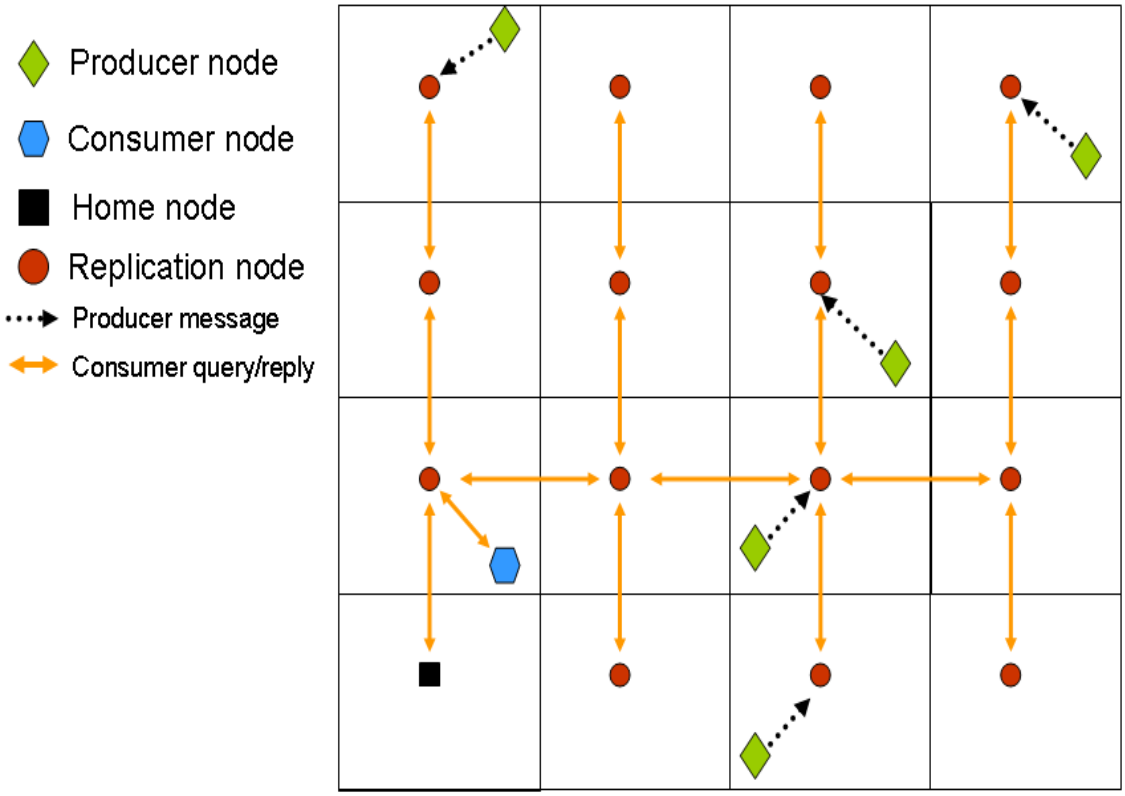


Figure 3.9: *ToW Write-one-Query-all mode using combing routing*

This work extends and complements the top level of a multi-replication DCS system like GHT with multiple replicas, but it also adds complexity since an extra role is included in the system, the monitor node, which implies more energy expenditures for that task. As previous proposals, this work does not provide any discussion about the appropriate number of zones, Z, to be used.

3.6.3 Tug of War (ToW)

None of the previous works study how many replicas should be allocated in a multi-replication DCS system depending on network conditions for a given event type.

Tug of War (ToW) [JH08] is one of the few works in the literature that has addressed this problem. ToW provides the optimal network depth (d), which leads to the number of replicas that should be used, following the 4^d grid introduced by multi-replication GHT [RKY+02, RKS+03], in order to minimize the overall network traffic.

ToW presents three new contributions to the multi-replication DCS research:

1. ToW provides an analytical model that computes the optimal value of the network depth, d, in order to minimize the network traffic.

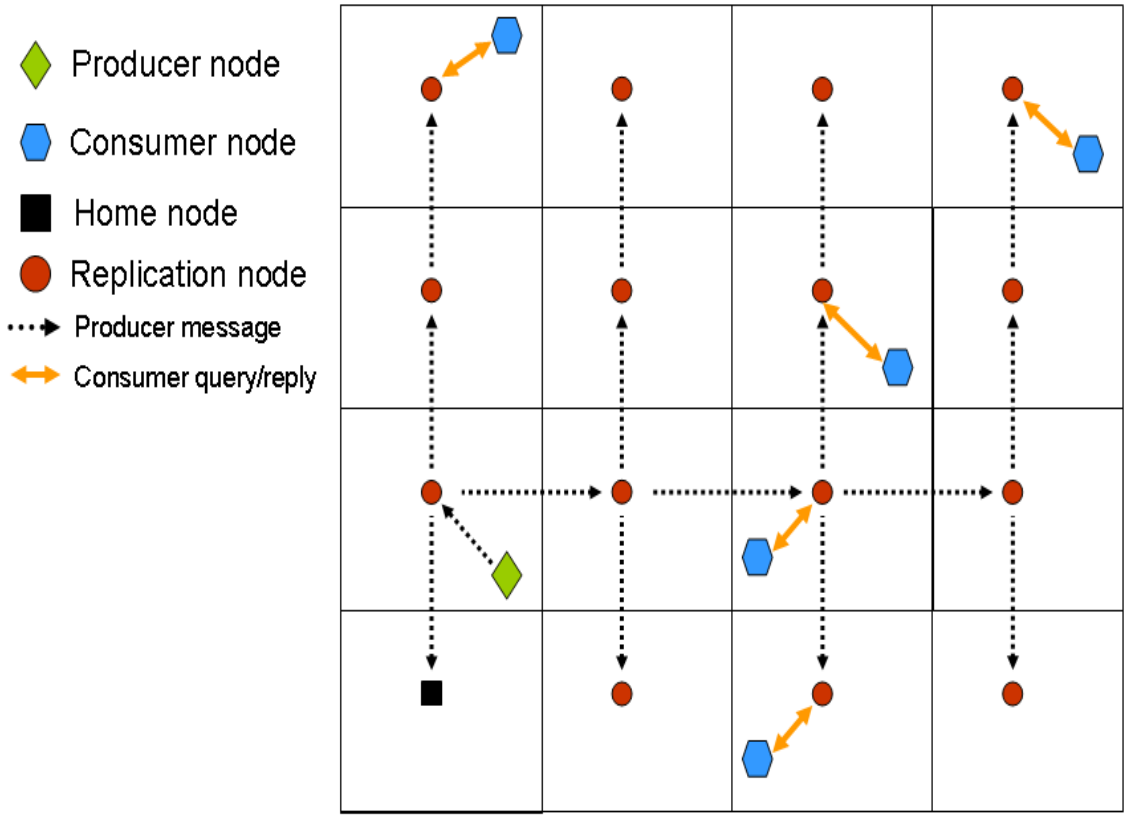


Figure 3.10: *ToW Write-all-Query-one mode using combing routing*

2. ToW describes two different DCS operation modes:

   - **Write-one-Query-all**: This is the same mechanism introduced in GHT with multiple replicas, which is only employed when there are more producers than consumers. Producers store the information in its closest replica. Therefore, a consumer query has to reach all replicas in order to retrieve all the information related to a given event type. An example of this operation mode is shown in Figure 3.9.
   - **Write-all-Query-one**: A novel mechanism is employed in the opposite case, when there are more consumers than producers nodes. Producers still send the data to the closest replica, that in turn forwards the information to the remaining replicas. Therefore, all the information related to a particular event type is stored in every single replica. Thus, a consumer only needs to query its closest replication node to retrieve all the information. Figure 3.10 shows an example of this operation mode.

3. ToW proposes a new routing mechanism, which is not hierarchical and recursive like the one proposed by multi-replication GHT. Since replicas are placed in a grid, the authors propose to connect them using a grid replication tree and call this mechanism

"combing routing". Therefore, when a replica receives some data from a producer, in the Write-all-Query-one mode, it forwards the information along its row and its column. The replicas belonging to the same row replicates the information along their own column (see Figure 3.10). In the case a replica receives a query from a consumer in the Write-one-Query-all mode, it follows the same strategy, the replica forwards the query to the remaining replicas using the combing routing. In this case the replies follow the same path back (see Figure 3.9).

ToW is analytically compared with multi-replication GHT, and it is shown to be more efficient due to the utilization of the combing routing instead of the hierarchical-recursive routing proposed by GHT with multiple replicas.

Moreover, ToW defines an analytical model to compute the optimal value of the network depth, d (which leads to the number of replicas to be used). The authors analyze a scenario where n sensors are uniformly deployed in a square region, thus each side of that region has roughly $\sqrt{n}$ nodes. The authors call f_e and f_q the frequency of producers' events and consumers' queries respectively. In addition, they define a parameter k that denotes the average number of sensors that detect the same event.

The distance between two random nodes inside a unit square is[1], $\overline{\delta} = 0.52$. This helps to model what is the distance from a consumer or producer to its closest replica.

3.6.3.1 Write-all-Query-one mode analytical model

The measured cost (in distance) for a consumer to send a query to its closest replica with $d = 0$ has a cost of $\overline{\delta}\sqrt{n}$. If d increases, then the distance to the closest replica is divided by a 2^d factor. Therefore, the consumer querying cost per unit interval could be defined as

$$C_{q_{W_{all}Q_1}} = f_q \frac{\overline{\delta}\sqrt{n}}{2^d}$$

Later, the cost of storing an event in all the replicas (overall producers' event cost per unit interval) allocated for that event type is:

$$C_{e_{W_{all}Q_1}} = f_e k \frac{\overline{\delta}\sqrt{n}}{2^d} + f_e k (4^d - 1)\frac{\sqrt{n}}{2^d} = f_e k \sqrt{n}(2^d - \frac{1-\overline{\delta}}{2^d})$$

It is composed by two terms: (i) the first one refers to the cost of forwarding the data from the producer to its closest replica, (ii) the second cost is due to replicate the received data in all the remaining replicas using combing routing. Since we are assuming a uniform deployment, the distance between any two replicas is the same, $\frac{\sqrt{n}}{2^d}$. In order to compute the total cost it is enough with knowing the number of branches used to replicate the information, that is, $4^d - 1$.

[1] (http://mathworld.wolfram.com/SquareLinePicking.html)

Therefore, the total cost for the Write-all-Query-one mode is:

$$C_{W_{all}Q_1} = C_{e_{W_{all}Q_1}} + 2C_{q_{W_{all}Q_1}} = f_e k\sqrt{n}(2^d - \frac{1-\overline{\delta}}{2^d}) + 2f_q\frac{\overline{\delta}\sqrt{n}}{2^d}$$

Notice that the querying cost is multiplied by a 2 factor so that the cost of the reply to the consumer query is also computed.

The optimal value of d in order to reduce the overall cost in the network is obtained by performing an optimization of the previous equation in function of d. Therefore, this optimal value, d^*, is:

$$d^*_{W_{all}Q_1} = \frac{1}{2}log\left(\frac{2\overline{\delta}}{k}\frac{f_q}{f_e} - (1-\overline{\delta})\right)$$

3.6.3.2 Write-one-Query-all mode analytical model

In this case the cost of storing an event (production cost) in the closest replica is immediate:

$$C_{e_{W_1Q_{all}}} = f_e k\frac{\overline{\delta}\sqrt{n}}{2^d}$$

The authors also use a symmetric cost model for the consumption cost, and they define it as:

$$C_{q_{W_1Q_{all}}} = f_q\sqrt{n}(2^d - \frac{1-\overline{\delta}}{2^d})$$

Therefore, the overall cost and d^* are:

$$C_{W_1Q_{all}} = C_{e_{W_1Q_{all}}} + 2C_{q_{W_1Q_{all}}} = f_e k\frac{\overline{\delta}\sqrt{n}}{2^d} + 2f_q\sqrt{n}(2^d - \frac{1-\overline{\delta}}{2^d})$$

$$d^*_{W_1Q_{all}} = \frac{1}{2}log\left(\frac{\overline{\delta}k}{2}\frac{f_e}{f_q} - (1-\overline{\delta})\right)$$

This model is symmetric to the Write-all-Query-one one. This is only possible if aggregation is performed in the query reply path. It is clear that the consumer query travels once across each replication tree branch, however it is not obvious that the same happens for the replicas replies, since if each replica sends one reply message, more than one message would travel several of the replication tree branches. Therefore, in order to provide a symmetric model, so that the distance cost of a query is the same than that of its reply, the only solution is to apply aggregation in the path back to the consumer node.

In order to apply this model, regular sensors need to know two parameters: k and the $\frac{fq}{fe}$ ratio. For k, the number of sensors measuring the same event, the authors do not present any discussion and assume that it is a well-know value in the network. However, in a real WSN calculating the actual k value is quite complex and usually it will not be homogeneous across the network. For the $\frac{fq}{fe}$ ratio, replication nodes are in contact among them, therefore they have a global overview of this ratio in the network. Then, the ratio is attached to each message sent along the consumption reply path, so those nodes that are in the range of the replying path could also learn it. In addition, each regular node includes this ratio when it forwards any message. However, this mechanism cannot guarantee that all producers and consumers are aware of the current $\frac{fq}{fe}$ ratio, which could lead to a worse performance since some sensors could evaluate an incorrect number of replicas.

3.6.4 Scaling-Laws for Energy Efficient Storage and Querying in Wireless Sensor Networks

This work [AK06] uses a constrained optimization framework to derive fundamental scaling laws for both unstructured sensor networks (which use blind sequential search for querying) and structured sensor networks (which use efficient hash-based querying). Therefore, the second group is referring to a Data-Centric Storage network.

Basically, they propose to use a uniform deployment of replication nodes in the network where nodes are either uniformly or randomly placed.

They propose to model the traffic cost in the network based on the next assumptions:

- N nodes are deployed with a constant density in a two-dimensional square area.
- A total of r_i rendezvous nodes are used to store r_i copies of each event following a uniform distribution
- For each event i, there are a total of q_i queries that are uniformly generated by the nodes in the network, therefore each producer event is consumed by q_i consumers.
- They assume lossless links and a free-interference medium.
- The total energy cost for storage and querying is assumed to be proportional to the total number of transmissions.

They describe a simple operation mode, where events are copied across all the replication nodes, so that consumer queries just need to reach the closest node in order to retrieve the information.

Then, they present a quite complex mathematical model for reducing the traffic cost. From this model they derive the formula of the optimal number of replicas in order to reduce the traffic cost in the network as:

$$N_r^* = 2^{\frac{-2}{3}} * q_i^{\frac{2}{3}}$$

Finally, they further develop its mathematical model to propose: scaling conditions for bounded storage, the scaling behavior of energy costs and the network scaling on fixed energy budget.

Although this paper presents a very solid mathematical work, it does not address practical issues that would be very interesting in order to complete the work:

- It does not explain how a uniform replication could be made in practice.
- The model relies on a q_i parameter that is the number of queries per event. However, the authors do not describe how this parameter is known by the network nodes.
- They do not specify how consumers and producers are notified with the current number of replicas within the network in order to interact with the closest one.
- They do not present any validation of its model either via simulation or with a real testbed.
- Finally, as we will demonstrate later in this Thesis, using the optimal number of replication nodes provided by this model leads to a higher network traffic compared with other solutions such as ToW [JH08].

Therefore, we believe that it is a very interesting work from the mathematical point of view, but it lacks of practical discussion.

3.6.5 Double Rulings for Information Brokerage in Sensor Networks

Most of the previous multi-replication DCS proposals employ a structured grid-based replication. This work [SZG06] employs a much more complex approach that uses circumferences to replicate the data. The basic idea is: a producer generates a big circumference between itself and the home node, storing the information in all the nodes along the path. A consumer of that event type does exactly the same, so that, at some point, this two big circumferences crosses each other (at least in the home node). Thus, the nodes located at the crossing point can reply to the consumer with the data.

It is obvious that all the production curves related to a given event type have at least a common node, which is the home node. However, the paper demonstrates that there is a second point where all the production circumferences for a given event type collide. Therefore, there are at least two nodes that store all the events of a given type, meaning two global replication nodes (these are always two nodes in contrast with multi-replication GHT and ToW that can adapt that number). An example of the production curves generation is shown in Figure 3.11

Another contribution of this paper is that it defines several ways to retrieve the data depending on the application's interest:

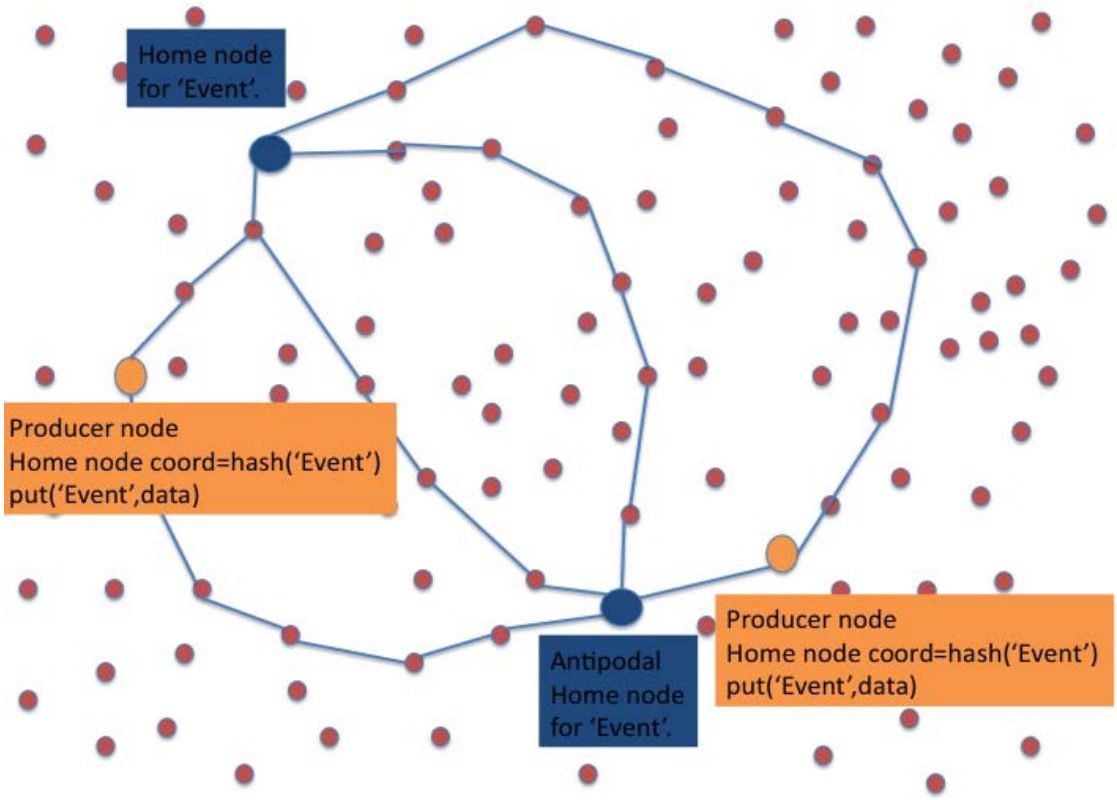


Figure 3.11: *Double Ruling production curves.*

- **GHT retrieval**: A consumer accesses one of the two home nodes (the closest one) and retrieve all the information.
- **Distance-Sensitive retrieval**: It defines a mechanism to guarantee an upper bound distance when a consumer wants to retrieve the data from a particular producer. Then the consumer has to find the crossing between its consumption circumference and the suitable production circumference.
- **Aggregated data retrieval**: The consumer uses a closed curve that separates both home nodes, collects all relevant data and computes the aggregates.
- **Double rulings retrieval**: It defines the full power data retrieval rule. That is, the consumer travels along any great circle and is able to retrieve all the data stored in the sensornet.

The authors compares its proposal in front of GHT without replication [SRK$^+$03] and GHT with replication [RKY$^+$02, RKS$^+$03] using a network depth $d = 1$. Table 3.2 shows the average producer and consumer costs in number of hops for the three approaches. The authors conclude that double rulings clearly outperforms GHT without and with replication. However, this may be not true in all the cases, since it very much depends on the number

	Producer Cost	Consumer Cost
GHT without replication	44.9	45.6
GHT with replication ($d = 1$)	172.4	17.4
Double Rulings	77.5	29.0

Table 3.2: *Average producer and consumer costs in number of hops for GHT, multi-replication GHT and Double Rulings.*

of producers' events and consumers' queries. For instance in a scenario with many producers' events and just few consumers' queries, GHT without replication would provide better results than Double Rulings just by following the data provided in the paper.

The main drawback of this paper is the complexity of implementing the proposed routing. Forwarding messages following a curve is far from being a standard routing mechanism and surely adds complexity to the implementation. Furthermore, as it has been already mentioned, Double Rulings can only employ 2 global replicas, which could be not enough in some scenarios, whereas other solutions like ToW or GHT with multiple replicas could adapt the number of replicas to the existing traffic load.

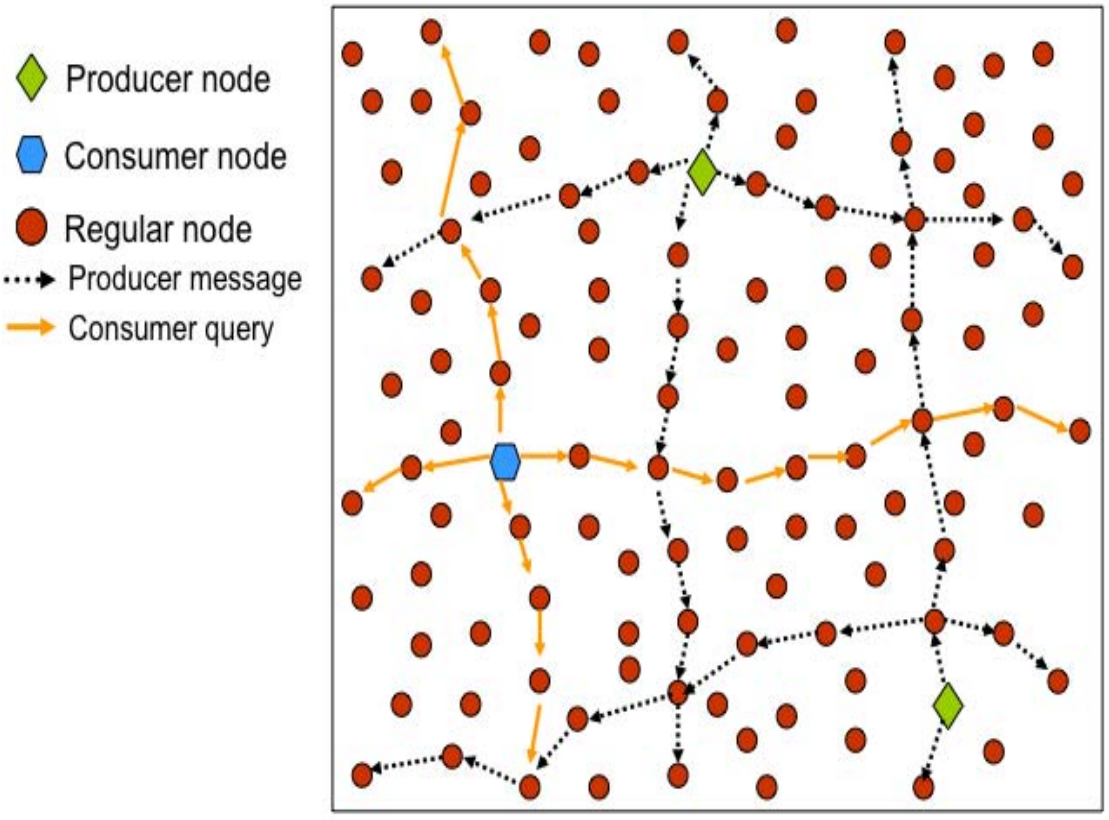


Figure 3.12: *Mesh Replication.*

3.6.6 Mesh Replication Proposals

There are other works in the literature [LHZ04,SEP99,LSJ06,LSS07,LSS08,SLJ08] that can be seen as an alternative to DCS systems. In this section we provide a brief overview of their basic idea, to let the reader understand how these systems work. A producer that has measured an event, forwards a message in the North (N), South (S), East (E) and West (W) direction, and all the nodes in these paths store the data. In turn, a consumer that wants to retrieve this data, also sends the query in the North, South, East and West directions, so that it is sure that the query will cross at least one of the nodes storing the data of interest. In addition, if there are several producers of a given data type, the consumer will be able to retrieve all the data with just one query. Figure 3.12 shows an example of these proposals.

The authors in [LHZ04] present a mathematical analysis and a complete performance evaluation via simulation of this mechanism. A proposal to find the position of a mobile node within the network is studied in [SEP99,LSJ06] and [SLJ08]. The authors propose that the mobile node updates its location sending a message in all the cardinal directions (E, W, S and N), so that all the nodes in those paths store the current position of that particular mobile node. Then, a node that wants to find the current position of the mobile node also sends a query in the four cardinal directions. In [LSS08], a distance-sensitive service discovery is proposed based on the mesh replication structure. Finally, the authors in [LSS07] use the proposed scheme and present a Mesh-based Sensor Relocation protocol (MSRP) for mobile sensor networks. This protocol maintains the sensornet sensing coverage by replacing failing nodes with others that are close by.

3.7 Other issues

Previous sections have covered the main issues related to Data-Centric Storage (DCS): multi-replication, routing and storage for DCS systems. However, there are more works in the literature that focus in different aspects of DCS. This section briefly introduces the most relevant ones.

In [XCCX08] the authors try to find the optimal home node position based on the event production and query consumption distributions. They define a mathematical model in order to solve the problem. However, the implementation of the defined algorithm is not distributed, but centralized, since the base station calculates the optimal home node location. For that purpose, it is required that the base station has full knowledge of event and query distributions for a given event type, that initially is something that is not required by other DCS proposals.

There are several works [ZCLP03, JX05, LLL06, Yu09] that compare different storage policies and analyzes which is the suitable relation between queries and events to apply each storage mode.

A new storage mode (different than local storage, DCS or external storage) is introduced in [ZCLP03] and later used in [JX05] and [LLL06]. This mode stores the data in a node close to the producer (or locally in the producer) and notifies this to the home node. The home

node has an index list with all the producers storing data for a given event type. Therefore, when it receives a consumer query, it retrieves the data from all the producers in the list and replies back to the consumer. In all these works the queries are generated from the sink.

The author of [Yu09] proposes an adaptive storage policy switching between local storage and data-centric storage depending on the event-production to query-consumption ratio. In this case, the queries are generated from several mobile sinks within the sensornet. In addition, the network is divided into multiple grid cells, and each cell decides locally whether to apply data-centric storage or local storage. The author describes the full communication model as well as the policies to switch from one storage mode to the other. This work assumes that each cell is assigned with an identifier and the messages are sent to a particular cell based on that identifier. In addition, it is also assumed that each sensor knows to which grid cell it belongs to.

There are some works [LKGH03, APC06] that employ DCS to support multi-dimensional data queries in WSNs. In order to achieve that goal, they create a *K-D tree* where nodes are assigned with bit-code identifiers that are related to their spatial location in the network. Then they split the events value range and assigns a bit code to a particular event type sub-range of values. Therefore, the multi-range queries are generated by a sequence of different event type bit codes. This sequence results in another bit code that is sent to the node with the closest bit-code identifier, thus creating the DCS system.

Finally, there are a set of works: Q-NiGHT [ACNP07b], DELiGHT [ACNP07a] and [AC09] that propose to use a novel hash function based on a rejection method [Neu51]. This function is aware of the sensor distribution and avoids to provide a home node location in areas with very low node density (e.g., borders of the network). Furthermore, these works add QoS to DCS. The authors claim that, depending on the event type, the number of local replicas should be different. Then, when a producer pushes an event towards the home node, it includes a Q parameter that indicates the number of local replicas to be used. In turn, the home node chooses the replicas from the surrounding area. Q-NiGHT and DELiGHT propose to replicate each event in all the replication nodes, whereas [AC09] applies erasure codes [Rab89] that codify and distribute the data in smaller pieces, storing each one in different replicas. Therefore, in order to reconstruct the information, a consumer only accesses a subset of the replicas, thus retrieving only a part of the codified data, but it is enough to reconstructs the event data. This mechanism better spreads the storage load among the replicas.

3.8 Summary and Research Challenges

This chapter has presented the most relevant works in Data Centric Storage (DCS) for Wireless Sensor Networks (WSNs) including routing, multi-replication, storage policies, *etc.* Table 3.3 summarizes the main proposals and their contributions in the field of Data Centric Storage.

However, we believe that there are still some research challenges that need to be addressed.

Proposals-Contributions	Routing	Multi-replication	Storage
GHT	GPSR	Single home node	-
GHT with multiple replicas	GPSR	Grid-based (4^d)	-
RendezVous Regions	Region-oriented	Single home node	-
HVGR	Landmarks + Region-oriented	Single home node	-
pathDCS	Landmarks + Path-oriented	Single home node	-
Dynamic DCS	GPSR	Single home node	Changes home node over the time
Grid-Based Dynamic	Geographic Routing + Local decision	Single home node	Changes home node when saturated
Resilient DCS	GPSR	Grid-based (z cells) Different type of queries	-
ToW	GPSR + Combing routing	Grid-based (4^d) optimal d	-
Scaling Laws	Not specified	Uniform replication	Theoretical boundaries
Double Rulings	Circumference Routing	Circumference replication	-

Table 3.3: *Summary of DCS proposals.*

- A first research topic that still presents some challenges is the *storage analysis for DCS systems* (see section 3.5). Storage in DCS has been studied in very few works [TWXD06, LSW10] in the literature. These papers basically state that when the home node is saturated a new one is chosen, and later another one and so on. However, they do not consider that usually sensor data is only useful during a given time window. In addition, there could be many different applications in a sensornet, thus there are some probability that a given node is selected as home node (or replica) for several applications. Therefore, it would be very interesting to study the maximum number of applications (event types) that could be deployed at the same time in a WSN due to storage constrains.

- In the area of Multi-Replication DCS (Section 3.6), we have found different structured replication techniques, mainly represented by multi-replication GHT and ToW, but they are not adaptive enough due to its 4^d replication scheme. Moreover, all these works assume a square sensornet and are tied to a grid structure that requires some previous knowledge on grid-cell number and size. In addition, structured replication mechanisms based on grid replica deployment limit its applicability to very few network shapes, such us square or rectangles. We believe that an *unstructured replication scheme* that selects the optimal number of replicas without depending on a grid structure would reduce the complexity of the structured replication proposals, as well as improving its performance and adaptivity to different WSAN shapes.

- Structured replication is only valid for WSANs and/or WSNs based on geographical information. Otherwise, once the home node is chosen it is not possible to deploy replication nodes in grid or structured manner. Therefore, in networks using a routing protocol that is not aware on location information, none of the structured replication mechanism could be used, and thus DCS cannot be applied. However, DCS could be employed in networks without geographic information if an unstructured replication mechanism is utilized. For instance, if the nodes are identified based on an address (e.g. IEEE 802.15.4 short addresses), the hash function over the event type could provide an address instead of a geographical coordinates. Then, the node with that address (or the node with the closest address to the one generated by the hash function) would be the home node. In addition, if more replicas are required, more addresses would be obtained with the hash function, selecting extra home nodes. However, those nodes cannot be deployed following a given structure just by using a hash function, but there would be randomly placed in the network. Therefore, since there are many routing proposals that do not rely on geographic information, it would be very interesting study unstructured replication system (i.e. random replication).

- There is an inherent problem to DCS systems that has been only mentioned in [TNK04] but not fully studied. This problem is the *unfair energy consumption distribution that DCS creates among the sensors* in static WSNs. It is clear that home nodes (in single-replica DCS scenarios) and replication nodes (in multi-replication DCS scenarios) incur in higher energy expenditures than other nodes within the network. Then, we believe that the role of the home node should change over the time in order to balance the energy consumption among all the nodes in the sensornet. We think that improving this energy balance will extend the sensornet lifetime significantly. Obviously, changing the replicas over the time balances the energy expenditures among the sensors, but it is not free of cost. Changing the replication nodes without information loss requires the data stored in the old replicas to be moved to the new ones. Therefore, it should be studied what is the impact of this additional cost on the sensornet lifetime. In addition, a dynamic DCS system requires new mechanisms and protocols, and in order to be practical they should not add too much complexity. It must be highlighted that this research challenge is only interesting in static WSNs because, if the sensors are mobile nodes, the home node (or replicas) already changes over the time due to node mobility, and thus no additional mechanism needs to be applied.

- We have already mentioned in this paper the Autonomous Wireless Sensor and Actor Networks (AWSANs) concept, where the network is self-operated without the necessity of a sink or gateway that connects it to the external world. Most of the research papers presented in this survey are focused on standard WSNs, and assume that queries are generated from a single point, the sink. We believe that the interest on DCS will increase when AWSANs become more relevant. In such a case, many different event types will be used in the network in the different planes: control, management and data. Therefore, DCS seems to be a very efficient storage solution for AWSAN.

- Most of the proposals only provide theoretical results since they propose algorithms, analytical models, how to deal with storage saturation in the home node, etc, but

they do not consider practical issues. For instance: (i) in case a replica fails how is this solved?, (ii) how consumers and producers are informed about the number of replication nodes used in the network? (iii) how consumers and producers are updated in case the number of replication nodes changes? (iv) if a replication node fails, does the backup node recover the information from other replicas or it just stores the data from the moment it was replica? In addition, when analysing and proposing solutions taking into account practical issues, maybe some extra overhead is added. Then, a performance evaluation of these solutions is required to really understand the impact of applying DCS in practice. Therefore, from our point of view, it is crucial to perform a serious analysis of practical issues. This is a key step towards the real usage of DCS in WSNs and WSANs,

- All the works proposing multi-replication schemes assume a homogeneous distribution of production events and/or consumption queries across the network. In such cases, it makes sense to allocate replicas (e.g. with grid-structured replication) to cover most area of the network. However, there would be many scenarios that do not satisfy this homogeneous traffic assumption, thus placing replicas covering all the network area is probably not the best option. Instead, allocating the replicas close to those areas where there are manyproduction events and/or consumption queries could lead to better results. To the best of our knowledge, there is no work in the literature that has studied DCS in heterogeneous traffic scenarios. We believe that this is a interesting area of research for multi-replication DCS networks.

- Finally (and linked to the discussion of DCS practical issues), DCS has been a very interesting topic for the research community and now it is time to start focusing in real implementations for DCS in order to demonstrate that it is a feasible mechanism, and it is able to support many applications running in parallel on top of it.

Part III

Dynamic multi-replication DCS proposals: Analysis, simulation and implementation

Chapter 4

Problem Statement

This chapter briefly describes which are our main objectives, in what type of networks we are focusing and which are the basic assumptions of our work.

The focus of this PhD Thesis is on Data Centric Storage (DCS) networks running distributed applications operating autonomously over a Wireless Sensor and Actor Network (WSAN) without external intervention or communication. For example, an irrigation control application for a remote field might be based on sensors that monitor the temperature and humidity of the soil, i.e., produce information, and actors that consume this information to coordinate the closing and opening of water valves. The objective is to have a network that can operate fully autonomous with only intermittent connection to the outside world, which may periodically query historical data to assess the efficiency or operational state of the system. In addition, we want to balance the energy consumption among the network nodes in order to extend the network lifetime as much as possible.

In order to achieve these objectives, we propose two novel Multi-Replication DCS proposals: Quadratic Adaptive Replication (QAR) and Random Replication, which try to minimize the overall network traffic. In addition, we also provide a framework that enables the change of replication nodes over the time to fairly distribute the energy consumption.

We assume the existence of a shared ontology (system of categories) for the applications (types of information services) supported by and for the network. The name of an application is known by all the consumer/producer nodes that are part of the application. Information may be periodically generated/updated by producers and is consumed on an as-needed basis by consumers. We assume there is little or no locality in the production and consumption associated with a given application running in the region of interest. At this point, the production events and consumption interest associated with a given application will be assumed to be roughly spatially homogeneous. We consider a static Wireless Sensor and Actor Network (WSAN) that involves a large number of homogeneously distributed nodes, which transport information by relaying packets across neighbouring nodes. Nodes are assumed to know their spatial location within the operational area as well as the network dimensions and realize a geographic routing service (e.g. GPSR [KK00]) that is able

to unambiguously route packets to the closest node to a given spatial location. Nodes are further able to compute hash functions of application names along with other attributes and able to calculate the distance between locations in the network. This is a fairly low computational requirement for addressing and routing. In addition to low computational capacity the WSAN's nodes are also assumed to have limited resources:

- Limited energy: Sensors run on batteries or have limited energy harvesting capability.
- Limited capacity: Sensors have a limited communication capacity thus excessive network traffic should be minimized.
- Limited storage per node: Since sensors have limited memory capacity, they can only serve simultaneously as a replication node for a limited number of applications.

These resource limitations are the primary motivation to devise a model which permits an assessment of trade-offs and optimization of our Multi-Replication DCS proposals.

Finally, we define two different operation modes depending on the amount of production events and consumption queries. On the one hand in the Consumption-dominates-Production mode, production events are copied in all the available replicas, so that consumer nodes just need to access the closest replica in order to retrieve all the information of a given event type. On the other hand, in the Production-dominates-Consumption mode, producer nodes just push the data to the closest replica, while consumer queries need to reach all the replication nodes in order to retrieve the information.

Chapter 5

Quadratic Adaptive Replication (QAR)

Multi-replication is one of the most important research lines of DCS. In particular, Tug of War (ToW) [JH08] is one of the most interesting works because along with Scaling-Laws [AK06] are the only ones that define a model that provides the optimal number of replicas to be used. However, as it will be demonstrated in Section 6.3.1, by applying both models in the same scenario, ToW leads to a much lower traffic utilization than Scaling-Laws. Therefore, in this section we will validate our proposal by comparing it in front of ToW.

ToW is based on a geometric replication formula, that calculates the number of replicas as $N_r = 4^d$, being *d* the so called *network-depth*. Once the optimal d value (d^*) has been defined, the optimal number of replicas to be used is also known. The main drawback of this replication scheme (that is also used in GHT with multiple replicas [RKY$^+$02, RKS$^+$03]) is that it is not very flexible, because the number of replicas could be only one of these values: 1, 4, 16, 64, 256, 512, 1024, *etc*. Therefore, there could be cases where, for instance, neither 16 replicas nor 64 are the optimal value, but some other value in between.

In order to provide a more flexible solution that allows selecting a more adaptive number of replicas, we have designed the so-called Quadratic Adaptive Replication (QAR) scheme [CUnRL10], where the number of replicas follows a quadratic formula, $N_r = d^2$. QAR is also a grid-based replication scheme that uses the *combing routing* proposed by ToW (detailed in section 3.6.3). QAR is demonstrated to be much more adaptive than ToW, since now the number of replicas that can be selected is: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, *etc*. Figures 5.1 and 5.2 show an example of QAR for 9 and 16 replicas respectively. It must be noted that ToW is not able to use 9 replicas, but jumps from 4 to 16.

We also present an alternative model to the one of ToW, which provides the optimal number of replicas to be used in order to minimize the overall network traffic. Our model considers the same cases proposed in ToW: when the traffic related to queries overpass the one generated by events ("Consumption-dominates-Production") and the opposite case ("Production-dominates-Consumption"). These cases are the same described in ToW as "Write-all-Query-one" and "Write-one-Query all", respectively.

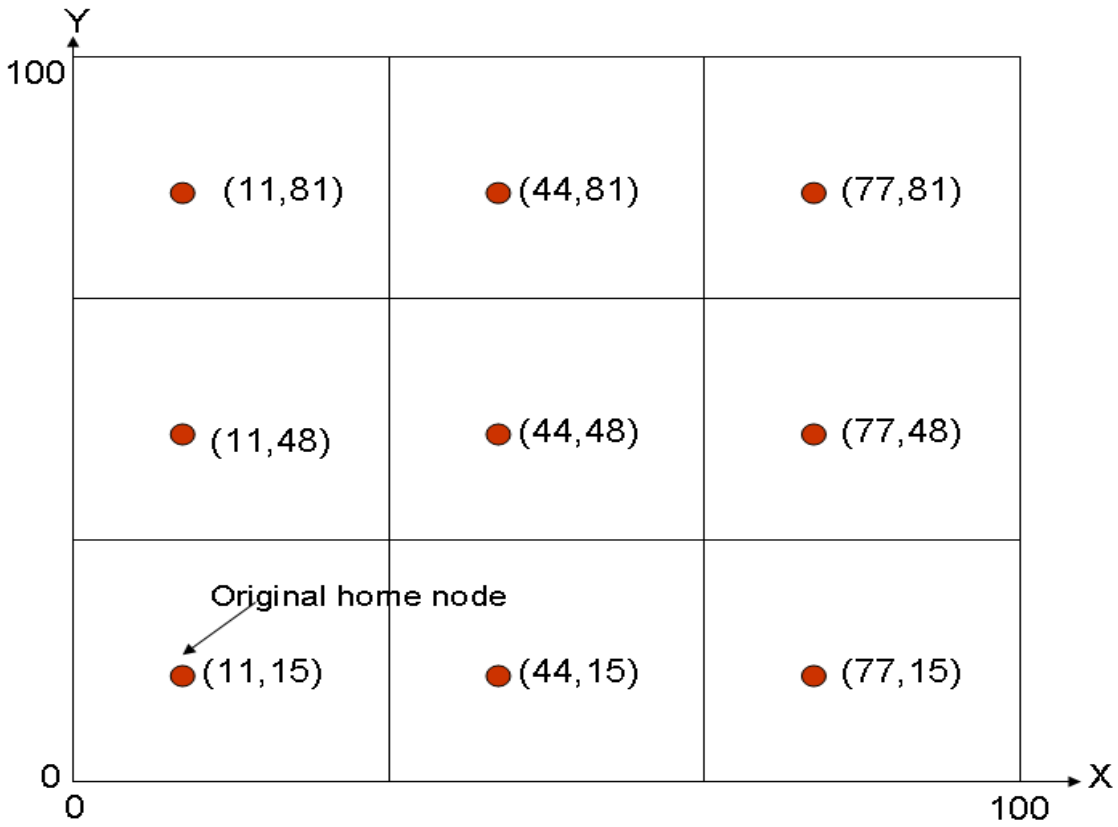


Figure 5.1: *Multi-replication grid of Quadratic Adaptive Replication (QAR) with 9 replicas.*

When there are many consumer queries (Consumption-dominates-Production), it is worthy to replicate the few producer events in all replicas using the combing routing mechanism, while consumers only need to access the closest replication node in order to retrieve all the data. On the other hand, when there are few consumers queries (Production-dominates-Consumption), replicating all data as in the previous case is quite inefficient. Instead, in this case it is better that producers just store the events in its closest replica, while the few consumers queries need to reach all replication nodes using combing routing in order to retrieve all the data from a given event type.

Finally, it must be noted that our mathematical model provides an optimal number of replicas that is a real number (this also happens in the ToW model). Thus, in order to perform the rounding, we evaluate the overall traffic cost for the immediate higher and lower integer number obtained from the quadratic series N_r^*=1, 4, 9, 16, 25, *etc.* For instance, if the model output is $N_r^* = 29.31$, we first evaluate the traffic cost with $N_r = 25$ and $N_r = 36$ and then select as the final number of replicas the one providing the lower traffic cost. It must be noted that in the same case, ToW could only choose among $N_r = 16$ or $N_r = 64$.

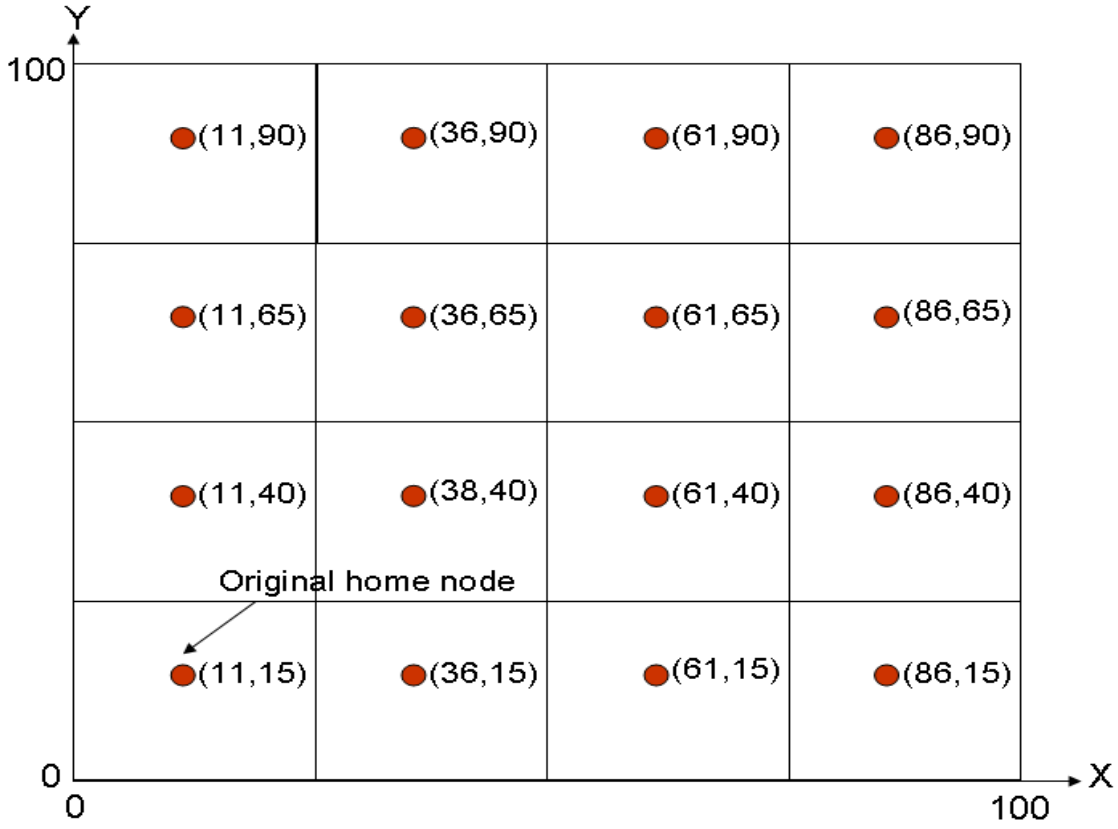


Figure 5.2: *Multi-replication grid of Quadratic Adaptive Replication (QAR) with 16 replicas.*

5.1 QAR Analytical Model

In this section we present our analytical model that will enable us to establish the optimal number of replicas so as to minimize the overall network traffic.

We shall start by developing a model for the overall network traffic generated by consumption and production nodes across the network. The overall metric is the total traffic load, measured in bits-meter/second that is supported by the network. Recall that in an ad hoc wireless network traffic load can not simply be measured in terms of bits/sec, but must also account for the distance packets must travel, since this involves relaying, and thus resources along the path. Measuring network load in terms of bits-m/s captures the amount of traffic and the distance that must be traveled. In turn, we assume the power expenditures for transporting traffic to be roughly proportional to the overall network traffic.

The locations of nodes of the wireless sensor network are fixed and modeled by a homogeneous spatial Poisson Point Process Π_n [SKM95], *i.e.*, a random set of points on the plane, with intensity λ_n nodes per unit area. Production and consumption events generated by network nodes are in turn modeled via independent homogeneous spatio-temporal Poisson Point Processes Π_p and Π_c each one with intensities λ_p and λ_c events per unit time and unit area respectively. There are λ_r nodes per unit area placed in a grid fashion serving as

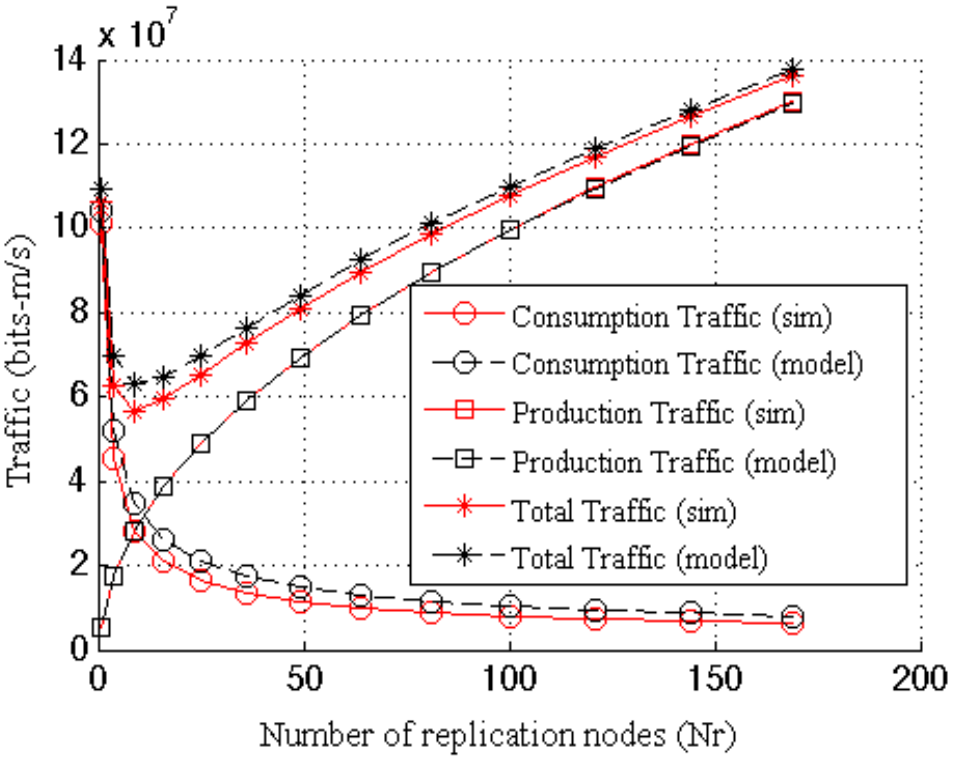


Figure 5.3: *Overall network traffic generated by QAR using different number of replicas for the case when Consumption-dominates-Production (A = 1000 × 1000 m^2, N = 5000 nodes, α = 200 bits/query, β = 100 bits/event, N_p = 100 producers; N_c = 1000 consumers, Iterations = 50).*

replication nodes. Although the model corresponds to one on an infinite plane, we shall restrict attention to a fixed region $\mathcal{A} \subset \mathbb{R}^2$ modeled as a convex set with area $A = |\mathcal{A}|$, and then optimize operation on $\mathcal{A}$, roughly ignoring edge effects. On average there are $N_r = \lambda_r A$ replication nodes in $\mathcal{A}$. Thus, the *quadratic network depth* could be defined as, $d = \sqrt{N_r}$.

5.1.1 Consumption-dominates-Production model ($\lambda_c > \lambda_p$)

When Consumption dominates Production, consumers retrieve data from the closest replication node. Then the average distance between a node and its closest replica is roughly the average distance between two nodes inside one of the replica's square cells. This distance can be modeled as $\frac{\delta}{\sqrt{\lambda_r}}$ with $\delta = 0.52$ (the same value used in ToW). Taking into account that the total number of consumption events in a sensornet of area A, is $\lambda_c A$, the next expression defines the overall network traffic associated with consumption traffic (T_c):

$$T_c(\lambda_r) = \alpha \lambda_c A \frac{\delta}{\sqrt{\lambda_r}} \text{ bits-m/s},$$

where α is a proportionality constant corresponding to the average number of bits per consumption event that are exchanged between the consumer and its nearest replication node.

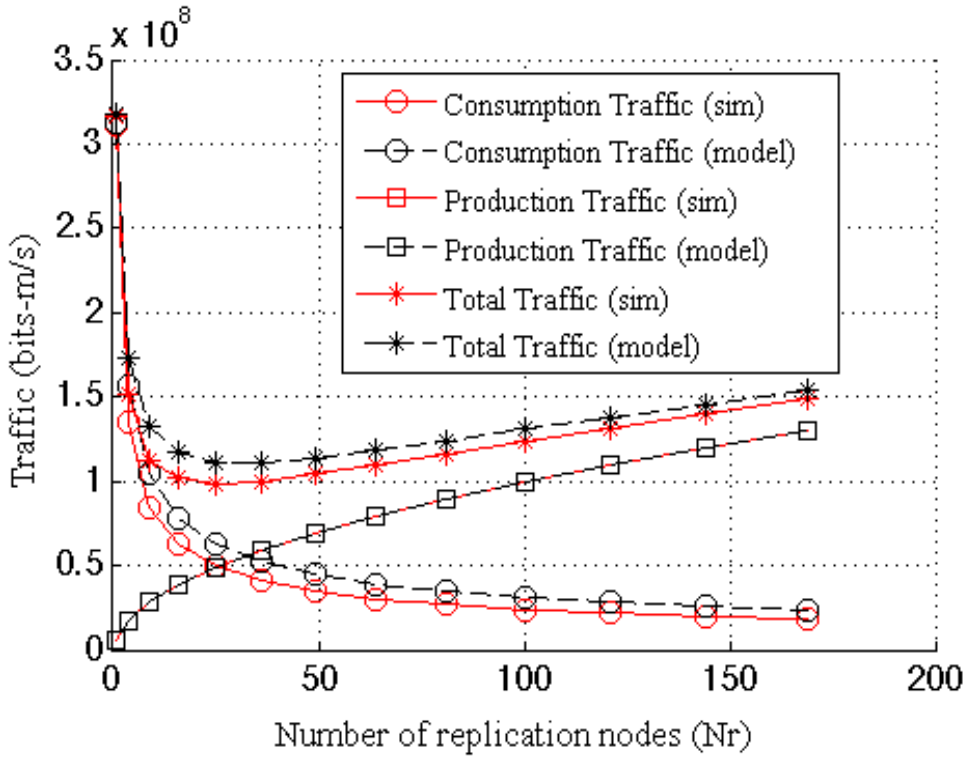


Figure 5.4: *Overall network traffic generated by QAR using different number of replicas for the case when Consumption-dominates-Production (A = 1000 × 1000 m^2, N = 5000 nodes, α = 200 bits/query, β = 100 bits/event, N_p = 100 producers; N_c = 3000 consumers, Iterations = 50).*

Next we consider the replication cost when data is produced. This cost is composed of two different factors: the first one is due to a producer sending information to its closest replica, and the second cost due to the replication process from that closest replica to the remaining replication nodes.

The former cost is calculated in the same way than the consumption cost. The latter needs to consider all the branches in the generated combing replication tree. Since we rely on a grid-replication structure, the length of all the branches is exactly the same. It is the distance between the center of two neighbour square cells, that is, the distance of one side of the square cell, which is $\frac{1}{\sqrt{\lambda_r}}$ in a square-unit area.

Therefore, the overall production traffic in a square WSAN of area A, is composed by: (i) the number of branches to be traversed by the replicated information, which is the number of replicas minus 1, thus $\lambda_r A - 1$ ($N_r = \lambda_r A$); and (ii) the number of production events in the sensornet, $\lambda_p A$. Thus, the overall production traffic i:

$$T_p(\lambda_r) = \beta\lambda_p A\frac{\delta}{\sqrt{\lambda_r}} + \beta\lambda_p A(\lambda_r A - 1)\frac{1}{\sqrt{\lambda_r}} \text{ bits-m/s},$$

where β plays the same role for production cost than α in the consumption cost, that is, the average size in bits of the production event messages.

Thus, the overall network traffic is computed as $T_t(\lambda_r) = T_c(\lambda_r) + T_p(\lambda_r)$. By optimizing this expression we are able to find the optimal number of replicas (N_r^*) in order to minimize the overall network traffic:

$$N_r^* = \lambda_r^* A = \delta \frac{\alpha \lambda_c}{\beta \lambda_p} - (1 - \delta)$$

In order to validate this distance-based model we have run simulations in a scenario of area A=1000x1000 =$10^6 m^2$, with N=5000 nodes randomly deployed and being $\alpha = 200$ bits/query and $\beta = 100$ bits/event. Then we evaluate two different scenarios in order to validate our model for different values: (i) A scenario with N_p=100 producers and N_c=1000 consumers selected at random from the 5000 nodes, which leads to a $\lambda_p = 100 * 10^{-6} \frac{events}{s*m^2}$ and a $\lambda_c = 1000 * 10^{-6} \frac{queries}{s*m^2}$. Each producer and consumer generates 1 event or query per second. This means a λ_c/λ_p ratio equal to 10. (ii) We use the same parameters but increasing the number of consumers up to N_c=3000. This generates a λ_c/λ_p ratio equal to 30. In both cases we evaluate the consumption, production and overall network traffic for different number of replicas: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144 and 169. We use 50 different sample scenarios for each of the evaluated cases and present the average value of the 50 samples as result.

Figures 5.3 and 5.4 show the model validation via simulation (confidence intervals are not shown because they would be undistinguishable). The results demonstrate that the model is accurate enough and can be used to find the optimal number of replicas to apply QAR. In addition, as indicated by the formulas, the larger the λ_c/λ_p ratio the higher the optimal number of replicas. Then, Figure 5.3 shows the case in which that ratio is 10, and the figure shows that both the model and the simulation define 9 as the optimal number of replicas. However, when the ratio grows up to 30, as it happens in the scenario evaluated in Figure 5.4, then the optimal number of replicas is 25.

5.1.2 Production-dominates-Consumption model ($\lambda_p > \lambda_c$)

In this case producers just push the data to the closest replication node, whereas a consumption query has to reach the closest replica first, which then routes the query to the rest of replication nodes (using combing routing) that finally reply back with the requested information if any is available. Therefore, we can use the same model than in the previous case, but interchanging consumption and production costs, but only if we assume that the distance traveled by the query to all replicas is the same as the distance of all the reply messages. We already mentioned when introducing ToW (see section 3.6.3) that this is only possible if aggregation is assumed. Therefore, in order to fairly compare our proposal with ToW, we follow the same assumptions considered by ToW.

Then the consumption and production costs and N_r^* are in this case:

$$T_c(\lambda_r) = \alpha \lambda_c A \frac{\delta}{\sqrt{\lambda_r}} + \alpha \lambda_c A (\lambda_r A - 1) \frac{1}{\lambda_r} \text{ bits-m/s},$$

$$T_p(\lambda_r) = \beta \lambda_p A \frac{\delta}{\sqrt{\lambda_r}} \text{ bits-m/s},$$

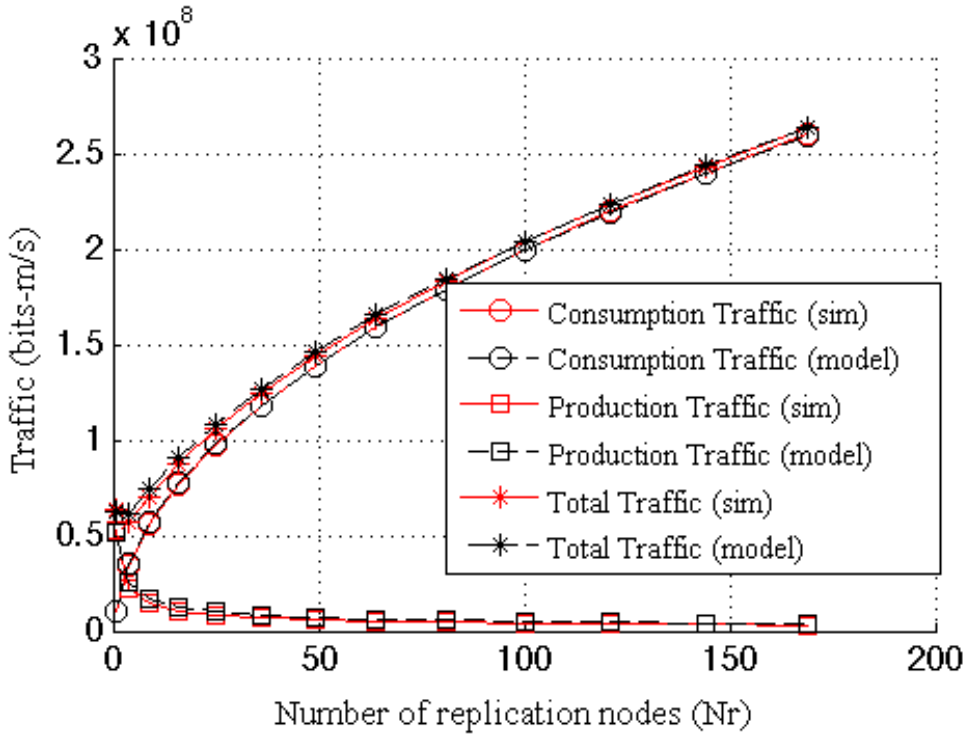


Figure 5.5: *Overall network traffic generated by QAR using different number of replicas for the case when Production-dominates-Consumption (A = 1000 × 1000 m^2, N = 5000 nodes, α = 200 bits/query, β = 100 bits/event, N_p = 1000 producers; N_c = 100 consumers, Iterations = 50).*

$$N_r^* = \lambda_r^* A = \delta \frac{\beta \lambda_p}{\alpha \lambda_c} - (1 - \delta)$$

This model ($\lambda_p > \lambda_c$) is symmetric to the previous one ($\lambda_c > \lambda_p$). However in this case the replication tree (formed by the combing routing mechanism) is traversed twice, by the consumer query and by the replication nodes replies, whereas in the Consumption dominates Production case the production events only traverse the tree once. This is translated that now placing replicas is more expensive than in the previous case, and using similar deployments the Production dominates Consumption scenario leads to a lower optimal number of replication nodes.

We have also run simulations to validate the Production-dominates-Consumption model using a similar set up than the one of Consumption dominates Production. Thus we have deployed N=5000 sensors at random in a network of area A=1000x1000 m^2. We keep $\alpha = 200$ bits/query being the double than $\beta = 100$ bits/event. We also use two different scenarios in order to better validate this model: (i) The first scenario presents N_p=1000 producers and N_c=100 consumers that generate 1 event and 1 query per second respectively. Thus, this scenario presents a λ_p/λ_c ratio equal to 10. (ii) The second scenario differs because the number of producers increases up to N_p=3000. Therefore, the λ_p/λ_c ratio becomes 30. Again, we measure the consumption, production and overall network traffic over 50 different sample scenarios and present average traffic values as results.

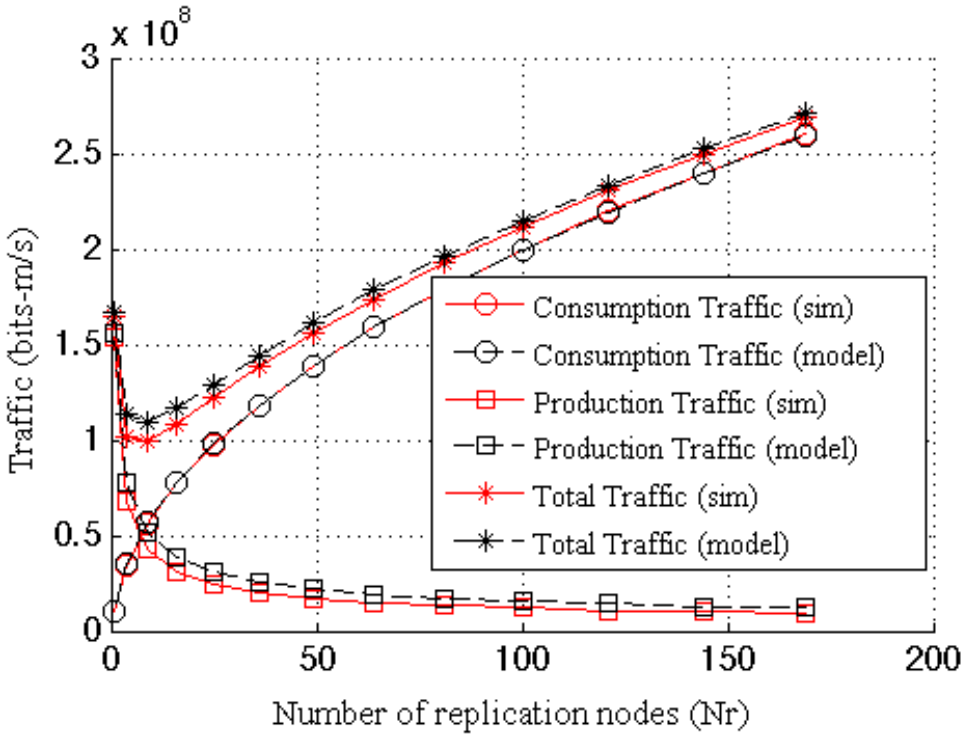


Figure 5.6: *Overall network traffic generated by QAR using different number of replicas for the case when Production-dominates-Consumption (A = 1000 × 1000 m^2, N = 5000, α = 200 bits/query, β = 100 bits/event, N_p = 3000 producers; N_c = 100 consumers, Iterations = 50).*

Figures 5.5 and 5.6 show the model validation via simulation for the cases when the λ_p/λ_c ratio is 10 and 30, respectively (confidence intervals are not shown because they would be undistinguishable). Then, it is clear that our model is very accurate and provides an optimal number of replicas to be used by QAR in order to minimize the overall network traffic. Furthermore, as it was already mentioned in the Consumption-dominates-Production case, the larger the λ_p/λ_c ratio the higher the optimal number of replicas to be placed in the network.

However, the growth of the optimal number of replicas in relation with the ratio's increment is lower than in the previous case. Figure 5.7 depicts how the optimal number of replicas increments with the ratio in both Consumption-dominates-Production and Production-dominates-Consumption cases. It must be noted that this analysis is only valid when both cases define the same values of α and β.

5.2 Performance Evaluation

This section first compares the network cost in terms of hops when the most relevant multi-replication solutions, ToW and multi-replication GHT, are compared with QAR. To be fair we need to compare ToW Write-one-Query-all mode and QAR Production-dominates-

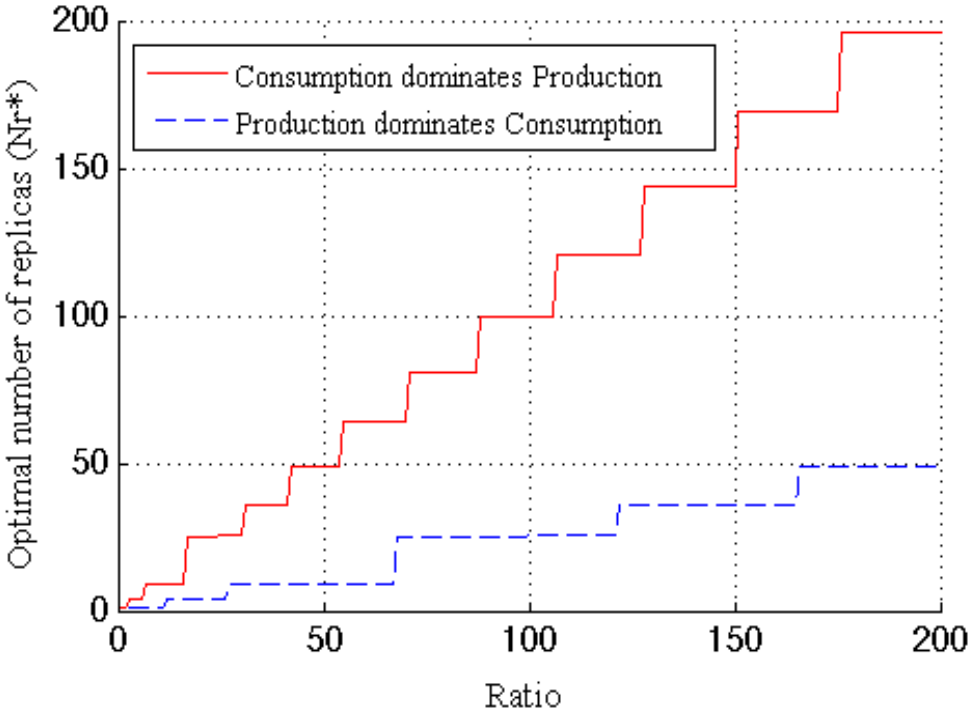


Figure 5.7: *Consumption-dominates-Production vs. Production-dominates-Consumption. Growth of the optimal number of replicas for both cases with respect to the ratio. The ratio refers to λ_c/λ_p for the Consumption-dominates-Production case and λ_p/λ_c for the Production-dominates-Consumption case.*

Consumption mode to the multi-replication GHT functionality. We remember that in GHT with multiple replicas producers push the event data to the closest replica, then consumer queries need to reach all the replicas in order to retrieve the information. In addition, it must be highlighted that GHT with multiple replicas uses the hierarchical routing described in Section 3.6.1, whereas ToW and QAR use the combing routing introduced in Section 3.6.3.

We simulate a sensornet with an area A = 1000 $\times$ 1000 m^2, N = 5000 sensors deployed at random, and a transmission range Tx = 60 m. In order to simulate the traffic load among the 5000 sensors, we randomly select N_c=20 consumers and N_p=2000 producers. Consumers and producers generate traffic in a regular basis, that is, each consumer produces 1 query per unit time and each producer generates 1 data event per unit time. So that, we measure the traffic load in total number of hops in the sensornet per unit time using different number of replicas for each solution (*i.e.*, N_r=1, 4, 16, 64 and 256 for ToW and GHT with multiple replicas, and N_r=1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225 and 256 for QAR that, as it has already been described, has more granularity). The results shown in Figure 5.8 are the average traffic values (in number of hops per unit time) over 50 different scenarios for each number of replicas being evaluated (confidence intervals are not shown because they would be undistinguishable).

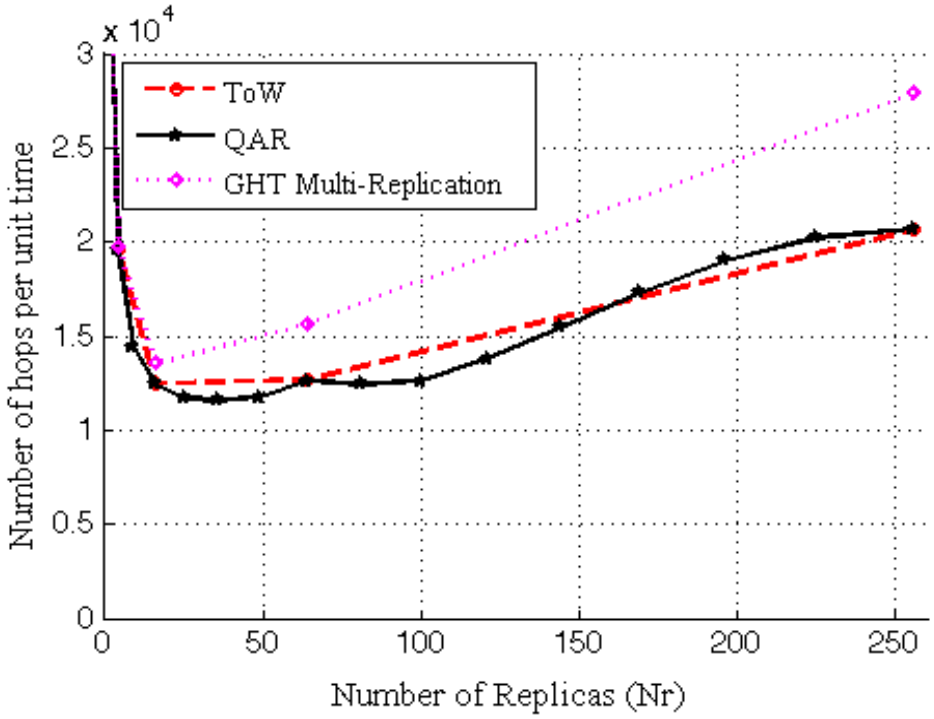


Figure 5.8: *ToW vs Multi-Replication GHT vs QAR (A = 1000 $\times$ 1000 m^2, N = 5000 nodes, Tx = 60 m, N_c = 20 consumers, N_p = 2000 producers, Iterations = 50)*

The first conclusion is that GHT with multiple replicas is the worst solution. The hierarchical routing when using several replicas incurs in a higher cost than the combing routing. That is why in all the cases GHT with multiple replicas has the highest traffic cost. On the other hand, the figure shows the better granularity of QAR compared to the other solutions. QAR can choose up to 9 different number of replicas between 64 and 256, whereas ToW and GHT with multiple replicas only support these two values. Finally, it can be seen that in case of choosing the number of replicas that minimizes the overall traffic (16 for ToW and 25 for QAR), QAR is the best solution globally.

After demonstrating that QAR and ToW outperforms GHT with multiple replicas, we will analyze and compare the performance for different λ_c/λ_p and λ_p/λ_c ratios using the optimal number of replicas, N_r^*, provided by ToW and QAR analytical models. We compare the traffic load in number of hops for GHT with a single replica, ToW (with $k = 1$) and QAR in the same scenario. We cannot include GHT with multiple replicas in this evaluation because the authors do not propose any model or heuristic mechanism to obtain the optimal number of replicas. An option could be to use the optimal number of replicas of ToW also in GHT with multiple replicas since both solutions follows a 4^d replication approach. However, since it has been already demonstrated that GHT with multiple replicas performs worse than ToW, we will compare our proposal, QAR, only with ToW.

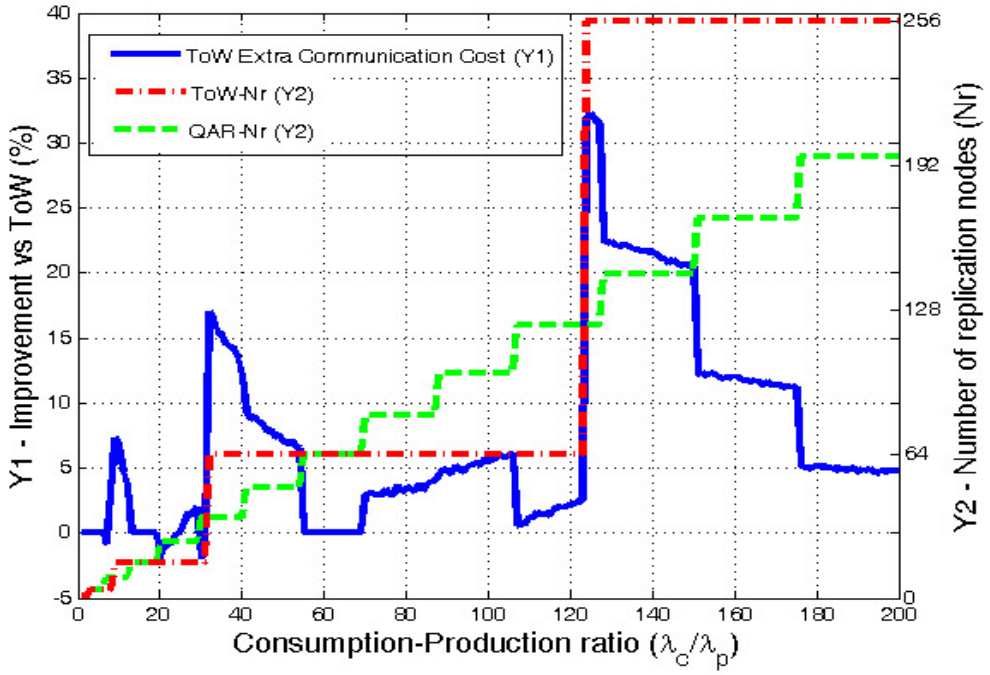


Figure 5.9: *Extra communication cost produced by ToW in front of QAR for the case when Consumption-dominates-Production in Y1 axis. Number of replica used for each approach in Y2 axis (A = 1000 × 1000 m^2, N = 5000 nodes, Tx = 60 m, Iterations = 100).*

5.2.1 Consumption-dominates-Production evaluation ($\lambda_c > \lambda_p$)

We evaluate a scenario with an area A = 1000 x 1000 m^2, where N = 5000 sensors have been deployed at random, with a transmission range Tx = 60 m. The ratio λ_c/λ_p is varied from 1 to 200. For each particular value we provide the average result obtained from 100 simulated scenarios. Finally, we consider the value of α to be twice of β, since each query needs a reply message, while production events just generate a single message. Under these conditions, we measure the extra overall network traffic (in number of hops) generated by ToW in front of QAR.

Figure 5.9 clearly shows that QAR is more adaptive than ToW because the number of replicas increases in smaller steps than ToW. Moreover, QAR is the one that minimizes the communication cost. ToW needs in average 7.56% more extra hops in the communications. If we carefully look at the results, they show that the improvement is specially significant when the ToW model increases the number of replicas. Then, during the cases where ToW uses d = 2 (16 replicas) the results show a maximum extra communication cost of 7.28% compared with QAR. In the cases where ToW chooses d = 3 (64 replicas), QAR improves ToW in average a 5%, with a maximum of a 17%. Finally, the case when ToW selects d = 4 (256 replicas) demonstrates that applying ToW with high ratios is not good enough due to its low adaptivity. In this case the maximum extra cost is 32.3% being the average gain of QAR a remarkable 13.5%.

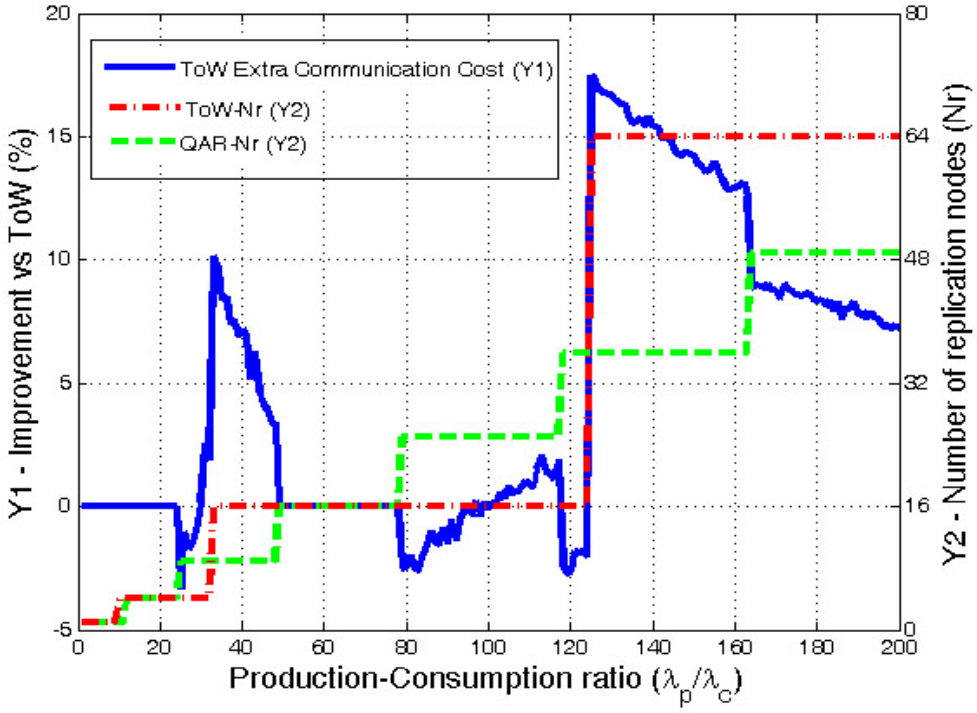


Figure 5.10: *Extra communication cost produced by ToW in front of QAR for the case when Production-dominates-Consumption in Y1 axis. Number of replicas used for each approach in Y2 axis (A = 1000 × 1000 m^2, N = 5000 nodes, Tx = 60 m, Iterations = 100).*

It must be noted, that in some few cases ToW slightly outperforms QAR, but it never goes further than a 2% improvement. This happens in very few cases where our model, which is a distance-based one, does not always provide the best number of replicas in terms of hops, but the in terms of distance, which leads to a similar result in number of hops, but it is not always optimal.

As a side note, the original DCS proposal having a single replica has an average cost 3 times (300%) greater than our solution. This further demonstrates that using multi-replication for Data Centric Storage is a must.

5.2.2 Production-dominates-Consumption evaluation ($\lambda_p > \lambda_c$)

We use a scenario with the same parameters than in the previous case. Again, the inverse ratio λ_p/λ_c changes from 1 to 200.

First of all, it must be noted that in this case increasing the number of replicas requires a higher λ_p/λ_c ratio than the λ_c/λ_p ratio required in the previous section. This is why in this case the ToW solution only goes up to 64 replicas (d = 3) while QAR reaches 49. Figure 5.10 shows the extra communication traffic that ToW generates when compared with QAR.

In this case the average extra cost generated by ToW is a 4.8%. Analyzing the results for the different network depth values chosen by ToW, when d = 1 the improvement is not very significant. However, when ToW moves to d = 2, it produces a maximum extra cost of 10.15%. In the case of d = 3, QAR improves ToW in a 11.45% in average, reaching a maximum of 17.5%.

Again, there are some few cases when ToW outperforms QAR, but the difference is always lower than a 3%. Again, the mismatch between the QAR distance model and the evaluation in number of hops is the reason of this behavior.

When comparing our solution versus the original DCS with a single home node, the average extra communication cost produced by GHT is 145%.

We can finally conclude that the quadratic evolution of QAR is much more adaptive than previous solutions in the literature that follows a 4^d grid structure replication, like ToW or the original multi-replication GHT. QAR improves the average network overall communication traffic and reaches peak traffic reduction of 32% in front of ToW, which has been demonstrated to be less adaptive to network conditions. In addition, QAR does not add any additional complexity to the replication mechanisms proposed in the literature.

Chapter 6

Random Multi-Replication DCS

Summarizing the state of the art in Multi-Replication DCS systems, we have already analysed the most relevant works in the literature. In addition, we have presented our Quadratic Adaptive Replication (QAR) [CUnRL10] proposal, which has been shown to be the most efficient grid multi-replication proposal in terms of overall network traffic reduction. However, most of these schemas rely on structured replication that adds some complexity and is only applicable to sensornets with regular shapes. Moreover, they are not adaptive enough, even in the QAR case. For instance, QAR has to decide to use either 25 or 36 replicas when its analytical model for the optimal number of replicas concludes that 29 replicas should be placed in the field. In this case, ToW [JH08] or GHT with multiple replicas [RKS+03] are even worse since they would need to choose between 16 or 64. Scaling-Laws with the proposed uniform replication could be more accurate when choosing the optimal number of replicas, however as we see later in this chapter following the optimal number of replicas provided by Scaling-Laws [AK06] solution leads to a very bad performance when compared with Random Multi-Replication, ToW and QAR.

Therefore in front of all these solutions we wanted to analyse and evaluate what would occur if the replication nodes positions were just randomly placed inside the network [CUndV10].

Assuming that consumption queries and production events are roughly homogeneously distributed within the network, it seems that those solutions proposing structured-replication will better cover the network area than a random distribution of replication nodes. By using random placement one could think that it is likely that two replication nodes could be close together, or the position of some replica could be close to the network border, thus offering service to very few nodes in the network, etc. However, as it is demonstrated in this chapter, Random Multi-Replication leads to an effective traffic reduction in front of all the previous approaches presented in the literature

In this chapter, we first overview the proposed Random Multi-Replication DCS system. Next, we present an analytical model to easily compute the optimal number of randomly placed replication nodes in order to minimize the overall traffic. It is different than previous

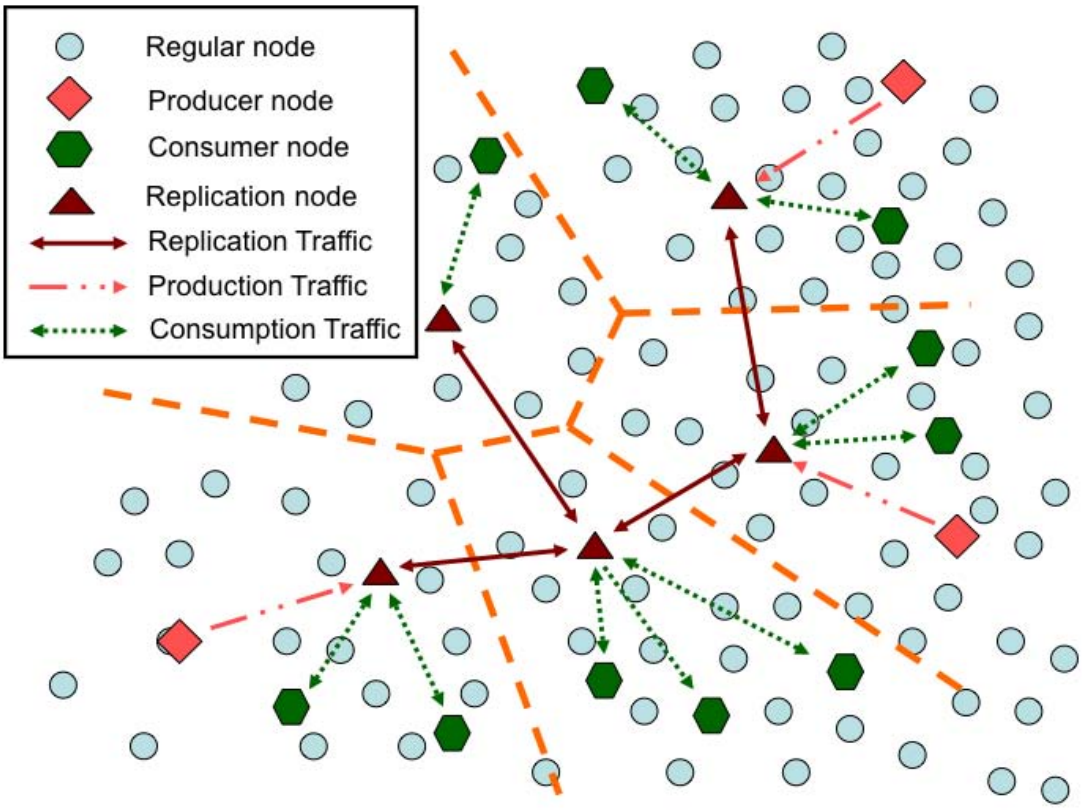


Figure 6.1: *Example of Data-Centric Storage with five Randomly-placed Replicas.*

analytical models, since in this case the replication tree is no longer a grid (like ToW or QAR). Thus we had to think in a simple and accurate model that captures the length of that irregular replication tree. It is very important to find a simple model since it has to be evaluated by the sensors themselves in order to decide what is the optimal number of replicas in each moment. In addition, that formula should rely on measurable network parameters so that the model could be used in practice. For instance, ToW model relies on a k parameter that defines the average number of sensors that detect the same event in the network. That parameter is very hard to measure, therefore it makes the ToW model difficult to apply in practice. However, the QAR model only depends on the consumption and production intensities that is something easily measurable by the replication nodes, therefore it is a good model to be applied in practice. Hence, the model for Random Multi-Replication, as we did for QAR, must be simple in order to be computed by real sensor motes.

Furthermore, we also model the storage limitation of a multi-replication DCS using random replica allocation. With this model we are able to define what is the maximum number of replicas that can be allocated per application in order to avoid storage overflow with a given probability. This is, to the best of our knowledge, the first study with this scope in the field of multi-replication DCS. Therefore, we present the first results on that direction.

Finally, we compare Random Multi-Replication quantitatively and qualitatively with all

the previously presented approaches, showing that Random Multi-Replication is the best mechanism to be applied in multi-replication DCS networks.

6.1 System Overview

In this section we introduce the basic functionalities required for Random Multi-Replication, focusing on the case where Consumption dominates Production traffic, see e.g., Fig. 6.1. Let us consider that the application's name is *APP*, and based on the current ratio of consumption to production demand (λ_c/λ_p) and the network dimensions, the optimal number of replication nodes is N_r. To simplify the description, we start assuming that this info is known by every application's node.

6.1.1 Producers and consumers functionality

Suppose a producer or consumer node generates an event or queries some data (respectively) related to *APP*. It then determines the closest replication point by computing the Euclidean distance between its spatial location and that of all replication points obtained from the hash operation: $hash(APP \oplus i), \quad \forall i \in [1, N_r]$. Once the producer/consumer node determines the closest rendezvous point, it forwards a message/query to that location, i.e., to the node n_1 closest to its location. In case of consumption, the rendezvous node just responds with the suitable data to the query. This replication location will be used for some time, so the producers and consumers may cache the replication point coordinates avoiding recomputation for every single message[1].

6.1.2 Creating a tree to replicate data over replication nodes.

The next step for producer events is creating a *minimum spanning tree* rooted at n_1 over which data replication takes place. Each replication node needs to determine the set of nodes (if any) to which it should forward new data. Since all replication nodes know the hashed locations, we will consider, without loss of generality, the construction of the replication tree from the point of view of a given rendezvous node, e.g., the root node. The root node, n_1, manages three sets of replication nodes:

- $\mathcal{C}$: the set of replicas already covered by the replication tree, where initially $\mathcal{C} = \{n_1\}$.
- $\mathcal{R}$: the set of replicas to be reached, which initially contains all the replication nodes except the root node: $\mathcal{R} = \{n_2, n_3, ..., N_r\}$.

[1]In this section we will equivocate the replicationx nodes with the associated hashed locations. There are several ways of finding the closest node to a given location, but they are energy consuming. Thus, in case of using geographic locations we consider that the first time a replication node is contacted by other node, it notifies to that node its actual location, so from that moment the contacting node can directly communicate with the replication node avoiding the energy expenditures of finding the closest node to a given location for each message.

Algorithm 1 Replication Tree construction algorithm run in a replication node n_i to know what are the replicas to whom it has to forward production events.

```
/* Initial sets from n_0 /*
myself = n_i;
C = {n_1}
R = {n_2, n_3, ...., n_r}.
F = ∅
/* Algorithm */
while R! = ∅ do
  for i = 1 to C.length do
    min_distance = ∞;
    initial_node = C[i];
    for j = 1 to R.length do
      dest_node = R[j];
      aux_distance = distance(initial_node, dest_node);
      if aux_distance < min_distance then
        min_distance = aux_distance;
        initial_node_selected = initial_node;
        dest_node_selected = dest_node;
      end if
    end for
  end for
  C.add( dest_node_selected);
  R.remove(dest_node_selected);
  if initial_node_selected == myself then
    F.add(dest_node_selected);
  end if
end while
```

- $\mathcal{F}$: the set of replicas to which the current replication node should forward the event, which is initially empty: $\mathcal{F} = \emptyset$

The root node computes which replication node in $\mathcal{R}$ is the closest one to n_1. Suppose it is n_2, then n_2 is removed from $\mathcal{R}$ and included in both $\mathcal{C}$ and $\mathcal{F}$., i.e., $\mathcal{C} = \{n_1, n_2\}$, $\mathcal{R} = \{n_3, ..., n_r\}$, and $\mathcal{F} = \{n_2\}$. Next, it computes which replication node in $\mathcal{R}$ is the closest one to anyone in $\mathcal{C}$. If the closest distance is between the root n_1 and n_3, then n_3 is removed from $\mathcal{R}$ and included in $\mathcal{C}$ and $\mathcal{F}$. However, if the closest distance is the one between n_2 and n_3, n_3 is also removed from $\mathcal{R}$, but only included in $\mathcal{C}$. The process is repeated until $\mathcal{R}$ is empty, at which point $\mathcal{F}$ contains all the forwarding rendezvous nodes of n_1. This process is fully specified by Algorithm 1.

Assuming that each node knows who the root is, each replica can similarly compute its associated forwarding sets $\mathcal{F}$. Note that if the above distributed mechanism is used, there will be one replication tree per replication node, which serves as its root. The routing table of a replication node associated with a given application would have one entry per replication node acting as root node for production events, with the associated forwarding nodes $\mathcal{F}$ obtained after running the algorithm. Alternatively only one of these trees could be shared among all replicas.

6.2 System Model

In this section we propose a simple stochastic geometric model for the network that permits the optimization of large scale system's parameters, i.e., intensity of replication nodes. The approach follows the seminal work of [cBZ96, cBKLZ97] and another work in applying this methodology to ad hoc wireless networks, e.g., [BdVS04, BdV07].

The locations of nodes in the Wireless Sensor and Actor Network (WSAN) are assumed to be fixed, and modeled by a homogeneous spatial Poisson Point Process Π_n, i.e., a 'random' set of points on the plane, with intensity λ_n locations per unit area [SKM95]. A fraction of those nodes are randomly, independently sampled to serve as replication nodes. Under these conditions the replication nodes also follow a homogeneous spatial Poisson Point Process Π_r, with intensity $\lambda_r < \lambda_n$. Production and consumption events, generated by some networks nodes, are in turn modeled by independent homogeneous spatio-temporal Poisson Point Processes Π_p and Π_c each with intensities λ_p and λ_c events per unit time and unit area respectively. To avoid unnecessary complications, we shall assume that spatial process Π_r and spatial temporal point processes Π_p and Π_c are mutually independent. Note that this is not the case in reality, since they are connected through the locations of the nodes Π_n in the network. However if λ_n is high enough, the impact on our model is minimal– we shall verify this via simulation and with a small prototype testbed in the sequel. Although the model corresponds to one on an infinite plane, we shall restrict attention to a fixed region $\mathcal{A} \subset \mathbb{R}^2$ modeled as a convex set with area $A = |\mathcal{A}|$, and optimize operation on $\mathcal{A}$ roughly ignoring edge effects. Therefore, on average there are $N_r = \lambda_r A$ replication nodes in $\mathcal{A}$.

6.2.1 Evaluating overall network traffic and energy costs.

Let us first consider the overall network traffic generated by consumption and production events inside the network. The overall metric here is the total traffic load, measured in bits-meter/second that need to be supported by the network, i.e. in region $\mathcal{A}$. As we already explained for QAR, in an ad hoc wireless network traffic load can not simply be measured in terms of bits/sec, but must also account for the distance packets must travel. Therefore, measuring network load in terms of bits-m/s captures the amount of traffic and the distance that must be traveled. In turn, we assume the power expenditures for transporting traffic to be roughly proportional to the overall network traffic.

Case 1: Consumption-dominates-Production model ($\lambda_c > \lambda_p$)

We assume consumers retrieve data from the closest replication node. Thus consumption events can be partitioned based on the Voronoi tessellation [cBKLZ97] induced by the replication nodes. The average size of such cells is $1/\lambda_r$, the mean number of consumption events in such a region per unit time is λ_c/λ_r. Meanwhile, the typical distance from a consumer to its nearest replication node can be shown to be $\frac{1}{2\sqrt{\lambda_r}}$ [cBZ96]. Thus the total consumption traffic, $T_c(\lambda_r)$, for the region $\mathcal{A}$ is proportional to the number of replication nodes $\lambda_r A$, times the number of consumers per replication node cell λ_c/λ_r, further multiplied by the mean distance between consumers and replication nodes $\frac{1}{2\sqrt{\lambda_r}}$, i.e.,

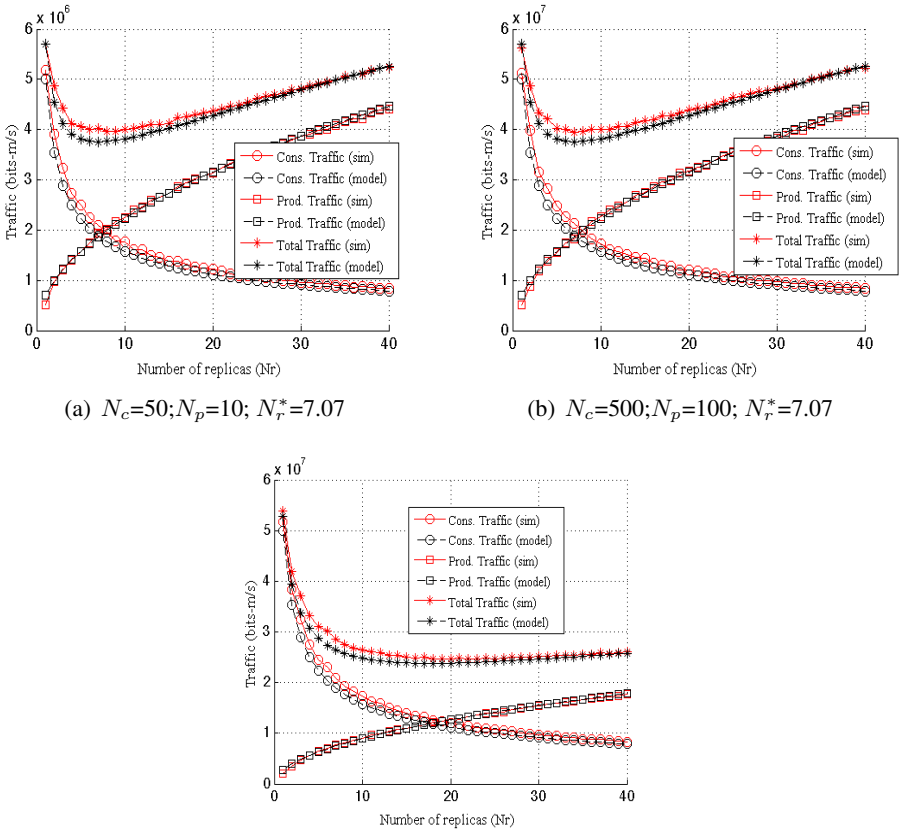
$$T_c(\lambda_r) = \alpha\lambda_r A\frac{\lambda_c}{\lambda_r}\frac{1}{2\sqrt{\lambda_r}} = \alpha A\frac{\lambda_c}{2\sqrt{\lambda_r}} \text{ bits-m/s},$$

where α is a proportionality constant corresponding to the average number of bits per consumption query that are exchanged between the consumer and its nearest replication node.

Next, we consider the replication cost when new data is produced. Again new data is produced on our network at a rate $\lambda_p A$ events per unit time. We shall assume that data associated with each new event is distributed to the replication points in the network along a *radial spanning tree* [cBB07] which includes all the replication nodes. The total length per unit area for radial spanning trees over a homogeneous Poisson Point Process can be computed and is indeed very close to that of a minimum cost spanning tree. Moreover, radial spanning trees are fairly easy to construct so this appears to be both a practical and mathematically tractable model for distribution. In particular for a large disc of radius x, the total length for a radial spanning tree centered at the origin grows as $\frac{\pi x^2\sqrt{\lambda_r}}{\sqrt{2}}$, so the average length of the tree per unit area is given by $\sqrt{\lambda_r/2}$ [cBB07]. The total production traffic generated, $T_p(\lambda_r)$, is thus given by β bits per event, times the rate of production events $\lambda_p A$ in the network, times the length of the associated radial spanning tree:

$$T_p(\lambda_r) = \beta\lambda_p A\sqrt{\frac{\lambda_r}{2}}A = \beta A^2\lambda_p\sqrt{\frac{\lambda_r}{2}} \text{ bits-m/s}$$

Note that we have assumed for simplicity that the radial spanning tree is rooted at the location where the event is produced. Alternatively one could assume that the new event is first



(a) N_c=50;N_p=10; N_r^*=7.07

(b) N_c=500;N_p=100; N_r^*=7.07

(c) N_c=500;N_p=40; N_r^*=17.67

Figure 6.2: *Consumption, Production and Total traffic generated by using different number of Random Replication nodes for the case when Consumption-dominates-Production (A=1000x1000 m^2, N=5000 sensors, α=200 bits/query, β=100 bits/event).*

transported to the nearest replication node that then employs a radial spanning tree to reach the remaining replicas. The replication cost in this second case has a similar scaling.

The total network traffic, $T(\lambda_r)$, is thus given by:

$$T(\lambda_r) = T_c(\lambda_r) + T_p(\lambda_r) = \alpha A \lambda_c \frac{1}{2\sqrt{\lambda_r}} + \beta A^2 \lambda_p \sqrt{\frac{\lambda_r}{2}} \text{ bits-m/s}$$

We can optimize this to obtain an optimal spatial intensity for replicas λ_r^* given by:

$$\lambda_r^* = \frac{\alpha \lambda_c}{\sqrt{2}\beta A \lambda_p} \text{ replicas/m}^2$$

and the associated minimum overall network traffic is given by:

$$T(\lambda_r^*) = 2^{1/4}\sqrt{A}\sqrt{(\alpha\lambda_c A)(\beta\lambda_p A)} \text{ bits-m/s}$$

Remark 1. ***Scaling characteristics.*** *Roughly speaking, the optimal average number of replicas for the network covering an area A is given by:*

$$N_r^* = \lambda_r^* A = \frac{\alpha \lambda_c}{\sqrt{2}\beta \lambda_p}. \tag{6.1}$$

Surprisingly, this only depends on the ratio of the intensity of consumption to production. Thus if one were to double the intensity of consumption and production for a fixed area, the same number of replicas would be optimal. If however one stretches the area by a factor of two, this would decrease the intensity of production and consumption by 2, maintaining the same ratio, yet the optimal intensity λ_r^* per unit area would also have to decrease by a factor of 2. Furthermore we note that the overall network load, in bits-m/sec scales as $\sqrt{A}$ times the geometric mean *of the total rate of consumption, $\alpha\lambda_c A$ in bits/sec and the rate of production $\beta\lambda_p A$ in bits/sec. This gives a sense of the growth of overall traffic with network size.*

Remark 2. ***Is broadcasting preferable?*** *Note that, if the intensity of consumers is very high, producers could be tempted to simply broadcast new data to all nodes in the network. The overall traffic, T_b associated with broadcasting to all nodes in the network, $N = \lambda_n A$, can be modelled based on the length of the radial spanning tree reaching all nodes (some of which would be consumers):*

$$T_b = \beta A^2 \lambda_p \sqrt{\frac{\lambda_n}{2}} \text{ bits-m/sec}.$$

Under this simple model, broadcasting would be favorable only if:

$$T(\lambda_r^*) > \beta A^2 \lambda_p \sqrt{\frac{\lambda_n}{2}}$$

which is equivalent $\lambda_n < 4\lambda_r^*$. Thus, unless the optimal number of replicas is very high (i.e. on the order of $1/4$ of the total number of nodes in the network), brute force broadcasting is not likely to be efficient. Note that this does not account for the so called wireless "broadcast advantage" whereby a node can send data to multiple nodes in a single transmission, and perhaps more efficient methods of realizing and modeling broadcasting. Still the key here is that the optimal number of replication points would have to be very high indeed if broadcasting were to become more efficient than this Random Multi-Replication Data Centric Storage approach.

In order to validate this model we have first simulated random realizations of the network and obtained the consumption (T_c), production (T_p) and total network cost (T) for different numbers of replicas. Unless otherwise stated, all results correspond to at least 50 simulations of different network realizations where $N = 5000$ nodes are randomly placed in a 1000×1000 region. We set β=100 bits/event, assuming that producers periodically send the information to the closest replica without any acknowledgement. We set α=200 bits/query since we assume that a consumer first sends a query message to its closest replica and then receives a reply from it. We show 90% confidence intervals on all graphs unless they are so small that they cannot be distinguished.

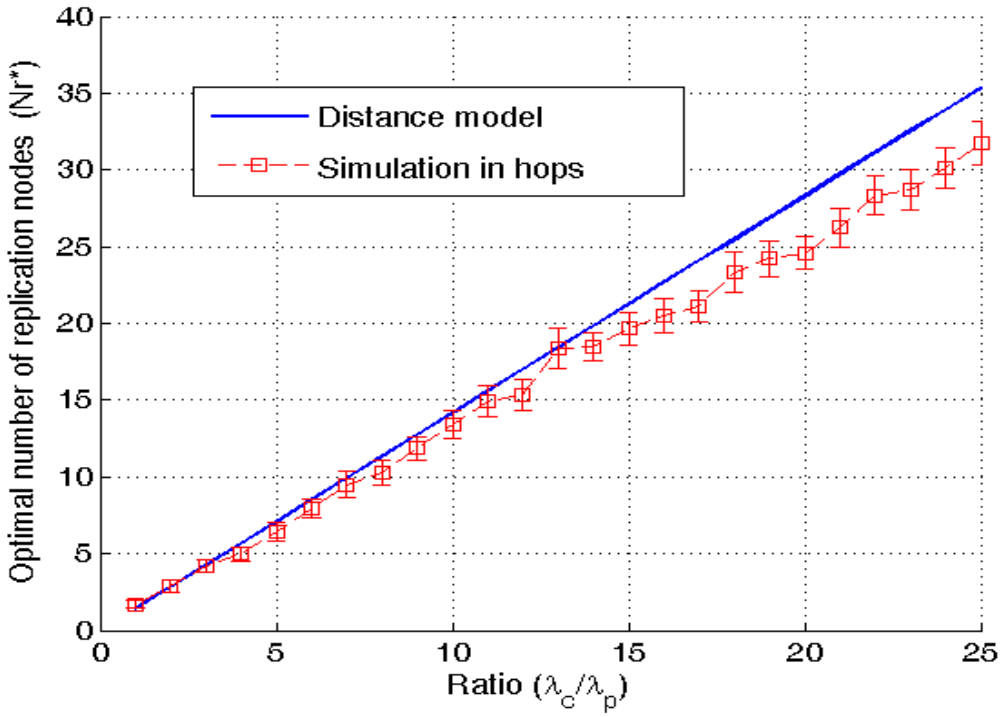


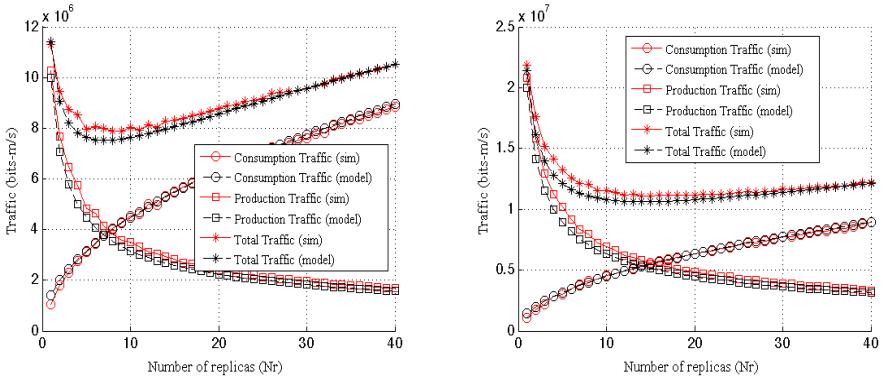
Figure 6.3: *Optimal number of Random Replicas that minimizes the overall number of messages (A=1000x1000 m^2, N=5000 sensors, Tx=50 m)*

Figure 6.2 exhibits the overall consumption, production and total traffic measured in bits-m/s obtained by the model and by simulation for three different (N_c, N_p) pairs: (50,10), (500,10) and (500,40) , which generates the next (λ_c,λ_p) pairs: ($50 * 10^{-6}$,$10 * 10^{-6}$), ($500*10^{-6}$,$100*10^{-6}$) and ($500*10^{-6}$,$40*10^{-6}$) $\frac{events}{s*m^2}$. The number of replicas employed varies from 1 to 40. Thus, the optimal average number of replicas for these cases is 7.07, 7.07 and 17.67 respectively. Figures 6.2(a) and 6.2(b) illustrate the scaling properties of the framework versus the ratio of consumer to producer intensities. Note that both scenarios have exactly the same optimal number of replicas, even though the latter's application generates ten times more production and consumption events than the former. It is worth noting that for applications with a high λ_c/λ_p ratio (see Figure 6.2(c)), there are several values around the optimal number of replicas that could be also employed, because they generate a similar overall traffic.

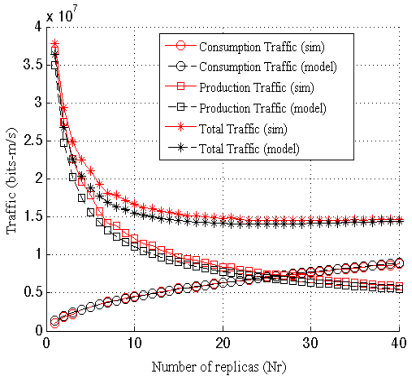
It must be highlighted that this simple model employs distance traffic metrics assuming routes follow straight lines. However, WSANs, which are the focus of this Thesis, are multi-hop networks where routes unlikely follow straight paths. To that end we have verified that for networks that have a sufficiently high density of nodes, the optimal number of replicas obtained by our idealized model reflects the actual optimal number of replicas on a given network. For this purpose we have simulated a sensor network employing greedy forwarding [KK00] and a transmission range $Tx = 50$ m. We considered a setup where the ratio λ_c/λ_p varied from 1 to 25. Figure 6.3 shows the number of replicas that minimizes the overall

simulated traffic based on the actual number of hops of all messages, compared to the optimal number of replicas suggested by our model. As it can be seen, when there is a low number of replicas, the model and the simulations are a good match. A small discrepancy occurs for high λ_c/λ_p ratios. However, as mentioned earlier, in the case this ratio is high, the overall cost is not very sensitive to the precise optimal value for the number of replicas.

Case 2(a): Production-dominates-Consumption ($\lambda_c < \lambda_p$) with Data Aggregation



(a) N_c=10;N_p=200;N_r^*=7.07 (b) N_c=10;N_p=400;N_r^*=14.14



(c) N_c=10;N_p=700;N_r^*=24.64

Figure 6.4: *Consumption, Production and Total traffic generated by using different number of Random Replication nodes for the case when Production-dominates-Consumption and the query replies are aggregated in the replication tree (A=1000x1000 m^2, N=5000 nodes, α=200 bits/query, β=100 bits/event).*

If the intensity of consumption is low relative to that of production it may be preferable not to copy data across all replication nodes. Instead producers can store data solely at the closest replication node. Subsequently consumers should contact *all* replication points to gather the information. This could be done in several ways.

First, a symmetric model to that presented in the case of Consumption-dominates-Production could be also proposed. However, that model would assume that both, queries

and replies, are sent through the replication tree once per branch, as it is done by ToW and QAR. This can only be achieved if replies are aggregated and such aggregation has implications that are out of the scope of this Thesis. In case that aggregation happens, we present a symmetric model to the one in consumption dominates production case. Therefore, the production and consumption traffic models are:

$$T_p(\lambda_r) = \beta A \frac{\lambda_p}{2\sqrt{\lambda_r}} \text{ bits-m/s}$$

$$T_c(\lambda_r) = \alpha A^2 \lambda_c \sqrt{\frac{\lambda_r}{2}} \text{ bits-m/s}$$

The overall network traffic is modelled as:

$$T(\lambda_r) = \alpha A^2 \lambda_c \sqrt{\frac{\lambda_r}{2}} + \beta A \lambda_p \frac{1}{2\sqrt{\lambda_r}} \text{ bits-m/s}$$

This can again be optimized to obtain the optimal spatial intensity for replicas λ_r^* given by:

$$\lambda_r^* = \frac{\beta \lambda_p}{\sqrt{2}\alpha A \lambda_c} \text{ replicas/}m^2$$

and the associated minimum overall network traffic is similar in form to Case 1:

$$T(\lambda_r^*) = 2^{1/4}\sqrt{A}\sqrt{(\alpha\lambda_c A)(\beta\lambda_p A)} \text{ bits-m/s}$$

Case 2(b): Production-dominates-Consumption ($\lambda_c < \lambda_p$) without Data Aggregation

However, many times the aggregation could be a really complex issue since it requires state and additional processing inside the network. In addition, many applications do not need data aggregation but just all the produced data. For all those cases, we consider a very simple model where consumers contact all the replication nodes directly.

In this case the overall production traffic is:

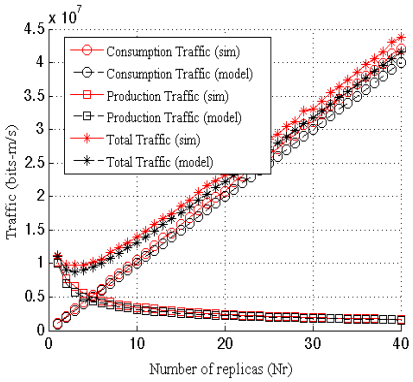
$$T_p(\lambda_r) = \beta A \frac{\lambda_p}{2\sqrt{\lambda_r}} \text{ bits-m/s}$$

The consumption cost can be modeled using the average distance between any two nodes of the network $\sqrt{A}/2$, as the distance from a consumer to each replica, times the number of consumers ($\lambda_c A$) and replicas ($\lambda_r A$). Thus the overall consumption traffic is given by:

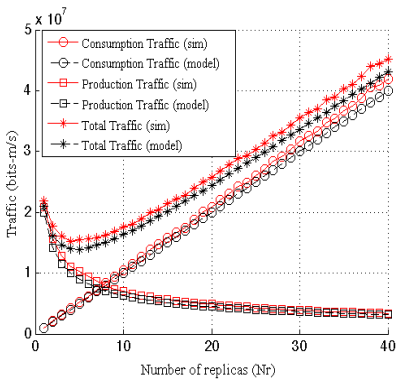
$$T_c(\lambda_r) = \alpha(\lambda_c A)(\lambda_r A)\frac{\sqrt{A}}{2} \text{ bits-m/s}$$

The total network traffic is then given by:

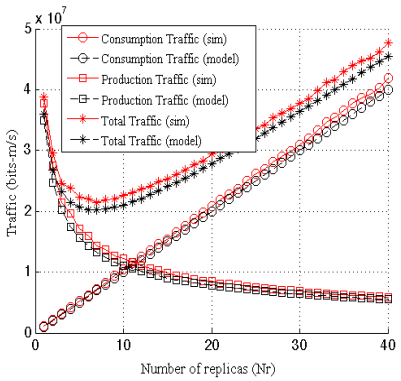
$$T(\lambda_r) = \alpha\lambda_c\lambda_r A^2 \frac{\sqrt{A}}{2} + \beta A \frac{\lambda_p}{2\sqrt{\lambda_r}} \text{ bits-m/s}$$



(a) N_c=10;N_p=200;N_r^*=2.92



(b) N_c=10;N_p=40;N_r^*=4.64



(c) N_c=10;N_p=700;N_r^*=6.74

Figure 6.5: *Consumption, production and overall traffic generated by using different number of replication nodes for the case when Production-dominates-Consumption and consumers directly query all replication nodes without using a replication tree(A=1000x1000 m^2, N=5000 nodes, α=200 bits, β=100 bits).*

One can again find the optimal replication λ_r^* for this case, which is given by:

$$\lambda_r^* = \frac{1}{A}\left(\frac{\beta\lambda_p}{2\alpha\lambda_c}\right)^{2/3} \text{ replicas}/m^2$$

The associated minimum overall network cost is:

$$T(\lambda_r^*) = (\beta\lambda_p)^{2/3}\,(2\alpha\lambda_c)^{1/3}\,\frac{3A\sqrt{A}}{4} \text{ bits-m/s.}$$

Note that in this regime the optimal intensity for replicas is a more 'complex' function, i.e., cubic of the ratio of production to consumption intensities, yet, in principle, still easily computable by sensors in real time.

Both models have been validated via simulation. We have used a scenario of area A=1000x1000 m^2 where N=5000 nodes where randomly deployed. We used a factor α=200 bits/query and β=100 bits/event.

Figure 6.4 exhibits the overall consumption, production and total traffic measured in bits-m/s obtained by the model and by simulation for the case when aggregation can be used. Then, three different (N_c, N_p) pairs have been evaluated: (10,200), (10, 400) and (10, 700). They generate the next (λ_c,λ_p) pairs: ($10*10^{-6}$,$200*10^{-6}$), ($10*10^{-6}$,$400*10^{-6}$) and ($10 * 10^{-6}$,$700 * 10^{-6}$) $\frac{events}{s*m^2}$. The number of replicas employed varies from 1 to 40. Thus, the optimal average number of replicas for these cases is 7.07, 14.14 and 24.64 respectively. The model is very accurate to the results obtained via simulation as it was expected since this model is a symmetric one to the Consumption-dominates-Production case.

In addition, we used the same (λ_c,λ_p) pairs to evaluate the Production-dominates-Consumption model when consumers use unicast routing to access replication nodes. Figure, 6.5 shows that again the proposed model is very accurate. As it was expected, when the replication tree can not be used, the traffic grows since the routing without aggregation is less efficient. Therefore, placing new replicas is more expensive, that is why the optimal number of replicas for the three (λ_c,λ_p) pairs are now 2.92, 4,64 and 6.74 respectively. These numbers are much lower than the optimal number of replicas when the replication tree is employed in both directions

We can conclude that both models are very accurate and both make sense in practice, since depending on the application, the utilization of the replication/aggregation tree will be feasible or not. If query replies cannot be aggregated, using the replication tree for route individual replies is highly inefficient, since the best option is using unicast routing.

6.2.2 Evaluating storage limits

If multiple applications share the same network storage resources, the storage capacity of sensors, say b bits, may limit the amount of replication. To better understand this, consider a network where m homogeneous applications, i.e., with the *same* consumption and production intensity and data storage requirements d, share a network with an intensity of λ_n nodes/unit area in region $\mathcal{A}$.

To model memory utilization in replication nodes, suppose a given application selects the nodes to serve as replication nodes as follows. It generates random spatial locations Π_r with intensity λ_r on the plane, and then network nodes that are the closest ones to these locations are chosen as replication nodes[2]. Note that if several points in Π_r are close to the same node, then that node is used only once. Let V be a random variable denoting the area of the Voronoi cell of a typical network node. If at least one point in Π_r is in the Voronoi cell of such node, it is selected as a replica. The probability that the region with area V contains no point from the process a Π_r locations, is given by its void probability $p(V) = e^{-\lambda_r V}$ [SKM95]. So the average probability a typical node is chosen by an application using a intensity λ_r for choosing replication nodes is given by:

$$1 - E[p(V)] \quad = \quad 1 - E[e^{-\lambda_r V}] \approx \lambda_r E[V] - \frac{\lambda_r^2}{2} E[V^2] = \frac{\lambda_r}{\lambda_n} - 0.62\frac{\lambda_r^2}{\lambda_n^2}$$

[2]Note that nodes cannot be just selected randomly, since the Voronoi cells of randomly deployed nodes do not have the same size, and thus the probability of selecting each cell is different

where we have used the fact that $E[V] = \frac{1}{\lambda_n}$ and also that $\sqrt{\text{Var}(V)} = E[V](0.52)$ [Mol94].

Let X_i be a Bernoulli random variable which is 1 if application i uses the node as a replication site, and zero otherwise, i.e.,

$$P(X_i = 1) = 1 - E[p(V)] \text{ and } P(X_i = 0) = E[p(V)]$$

Suppose a given node has enough storage for b/d application's data, then the probability that it is overloaded is given by:

$$P(\sum_{i=1}^{m} X_i > b/d)$$

Note that X_i are not independent, because if a cell has a larger area, they are more likely to be 1. In other words they are only conditionally independent given the area of the cell. To estimate the overload probability, we shall still approximate the summation as a Gaussian random variable, i.e., $\sum_{i=1}^{m} X_i \sim N(\mu, \sigma^2)$ where μ and σ^2 correspond to the mean and variance of the sum. In particular, as shown above:

$$\mu = E[\sum_{i=1}^{m} X_i] \approx m\frac{\lambda_r}{\lambda_n} - 0.62\frac{\lambda_r^2}{\lambda_n^2}$$

To compute the variance of the sum we can condition on the size of the cell V to obtain:

$$\begin{aligned}
\sigma^2 &= \text{Var}(\sum_{i=1}^{m} X_i) = E[\text{Var}(\sum_{i=1}^{m} X_i|V)] + \text{Var}(E[\sum_{i=1}^{m} X_i|V]) \\
&= E[mp(V)(1-p(V)] + \text{Var}(m(1-p(V))) \\
&= mE[p(V)(1-p(V)] + m^2(E[(1-p(V))^2] - E[1-p(V)]^2)
\end{aligned}$$

Further expanding terms of the previous equation, we obtain:

$$\sigma^2 \approx m\left(\frac{\lambda_r}{\lambda_n} - 1.9\frac{\lambda_r^2}{\lambda_n^2}\right) + m^2\left(0.27\frac{\lambda_r^2}{\lambda_n^2} + 1.27\frac{\lambda_r^3}{\lambda_n^3} - 1.61\frac{\lambda_r^4}{\lambda_n^4}\right)$$

Now given these results we can roughly ensure that the risk of running out of storage space for a typical sensor is less than δ by assuring that:

$$P(\sum_{i=1}^{m} X_i > \frac{b}{d}) \approx Q(\frac{\frac{b}{d} - \mu}{\sigma}) \leq \delta$$

where $Q()$ denotes the complementary distribution function of a standard Gaussian random variable. This in turn gives a requirement that:

$$\frac{b}{d} \geq \mu + t(\delta)\sigma$$

where $t(\delta)$ is such that $Q(t(\delta)) = \delta$

This gives a constraint on the number of homogeneous applications one can support, or the maximum replication rate per application one can allow.

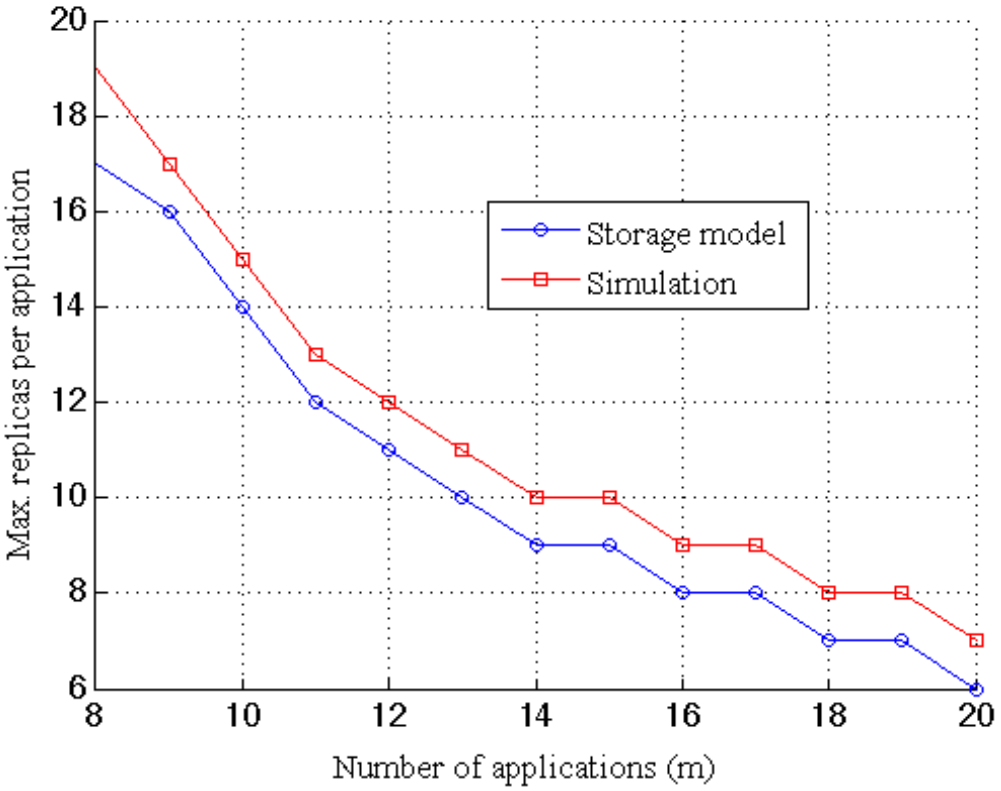


Figure 6.6: *Maximum number of Random Replicas per application to keep the probability of sensor storage overflow below a 10% (A=1000x1000 m^2, N=100 nodes, b/d=3, δ=0.1).*

In order to validate this model, we have simulated a network where the requirements on nodes' storage were fairly high. This is the case where the Gaussian approximation is effective and the model can provide useful results for network designers. Specifically we have simulated a network with $N = 100$ nodes, and varied the number of applications from 8 to 20, and the number of replicas per application from 1 to 20. We have considered the case where $b/d = 3$, i.e., a node can simultaneously support at most 3 applications. For each scenario we have evaluated the maximum number of replicas each application could use while ensuring that a typical node's saturation probability was lower than $\delta = 0.1$ both via simulation and with our analytical model. Figure 6.6 shows that the storage model and the simulation results are very close, showing a difference of just 1 replica in most of the cases. Moreover, it must be noted that the model is conservative, since it provides a lower value than the simulation, which is a desired property for safe network design.

The importance of these results is as follows. When multiple applications share the network infrastructure, our analysis shows that depending on the production and consumption intensity they may choose to use a large number of replicas. However by doing so, it may require replicas to store more data that they are in fact capable. Thus in practice the intensity of replication associated with multiple applications sharing the network may need to be limited, to prevent this overload from happening.

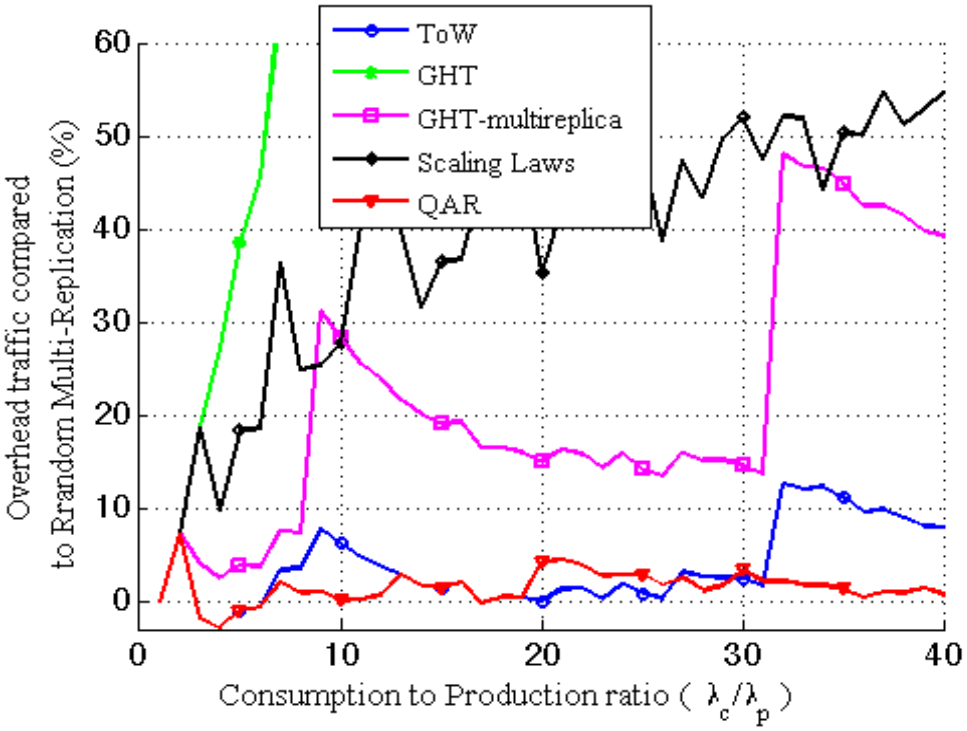


Figure 6.7: *Improvement of overall traffic when using Random Multi-Replication vs. ToW, QAR, Scaling-Laws, GHT and GHT with multiple replicas (A=1000x1000 m^2, N=5000 sensors, Tx=50 m, α=200 bits/query, β=100 bits/event)*

6.3 Performance Evaluation

We have compared our work with those that are similar in spirit: ToW [JH08], QAR [CUnRL10], Scaling Laws [AK06], the original GHT proposal [SRK+03], which uses a single replication node, and GHT with multiple replication nodes [RKY+02, RKS+03]. For the last case, since the authors do not propose any way to obtain the number of replicas to be used, we select the same number used in ToW since both works are grid-based and use the same $N_r = 4^d$ geometric formula for the number of rendezvous nodes.

6.3.1 Quantitative Comparison

In order to compare these approaches we have selected the case when Consumption-dominates-Production and we have run simulations for a large WSAN with the following characteristics: an area $A = 1000x1000\ m^2$, $N = 5000$ sensors, transmission range $Tx = 50$ m and a λ_c/λ_p traffic ratio ranging from 1 to 40. For each λ_c/λ_p ratio we have simulated 50 scenarios to estimate the mean network cost realized by the different replication approaches. In order to get meaningful results, we use the number of hops traversed by all messages as the measure of the overall traffic cost.

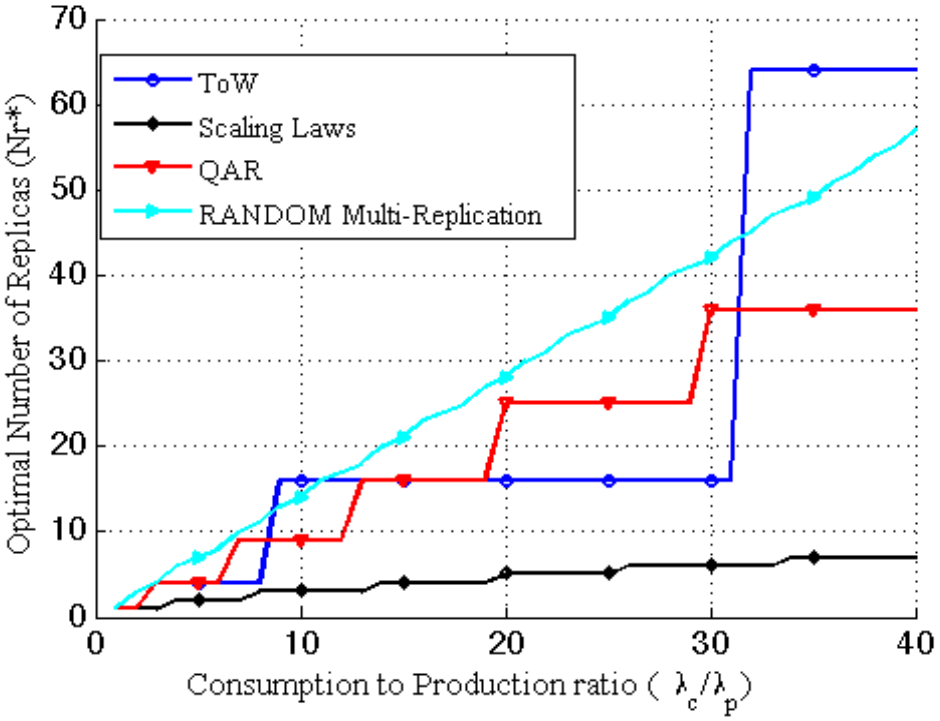


Figure 6.8: *Optimal number of replicas used by the different proposals: ToW, QAR, Scaling-Laws and Random Multi-Replication (A=1000x1000 m^2, N=5000 sensors, Tx=50 m, α=200 bits/query, β=100 bits/event)*

.

Figure 6.7 shows the network traffic improvement achieved using Random Multi-Replication compared to all the other approaches, and Figure 6.8 depicts the number of replicas used by each approach for the same λ_c/λ_p ratios.

Random Multi-Replication reduces the overall traffic by an average of 137% compared to GHT, 39% compared to Scaling Laws, 21% compared to GHT with multiple replication nodes, 4% compared to ToW and 1.5% compared to our previous QAR proposal. Moreover, this improvement reaches peaks around a 50% when compared to Scaling Laws and GHT with multiple replicas, 15% to ToW and 7% to QAR.

The main reason our solution achieves a better performance is that Random Multi-Replication allows a much smoother range over which to adapt its evolution. That is, the optimal number of replicas grows linearly, whereas ToW and GHT with multiple replicas employ a 4^d geometric growth and QAR a quadratic one (see Figure 6.8). For instance, in some cases ToW must choose between 16 or 64 replicas, where none of them is a good fit for the scenario of interest.

Therefore, Random Multi-Replication is demonstrated to be the one minimizing the overall network traffic improving all previous approaches in the literature that use grid-structured or uniform replication. Moreover, processing the random locations for the ren-

dezvous nodes is easier to be adapted for different sensornet shapes than those using structured replication. Hence, random replication is simpler and more cost-effective.

6.3.2 Qualitative Comparison

It must be noted that allocating replication nodes at random presents several advantages in front of any structured replication mechanism.

First of all Random Multi-Replication is the simplest mechanism since it only needs to generate several location using a simple hash function, whereas grid replication mechanisms such as ToW [JH08], QAR [CUnRL10] and GHT with multiple replicas [RKY+02] need some extra operations like dividing the network in different cells and place replicas in a grid.

Furthermore, Random Multi-Replication is easily applicable to any network shape, since random locations could be sequentially generated with the hash function until the output location is contained within the network field. However, replicas located in a grid would not match for irregular network shapes and an inner square envelope need to be pre-computed in order to be applied.

Finally, structured multi-replication DCS proposals only are valid when there is geographic information available, i.e. geographic routing. However, if we think in other routing alternatives, such as a distance-vector routing protocol without geographic information (number of hops to a destination node), those proposals could not be used. However, Random Multi-Replication works with any routing protocol, it does not matter if it uses geographic information or not. For instance, with a distance-vector or link-state routing protocol some nodes could be randomly selected as replicas and connected among them in order to generate the replication tree. Thus a very important benefit of Random Multi-Replication is that it can be applied without taking into account which is the routing protocol utilized in the network. While structured multi-replication DCS networks only work when geographic information is available.

Hence, we can conclude that Random Multi-Replication is not only more effective in quantitative terms of traffic reduction, but also have an important qualitative advantage in front of structured replication mechanism.

Chapter 7

Dynamic Muti-Replication DCS framework

Nodes selected as rendezvous nodes (and those close to them) will naturally expend more energy than other nodes. Thus, if the responsibilities of such nodes never change, they will run out of energy reserves before than other nodes. In most DCS proposals [SRK+03, RKY+02, RKS+03, JH08, CUnRL10] when this happens, a backup node close to the previous one is selected as the new rendezvous node, until its battery also expires, and so on. After some time, routing (and sensing) holes are generated around the original rendezvous coordinates, affecting the routing of the whole network.

However, if rendezvous points change over time, the extra energy expenditures associated with rendezvous nodes can be balanced across all the nodes in the network, thus delaying the creation of routing holes and extending the sensornet lifetime. However, it must be noted that changing replicas over the time has an extra cost. This does not automatically mean that network energy expenditures become higher than keeping replication nodes static. Indeed, when replication nodes are kept static, longer paths will be required to surround the routing holes, which in turn will consume more energy. In this chapter we demonstrate that routing holes can have more impact on the overall network energy expenditures than the cost of changing the set of replication nodes.

In fact, the cost associated with changing the replication nodes directly depends on the amount of data that old replicas have to forward to new ones. Therefore, applications that only employ real-time data, i.e. the last temperature measurement, do not require data transition when a new replication set is selected. This implies a very low-energy expenditure associated to the change of replication nodes. However, there could be some applications in which this transition could be very costly, because they require to move all previous data to the new replicas.

Furthermore, by changing the set of replication nodes, we can adapt the deployed number of replicas to the current traffic conditions. Therefore, this proposal opens the possibility

of an adaptive solution for dynamic traffic applications, e.g. applications that reduce their traffic at night.

Then, changing the replicas over the time really seems a promising idea but, in order to be really useful in the real world, a lot of practical issues need to be considered. Some of these open questions are:

- How do consumers and producers realize that the current set of replication nodes have changed, so that they compute a new set of replicas to maintain the application operation?
- How do we decide that a new set of replicas should be selected?
- How are new replicas notified of its new role?
- How do the current replicas move the information to the new replication set?
- How are consumers, producers and replicas synchronized in this dynamic environment?

Therefore, we need to extend the existing DCS protocols and develop some new algorithms to solve all previous questions in an efficient way, so that, a very low traffic overhead is generated.

The idea of changing replicas over the time is applicable to most DCS works, including QAR and Random Multi-Replication. However in this chapter we will just be focused on Random Multi-Replication, since it has been shown to be the best multi-replication DCS solution in both qualitative and quantitative ways.

We define an *epoch* as the time that a set of rendezvous node plays that role. We refer as an epoch transition the action of changing from an old set of replicas to a new one. Therefore, our framework protocol is designed taking into consideration the epoch definition. Then in order to generate the replicas location an epoch identifier (e) should be included in the hash function as parameter, so that any node will be able to generate the suitable location for the replication nodes at a given time. Therefore, the new hash function is $hash(app \oplus e \oplus i)\ \forall i \in [1, N_r]$

Hence, in this chapter we describe our Dynamic Random Multi-Replication DCS proposal [CUndV10] that covers most of the practical issues mentioned above. First, we will introduce the system operations including what we call Meta-Information Service, which is just another application that provides: bootstrapping for new consumers and producers that want to participate in an application running, fault-tolerance mechanisms, application initialization, etc. Next, we present an analytical model to quantify the cost of changing the replication nodes over the time. Finally, we present an exhaustive evaluation of our framework and compare it with a static deployment. The main conclusion is that our spatio-temporal adaptive framework has a huge impact on the network lifetime. It must be noted that we have been focused on the case when Consumption-dominates-Production in order to evaluate our framework proposal.

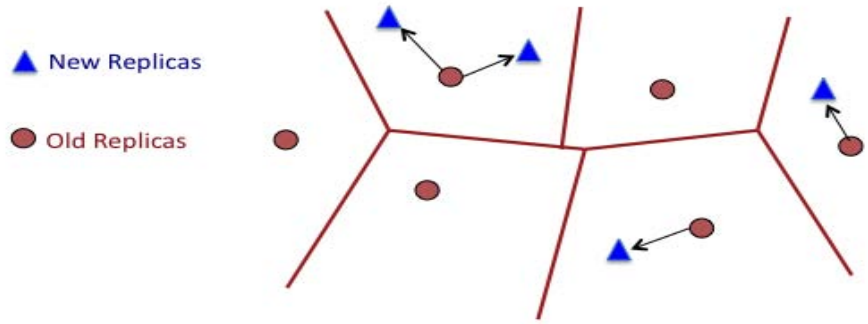


Figure 7.1: *Example of epoch transition from old to new replication set of nodes.*

7.1 System Operations

7.1.1 Changing the set of replication nodes

We consider two events that could trigger an epoch change: (i) when a node serving as a replication node exceeds certain threshold in the number of messages sent and received since the epoch started; and (ii) just before one replication node runs out of battery. Whichever happens first triggers a change of epoch.

A simple idea for changing the replicas may be to choose a fixed time period between epochs However, if we employ the same period for all the applications in a heterogeneous traffic scenario, this period could be suitable for some applications but not for others. In front of this, our mechanism is adaptive and each rendezvous node just needs to maintain a message counter per application. This mechanism is more adaptive because it makes replicas supporting a high load to use shorter epoch periods than low loaded applications, which can support long epoch periods to minimize the overhead of epoch transitions.

At the beginning of each epoch, rendezvous nodes gather local traffic statistics (number of messages sent and received, traffic intensity in bits/sec, etc) during a predefined time interval Δt. After that time, each rendezvous node broadcasts over its replication tree (by using piggybacking in data packets or dedicated control messages) its local production/consumption traffic measurements and its estimate for the residual time of the current epoch to the remaining replicas. In turn, based on the exchanged estimates, each replication node computes the minimum estimate for the current epoch's residual time, along with the number of rendezvous nodes that should be used in the next epoch, also based on the overall measured traffic. It must be noted, that these messages containing local traffic measurements must be acknowledged by the other replicas, and in case the ACK fails they need to be retransmitted. In addition, before agreeing on the current epoch deadline and number of rendezvous nodes for the next epoch estimation, each replica must be sure that it has received the information from all the other replicas (messages containing local traffic measurements) and that its information has been received by all the remaining replicas (by means of ACK messages). Otherwise, the computation with inconsistent data could lead to replica de-synchronization because of the different epoch duration estimations.

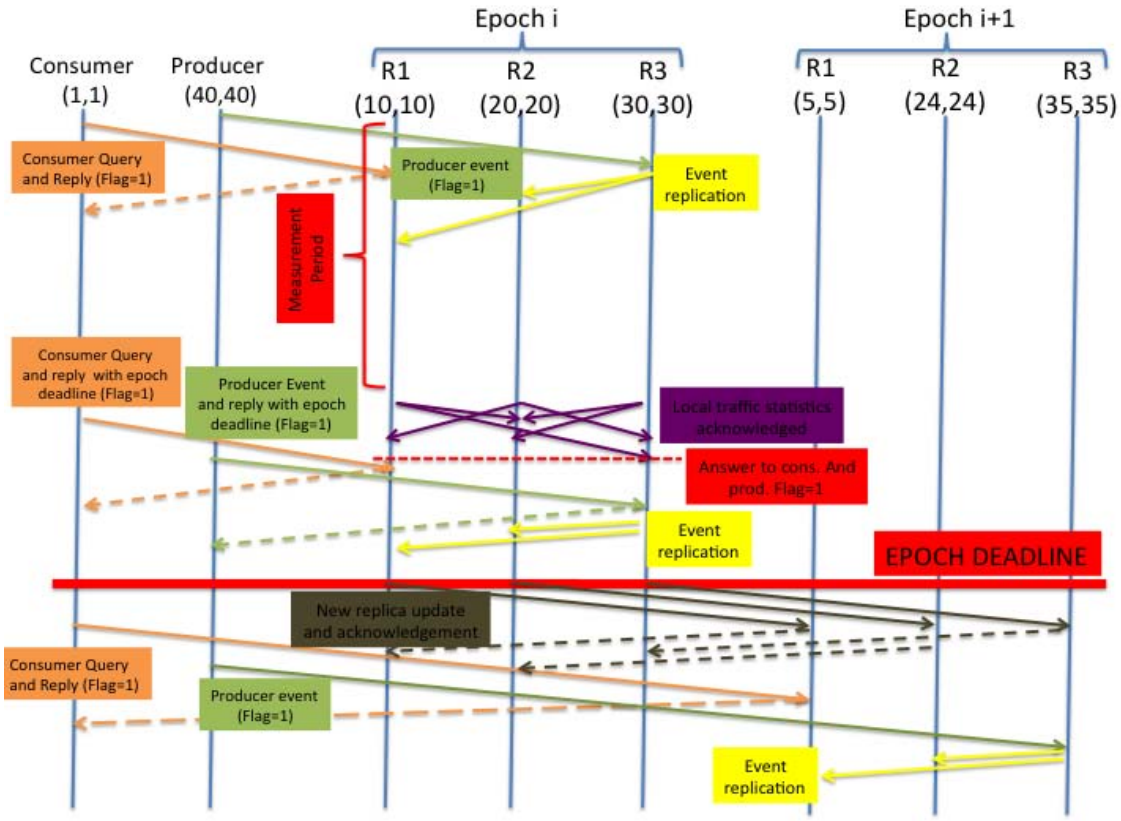


Figure 7.2: *Time diagram including the operations proposed for changing the replication set over the time in practice.*

When the estimated epoch deadline arrives, current replication nodes know the locations of the current set, and can compute the locations of the (possibly different) number of nodes in the set for the next epoch using the epoch-dependent hash function. Now each of the current rendezvous nodes needs only to determine if it is the closest node to one of the nodes in the subsequent set. If so, such nodes can directly transfer in parallel their stored data to the new locations. The message that notifies a node its new replica role in the new epoch must be also acknowledged and retransmitted if necessary. Figure 7.1 shows a simple example of an epoch transition.

7.1.2 Consistent notification of epoch changes to producers and consumers

Once the current set of replication nodes decides on the next epoch change, consumers and producers need to be notified about when it will be initiated and the new number of replicas to be used. This can be achieved as follows: At the beginning of an epoch, active consumers and producers set a flag in their messages. This flag indicates to the replication node that this particular consumer or producer does not yet know the current epoch duration, nor the number of replicas for the next epoch. After Δt, when the current replication nodes

have agreed both values, they send a notification message or piggyback this information back to producers and consumers, respectively. Consumers and producers receiving the information can then cancel the flag until the beginning of the next epoch. This simple and robust mechanism does not require replication nodes to know who the producers and consumers are, thus saving memory and enabling scalability. By proactively predicting and sharing information about epoch changes, replicas, consumers and producers are able to experience a smooth epoch transition.

Figure 7.2 summarizes all the described operations in a time-line diagram to help to understand how the messages flow between the different participants in an application that use our Dynamic Random Multi-Replication DCS framework.

7.1.3 Meta-Infomation Service

In order to become a viable solution, our Dynamic Random Multi-Replication framework has to be further developed to address several practical issues. This section provides additional insight on developing a real-world Data Centric Storage protocol with multiple randomly-placed replicas that change over the time. Such a protocol should implement additional mechanisms in order to:

- Provide a bootstrapping mechanism for finding the current set of replicas if the epoch value is unknown.
- Provide fault tolerance in the case replication nodes fail.
- Provide a mechanism to bring new applications online.

In order to solve the bootstrapping problem when a new node wants to participate as an application producer or consumer, we propose employing a Meta-Information service where every application in the network stores its current epoch value and the number of replication nodes currently in use. Once a new sensor acquires this information, it can then ask detailed information to the replicas concerning the time at which the epoch will expire and the number of replication nodes to be used in the next epoch by using the flag mechanism introduced above. This Meta-Information service is just another application that itself may use the proposed replication framework.

The question now is how a new sensor is able to know the current epoch of the Meta-Information service. A straightforward solution is broadcasting it to the network when a Meta-Information epoch change happens (e.g. once per hour/day). Since the number of changes could be arbitrarily low, the energy consumption would be negligible. Then, when a sensor bootstraps it can simply ask any of its neighbours what is the current Meta-Information epoch.

Another aspect that should be taken into account is determining how the Meta-Information service knows that a given application is changing its epoch. The first replication node (i=1) could be the one notifying each epoch change to its closest Meta-Information

service replication node, which in turn notifies the new epoch to the remaining Meta-Information replication nodes. That is, application's replicas behave as producers of the Meta-Information service. It must be noted that the message that notifies the epoch change must be acknowledged since the information it contains is vital for allowing new sensors to participate in the applications. Therefore, if the replication node selected to notify the epoch transition does not receive the acknowledge, it keeps retransmitting the message to the closest Meta-Information service node.

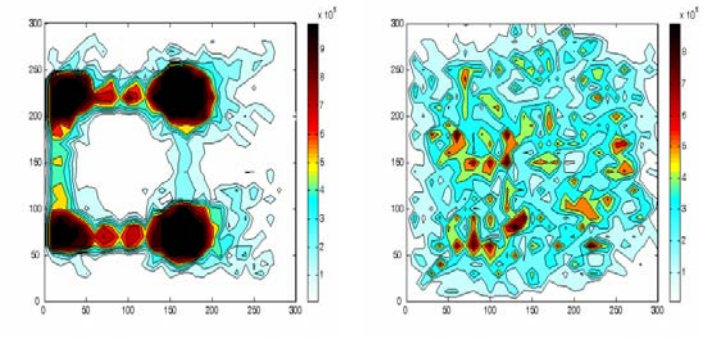
The Meta-Information service can be also employed as a fallback mechanism in case of replication node failure or epoch de-synchronization. If a node fails in accessing its closest replication node for a pre-defined number of times, it tries to contact the remaining replication nodes (sorted by distance) from the current epoch, because these locations can be also computed locally by the sensor. In the case the sensor has suffered an epoch de-synchronization, it can contact the Meta-Information service that replies with the current epoch and number of replication nodes being used for that application.

Finally, we shall define how a new application can be brought online on a sensor network. When any of the replication nodes of the Meta-Information service receives a query from a bootstrapping producer requesting the epoch and number of replicas of an unknown service, it understands that this application does not yet exist. Therefore it registers the application and assigns to that service a random epoch number and a single replication node. After that, the Meta-Information node notifies to the first application's replication node that the service needs to be started, sharing the initial epoch number with both the replication node and the first producer. From that moment any sensor can start using the new application.

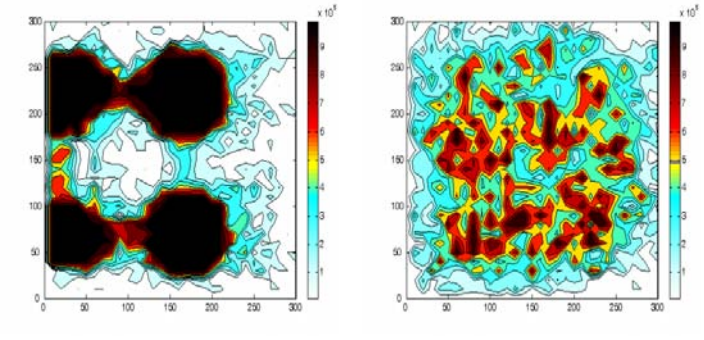
7.2 Cost model of changing the set of replication nodes to balance network loads.

We have argued that it would be worthwhile to periodically change the set of nodes where data is replicated. To that end we propose a mechanism whereby the next set of replication nodes locations are also randomly selected, and current replicas push (produce) the data to the closest node among the new set of replicas. Subsequently producers/consumers would push/get data from the new set of replicas. The cost of moving from one set of replica nodes to another should be relatively low since this is a highly-parallel distributed process. In particular, suppose the current intensity of replicas is λ_r^c and we wish to move to a new set of randomly-located replicas with intensity λ_r^n. Note that the new set of replicas does not need to have the same intensity as the current one. Also suppose that each replica node currently holds an average amount of data d.

A rough estimate of the energy cost associated with moving data from the current set of replication nodes to the new one $T_r(\lambda_r^c, \lambda_r^n)$, can be evaluated as follows. Each old replica would contact one of the new nodes. Given that the distance to a new randomly located replica from one of the current replication nodes is $\frac{1}{2\sqrt{\lambda_r^n}}$ the total cost in a network of area



(a) ToW-Static after 30K cycles (b) Random-Epochs after 30K cycles



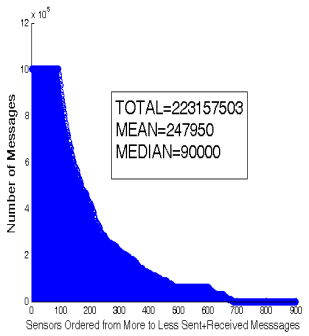
(c) ToW-Static after 50K cycles (d) Random-Epochs after 50K cycles

Figure 7.3: *Energy map from the number of messages sent and received by all nodes of the network (A=300x300 m^2, N=900 nodes, Tx=30 m, N_p=100 producers, N_c=300 consumers, L=10 cycles, M=1000 $\frac{messages}{epoch\ transition}$, Battery = 10^6 messages, E_{th}=3*10^5 messages).*

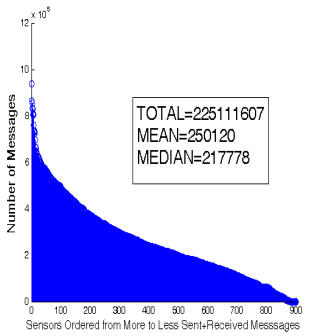
A would be :

$$T_r(\lambda_r^c, \lambda_r^n) = \frac{d}{2}\frac{\lambda_r^c A}{\sqrt{\lambda_r^n}}. \text{ bits-m}$$

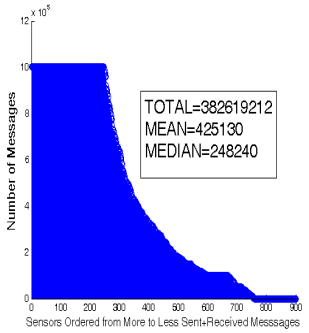
So if $\lambda_r^n = \lambda_r^c$ the cost is $T_r = \frac{d}{2}\sqrt{\lambda_r}$ bits-m. If the set of replication nodes changes infrequently, then the contribution to the overall network traffic and energy consumption of changing the set of replicas would be fairly small. However this does depend on λ_r, the size of the transfered data, and the frequency of such updates. We shall consider this in more detail in the next sections.



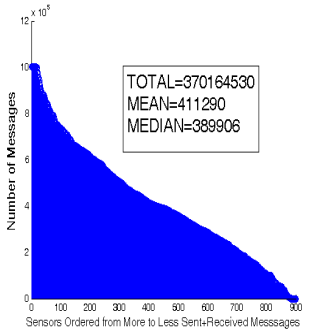
(a) ToW-Static after 30K cycles



(b) Random-Epochs after 30K cycles



(c) ToW-Static after 50K cycles



(d) Random-Epochs after 50K cycles

Figure 7.4: *Distribution of the number of messages sent and received per node (A=300x300 m^2, N=900 nodes, Tx=30 m, N_p=100 producers, N_c=300 consumers, L=10 cycles, M=1000 $\frac{messages}{epoch\ transition}$, Battery = 10^6 messages, E_{th}=3*10^5 messages).*

7.3 Performance Evaluation

7.3.1 Network Traffic and Lifetime evaluation

In order to verify the goodness of changing the replication nodes' set over the time, we have run simulations comparing ToW [JH08] using static replicas with Random Multi-Replication where the set of replication nodes changes over the time. Some other solution instead of ToW could have been chosen to evaluate the performance of a static solution. However, we have chosen ToW since it is the one with best static performance other than Random Multi-Replication and QAR that are our own proposals. We use a grid-based node deployment (which makes energy maps generation easier) with $N = 900$ nodes, over a square field of area $A = 300x300$ m^2. Each sensor has a transmission range $Tx = 30$ m. We use the number of messages in the network as a first order proxy for consumed energy. A node's energy is depleted once it sends and/or receives one million messages. Finally, the

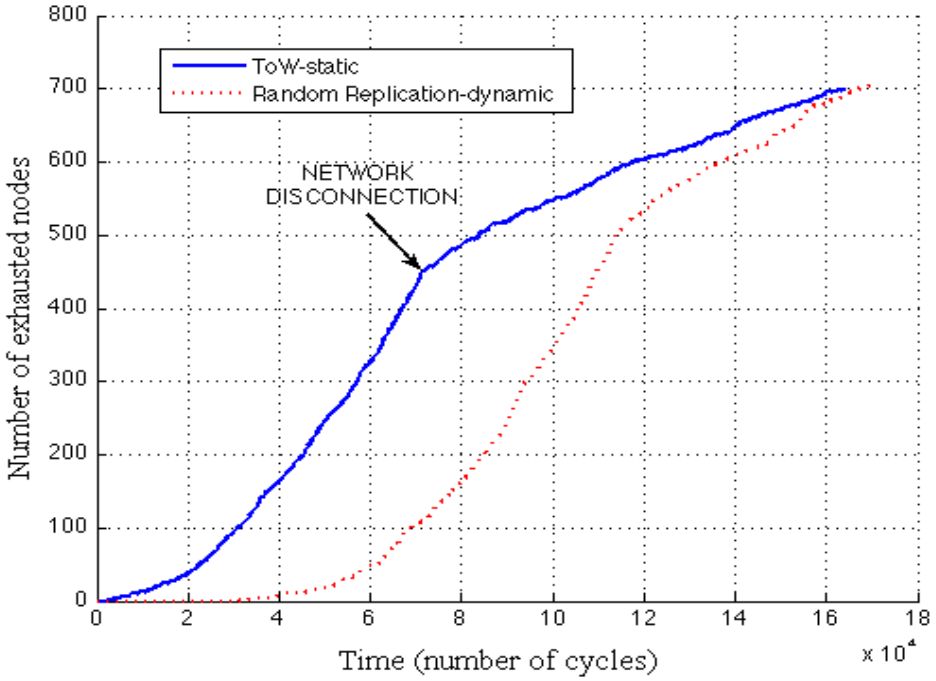


Figure 7.5: *Number of exhausted nodes along the time.(A=300x300 m^2, N=900 nodes, Tx=30 m, N_p=100 producers, N_c=300 consumers, Battery = 10^6 messages)*

Lifetime Criteria	1^{st} *dead*	*1% dead*	*10% dead*	*25% dead*	*10% cons+prod*	*25% cons+prod*	*Network disconnection*
ToW-Static (cycles)	2619	7328	29086	47830	31668	47984	70952
Random-Epochs (cycles)	31199	41124	66750	87968	65171	79717	170950
Improvement(%)	1091%	461.2%	129.5%	83.9%	105.8%	66.13%	140.9%

Table 7.1: *Sensornet Lifetime of ToW-Static vs Dynamic Random Multi-Replication*

threshold for the end of an epoch, E_{th}, is set to 300000 messages (30% of a full battery) [1].

For these simulations we have used geographical routing based on greedy forwarding [KK00]. When greedy forwarding fails, i.e. due to routing holes, we use the shortest path from the node where the greedy forwarding stopped to the destination node. If that path cannot be found (in consumption, production or replication), then we count it as routing error.

Time is measured in cycles in order to scale the simulations and to be able to deploy a large number of nodes. A cycle is the time period in which every consumer node performs one consumption query and every producer node generates a production event. Since energy is measured in terms of messages, the traffic[2] is measured in $messages/cycle$. We de-

[1] We also ran experiments for $E_{th} = 100000$, $E_{th} = 500000$ and $E_{th} = 700000$ messages and in all of them changing replicas clearly outperformed the static solution.

[2] The production (λ_p) and consumption (λ_c) intensities are then measured in $\frac{messages}{cycle*m^2}$.

ploy N_c=300 consumers, which means 300 queries and 300 replies per cycle, and N_p=100 producers that generate 100 production events per cycle. The consumption to production traffic ratio results in an optimal number of replicas equal to 4 for both ToW and Random Replication

In order to measure the cost of an epoch change, we assume that the produced data has a mean lifetime of L cycles. Then, since the production intensity is λ_p and the network area is A, the average data at each replication node is $d = \lambda_p AL = N_p L$ messages. L is set to 10 cycles for these simulations, thus the replication change is quite costly, because it means that 10 messages per producer have to be moved from the old replicas to the new ones. Hence, having 100 producers, this means a total cost of M=1000 messages per epoch change.

Figure 7.3 shows the energy distribution map after a simulation time of 30000 and 50000 cycles. Figure 7.4 shows the number of messages sent+received by each node during the same cycles, as well as the mean and median values per node, and the total messages sent and received in the whole network, that roughly captures the total energy consumed by all the network nodes.

As seen in Figures 7.3(a) and 7.3(c), keeping static replication points creates routing holes in the network, with 93 and 247 expired nodes after 30000 and 50000 cycles respectively. The number of depleted nodes are only 0 and 17, respectively, when replication points are changed over the time. In addition, more nodes participate of the network operation when epochs are used. As shown in Figure 7.4, all nodes except 18 (a 2%) after 30000 cycles and 15 (a 1.7%) after 50000 cycles, have sent and/or received at least one message, whereas in the case of static replication nodes more than 200 (a 22%) nodes have not sent or received any message after 30000 cycles, decreasing to 160 (a 17.8%) nodes after 50000 cycles. When considering the total energy consumed by the network, the dynamic approach uses just 0.8% more energy than the static one after 30000 cycles. However, a 3.3% extra energy is required by the static approach after 50000 cycles. This shows that the cost of using longer routing paths eventually exceeds that of changing the rendezvous nodes over time, even with so many transfer messages per epoch.

Furthermore, Figure 7.5 shows the number of nodes that runs out of battery over the time for ToW-static and our dynamic framework. It is clear, that in the static solutions the sensor runs out of battery quicker. In addition, at some point (after 70959 cycles) the network running ToW-static is disconnected. This means that the routing holes coalesce and divide the network in two disjoints parts that are isolated from each other. However, even in the long-term there is no network disconnection in the dynamic random approach.

In Table 7.1, we compare the sensor network lifetime using both approaches: static ToW and Dynamic Random Multi-Replication. Since the sensornet lifetime can be defined by different metrics [DD09] (first sensor running out of battery, some percentage of nodes running out of battery, important nodes like consumers and/or producers running out of battery, some part of the network disconnects and many messages are lost, etc), we provide a broad overview of metrics to let the reader establish a fair comparison depending on the criterion used to define the network's lifetime. The table shows the number of cycles spent until each lifetime criterion is reached. For all the criteria our solution extends the network's lifetime

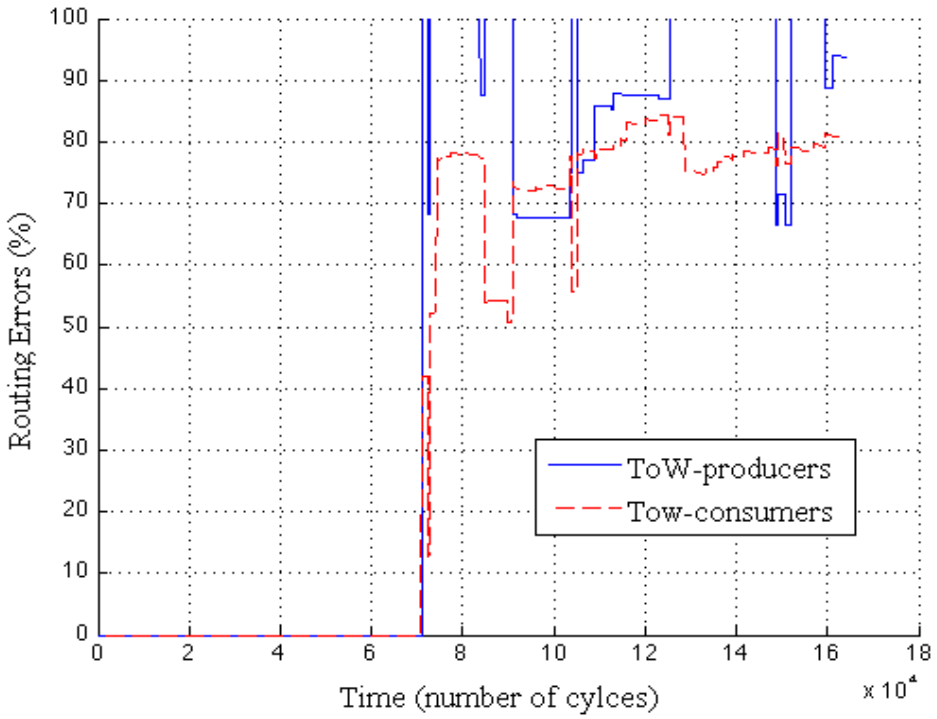


Figure 7.6: *Routing errors per time cycle using ToW with static replicas (A=300x300m^2, N=900 nodes, Tx=30m, N_p=100 producers, N_c=300 consumers).*

by at least 66%. In particular, this worst lifetime extension value is obtained when the 25% of the consumers or producers (100 nodes) run out its battery. Furthermore, the value provided by Dynamic Random Multi-Replication for the case of network disconnection does not refer to an actual disconnection, but the time when 700 nodes die in the simulation, which is our simulation stop condition. By that time none message has been lost if our solution is applied.

We note that in many cases, changing replicas over the time and using Random Multi-Replication extends the network's lifetime by a factor of 2x.

In addition, Figure 7.6 shows the routing errors accounted in each simulation cycle (it must be noted that in every cycle each producer generates one event and each consumer one query). A routing error happens when there is not a feasible route from a consumer to its closest replica, or when an event being replicated do not reach one or more of the replication nodes.

On the one hand, the network using ToW-static solution does not present any routing errors until the network disconnection. Once the network is disconnected, the percentages of consumption and production errors grow a lot, over a 50%. Therefore, we can conclude that after the disconnection time, an application using ToW-static would not work anymore due to the low message reliability.

On the other hand, our solution does not have any routing error even in the long-term.

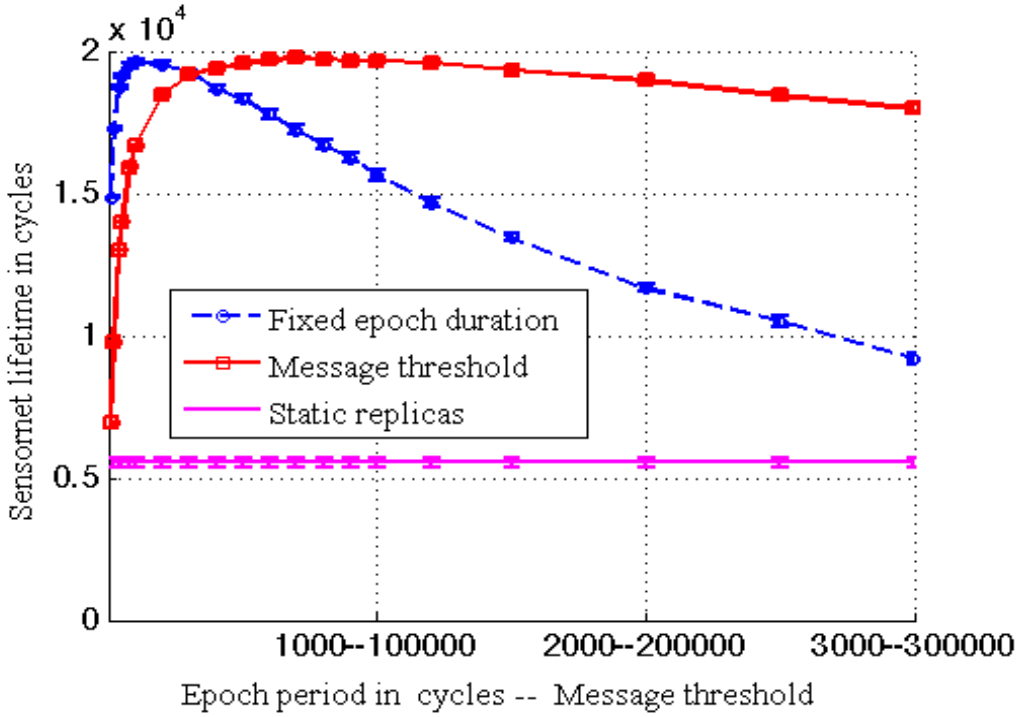


Figure 7.7: *Sensornet lifetime comparison for different epoch estimation mechanisms (A=1000x1000m^2, N=5000 nodes, Tx=50 m, L=10 messages/producer, m=5 applications). X axis refers to the cycles for changing the epoch in the fixed duration approach, or the message threshold employed.*

The reason is that, changing the replication point set over the time balances the energy across the network and the nodes that spend more energy are just those ones located around the network centre, since many routes pass through that region. Therefore, after some time the network presents a routing hole in the centre. After that, routing paths have to surround that hole making it grow along the time. However, nodes that are still alive are able to communicate to each other since there are not network disconnection but just a single routing hole in the middle of the network.

Finally, Figure 7.7 shows how using a message threshold to trigger epoch changes compares to employing a fixed epoch duration as proposed in [TWXD06] [3]. We simulated a larger (N=5000 nodes, A=1000x1000 m^2, Tx=50 m) multi-application WSAN with Random Multi-Replication. We have set up m=5 heterogeneous applications with (20, 40), (60, 120), (100, 200), (140, 280) and (180, 360), ($\frac{events}{cycle}$, $\frac{queries}{cycle}$) traffic pairs, and calculated the network's lifetime (1% of nodes expire) for the two dynamic approaches versus using a fixed static set of randomly located replicas. Again the need of changing replicas over the time versus using static ones is clear. In addition, as seen in the figure, our proposal to trigger epoch changes based on message counters is more robust to the precise setting of the message threshold than using a fixed epoch duration.

[3] note that this work actually refers to a single rendezvous node scenario, and that it proposes this change motivated for storage saturation instead of energy issues.

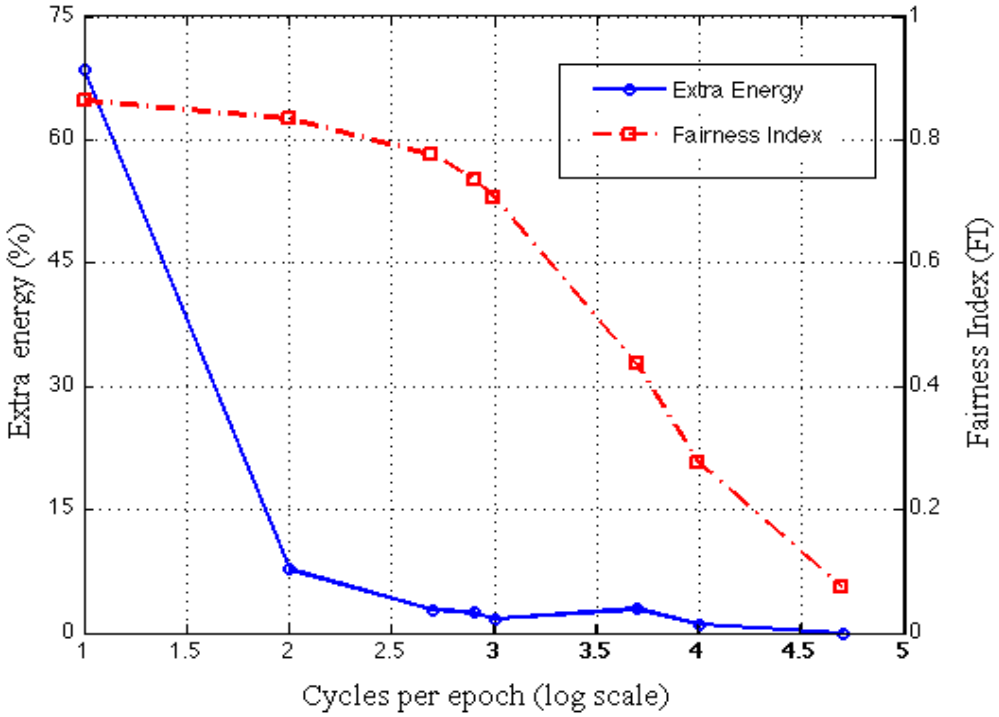


Figure 7.8: *Epoch duration analysis (A=300x300 m^2, N=900 nodes, Tx=30 m, L=10 messages/producer, Simulation Time=50000 cycles).*

7.3.2 Epoch duration analysis

We have already demonstrated the great improvement of changing the replication nodes over the time. Thus the next question is whether it is better to use shorter or longer periods before an epoch transition. At first glance, the best solution seems to be using short periods so that the load is better spread among the nodes. However, as we have already seen, there are some overheads associated with epoch transitions, such as moving all the data stored in the current replicas. Considering this tradeoff, using shorter epoch periods will lead to balance the energy consumption among the nodes, thus reducing the energy consumption variance per sensor, but it would also increase the average energy expenditures per node, which means increasing the overall energy expenditure. In contrast, using longer epoch periods increases the variance of the per node energy consumption since nodes will keep being replicas for more time, but it will reduce the overall (average) traffic on the network.

Thus the key of this trade-off is determining how much extra energy should be spent to balance the energy among the nodes. The decision depends on each particular application. For instance, for an application where all the nodes are needed, so that all of them should kept alive in order to allow the application working properly, (i.e. extend the time when the first sensor runs out of battery) the right selection is to balance the energy as much as possible, by means of using short epochs (frequent epoch changes). Of course the price of doing so is that a lot of extra energy is required for those frequent epoch changes. In front

of this, if the application requirement is to reduce the overall energy consumption, then very frequent changes is a wrong decision.

To evaluate this trade-off we have run simulations in a WSAN with the same simulation parameters used in the previous section. That is, a grid deployment in an area A=300x300 m^2, meaning N_s= 900 nodes. N_c=300 consumers and N_p=100 producers were used with a transmission range T_x=30 m. Ten messages per producer ($L = 10$) are stored in the replicas, being this factor the one establishing how costly an epoch transition is. We use the number of messages sent and received by each node as an approximation of the energy consumed by the network.

We have evaluated the network performance using different fixed epoch durations: 10, 100, 200, 500, 1000, 5000, 10000 and 50000 cycles per epoch. For each case we have run the simulation during 50000 cycles, thus going from 5000 epoch changes to none. For each epoch duration we have run 50 simulations to get an average value. It must be noted, that nodes do not run out of battery in this experiment,so no routing holes are generated, and that is why in terms of traffic overhead, the best option is to use none epoch changes. However, as it has been previously demonstrated the need of longer routing paths could have a larger impact in the traffic overhead than the change of replication nodes over the time.

We use the Fairness Index (FI) [JCH84] as a measure of how well balanced the energy consumption is across the nodes,

$$FI = f(x_1, x_2, x_3, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

and the relative energy increment ($\Delta\epsilon$) of the overall energy expended in the network relative to the minimum energy case (i.e. no changes).

Figure 7.8 shows the FI and $\Delta\epsilon$ for different epoch durations on a logarithmic scale. As expected, the lower the number of epochs per cycle (the more epoch changes) the better the network fairness, whereas the greater overall energy is required. However, there is a region, around 500 cycles per epoch (between 2.5 and 3 in the x-axis in the figure), where ($\Delta\epsilon$) is not that big, below a 5%, and the FI is very good (>0.75). This operational regime heavily depends on the value of L, that specifies the overhead associated wtih epoch changes. If this cost is low the region will move to the left, thus having a better FI and a lower $\Delta\epsilon$. However, if the transition cost is very high, the region with a low $\Delta\epsilon$ will move to the right, of course producing worse FI values, thus a bigger variance for the energy consumed per sensor. Although the pointed region could be a good operation area in general, we again note that each particular application will have a different suitable operation area.

Chapter 8

Implementation

We have implemented the proposed Dynamic Random Multi-Replication DCS framework in real motes. It supports an arbitrary number of replicas that change over the time, and uses the Meta-Information Service for bootstrapping purposes. Furthermore, replicas are able to exchange traffic measurements and compute the current epoch deadline, as well as the optimal number of replicas for the next epoch. All this provides an efficient synchronization mechanism for all the involved players of a particular application: consumers, producers and replication nodes.

It must be noted that currently the framework only supports the Consumption-dominates-Production mode. However, most of the code would be also valid for the Production-dominates-Consumption mode.

To test this implementation we have used 20 Jennic motes with two different hardware versions: JN-5121 (x5) and JN-5139 (x15) to check that both of them could interact together when our framework is implemented. JN-5139 wireless microcontroller device integrates a 32-bit RISC processor, with a fully compliant 2.4 GHz IEEE802.15.4 transceiver, 192kB of ROM, 96kB of RAM, and several analogue and digital peripherals. JN-5121 is an older mote version and only has 64kB of ROM, enough for the code of our implementation.

We have implemented the following functionalities:

- Greedy forwarding
- Producer node
- Consumer node
- Replication node
- Replication tree creation
- Computation of the optimal number of replicas

- Change of replication nodes over the time
- Epoch synchronization without global clock
- Meta-Information service
- Multiple applications in parallel

A node could be initially assigned with one of the following three roles: producer, consumer or relay node (in case it is neither a consumer nor a producer). The role of a particular node is expected to be assigned by the application using the framework.

Producers generate production events using two different mechanisms: (i) based on a pre-defined rate, that is, they generate a production event periodically (e.g. temperature reading every minute); (ii) or manually, when a button is pressed. Moreover, the number of data elements per event can be configured (e.g. three temperature samples per production event) by the application. Consumers also generate queries using the same two mechanisms utilized by the producers: constant query rate and manually mechanisms. Finally, all nodes act as relay nodes and forward messages from their neighbours.

8.1 Greedy Forwarding Routing

We have implemented a greedy forwarding algorithm on the motes as the routing layer to be used by our framework. In order to avoid more complex routing operations (i.e. face routing), we have set up scenarios in which it was feasible to route a message from any source to any destination node by only using greedy forwarding. Therefore, the implemented routing layer chooses as the next hop the neighbour that is the closest one in distance to the final destination point.

In addition, our framework also needs to find out which is the closest node to some given coordinates. In the scenarios we have used to evaluate our solution (without routing holes) if a node receives a message and it is closer to the destination coordinates than any of its neighbours, then it assumes that it is the closest one to the destination coordinates.

8.2 Framework Protocol Header

We have defined a common framework header for all the messages used by the framework. Figure 8.1 shows all the different fields included in the header. Next, we describe each of these fields:

- **OPERATION CODE:** It defines the type of message (PUT, GET, GET REPLY, etc).
- **FLAGS:** It is used for special operations: (i) consumers and producers use in the dynamic implementation one of the bits in this field to indicate that they do not yet know

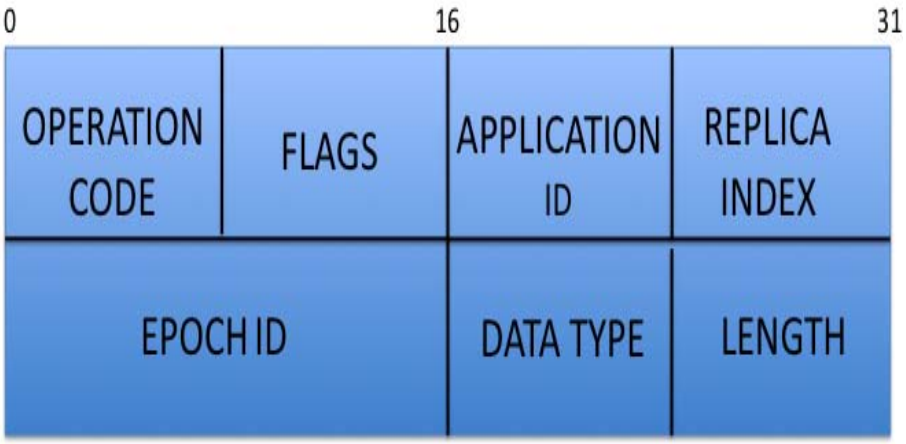


Figure 8.1: *Framework Header.*

when the current epoch finishes or what is the number of replicas to be used in the next epoch. (ii) ACK required flag, when the source node asks for an acknowledgement of this message. For instance when a producer wants to be sure that its data has been stored at the target replication point. This is the case for PUT operations of the Meta-Information service that need to be acknowledged to avoid epoch desynchronization.

- **APPLICATION ID:** It defines what is the application that is using the framework.
- **REPLICA INDEX:** It identifies (in the case that is required) the replication node that is the source of the message (e.g. that information is needed by all replication nodes in order to locally generate the replication tree). For instance, if there are 4 replicas in the field they are identified as 1, 2, 3 and 4 from the ID used in the hash operation.
- **EPOCH ID**: It is defined to indicate what is the current epoch of the source node and it is used for synchronization purposes.
- **DATA TYPE:** It is used to define the particular data structure used by the application to carry the measurements data.
- **LENGTH:** It indicates how many data structures are included in the message.

8.3 Static implementation

Initially we implemented the static version of our framework in which replication nodes do not change over the time, thus only random replica allocation and replication tree construction were implemented.

In the static case we have four different message types that are identified by different OPERATION CODE values:

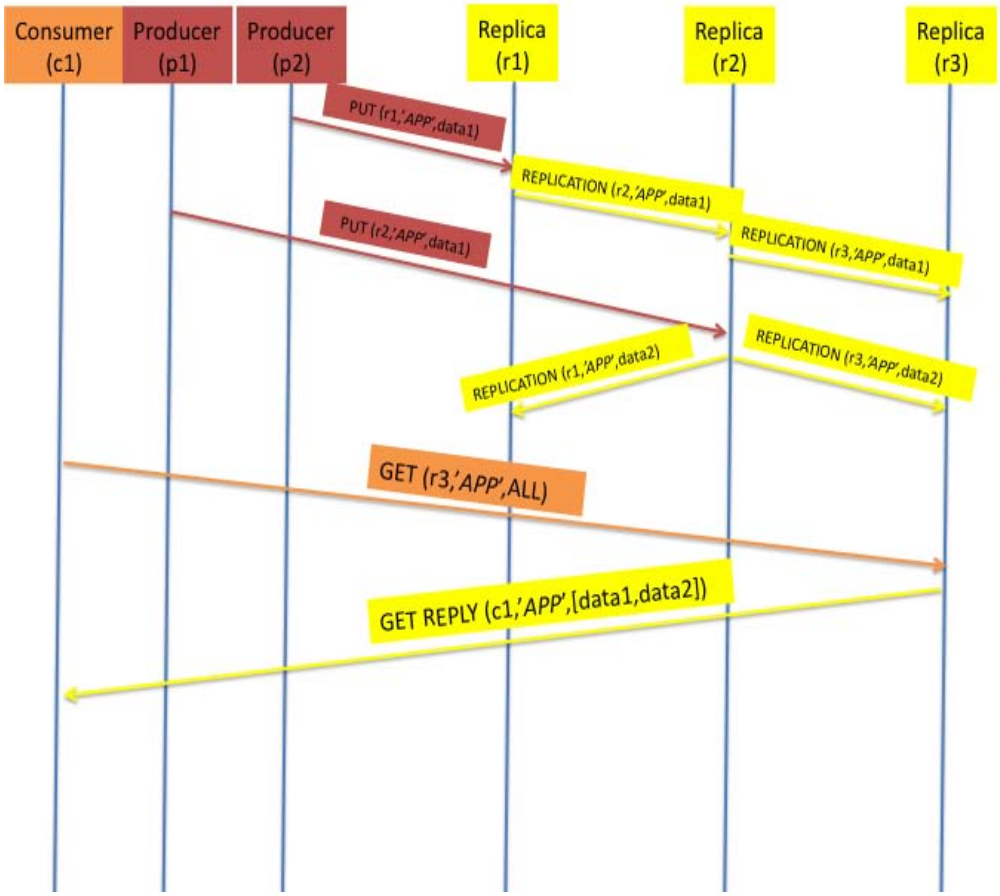


Figure 8.2: *Example of messages exchanged in a Random Multi-Replication scenario with three static replication nodes.*

- PUT: It is the message used by producers to send the measured data to the closest replication node. This message is confirmed with a PUT ACK message if the ACK required flag is enabled.
- GET: It is the query message used by consumers to obtain information from the closest replica.
- GET REPLY: It is the message sent by a replication node that answers the consumers' GET message. It contains the suitable information requested by the consumer.
- REPLICATION: It is the message used to replicate the producers' data received by one replica in the remaining replicas.

Figure 8.2 shows the messages exchanged by different operations of producers and consumers. As it can be seen, framework operation is quite simple, producers generate PUT messages at a predefined rate or when a button is pressed in the motes. PUT messages include one or more measurements from the temperature sensor present in the Jennic motes.

The number of temperature reads per PUT message is tuneable with a variable, so each application can specify how many measurements are included in a PUT message.

Consumers send GET messages to their closest replica also at a predefined rate or when a button is pressed. In the GET message the consumer can specify how many data measurements of the requested event type it is soliciting, which could go from one up to all the data stored at the replica node. In order to specify the number of data records the consumer is requesting the LENGTH field in the framework header is used. This field is set to $0xFF$ in case the consumer wants to receive all the data stored for that application.

Finally, replication nodes generate two different type of messages: (i) REPLICATION messages are used to copy production events into the remaining replicas. Then, when a replica receives a PUT message, it always generates one or more replication messages in order to transfer the received information to the other replication nodes. For that purpose, it runs the algorithm described in Section 7.1 (Algorithm 1) to determine to which replication nodes it has to forward the messages. In the current implementation each node re-calculates the replication tree every time it receives a PUT or REPLICATION message in order to decide whether they have to forward the production event or not. We have checked that this is a low-cost operation in terms of processing and it avoids the necessity of storing a replication tree per replica. (ii) Replication nodes also reply consumers GET messages with GET REPLY messages. They read the number of event measurements requested by the consumer in the GET message and reply with the most recent ones. In order to facilitate this operation, replica nodes store the application information in order, from the most recent data to the oldest one. In case the memory reserved for storing the information is saturated, the replica node removes the oldest data in order to store the newest one.

In this static case there are not synchronization problems since those nodes selected as replicas remain being replicas until they run out of battery. The number of replicas to be used was configured during the initialization phase. Then, when a node first receives a PUT message and it is the closest one to the destination coordinates (calculated using the hash function over the application ID) than any of its neighbours, it becomes a replication node and starts behaving as a replica. This simple mechanism allows that if a replica runs out of battery, a new node will take its responsibility immediately after receiving the next PUT message.

8.4 Dynamic implementation

Once we implemented the basic static functionality, we added the feature of changing the replication set over the time. Our dynamic implementation follows the design guidelines described in section 7.1.

Therefore, once a node is selected as a replica, it starts a measurement period in which it accounts the local consumption and production traffic it receives. The measurement period is pre-defined and is tuneable so different applications can choose different measurement periods.

There are some additional messages that are required for the dynamic case:

- REPLICATION: It is the message used to replicate the production data received by one replica to the remaining replicas.
- REPLICATION STATISTICS: This message is used by a replica to send to the other ones the consumption and production traffic accounted during the measurement period. This message is sent through the replication tree and it is retransmitted after some pre-defined time if an acknowledgement is not received. This message needs to be acknowledged with a REPLICATION STATISTICS ACK that does not use the replication tree but it is sent using a direct path.
- REPLICA INSTANTIATION: When the epoch expires, old replicas send a REPLICA INSTANTIATION message to the suitable new replication nodes, which become replicas for the new epoch after receiving this message. New replicas acknowledge those ones in the previous epoch with a REPLICA INSTANTIATION ACK that they have been instantiated and they are performing the replica role in the new epoch.
- PUT ACK: This message is used to notify producers about the epoch duration and the number of replicas to be used in the next epoch . This message contains the number of replication nodes that will be deployed in the next epoch and the time in seconds until the end of the current epoch.

Therefore, when a new epoch begins, the new replicas first gather local traffic statistics during the measurement period. When that period expires they exchange information among them and thus they can calculate the same epoch duration, as well as the same number of replication nodes for the next epoch using the formula presented in Eq. 6.1. For obtaining the epoch duration, we have implemented the threshold mechanism described in Section 7.1. The motes are configured with a message threshold parameter that defines the number of messages that should be sent+received by a replica node before changing the epoch. Therefore, after exchanging their traffic measurements, each replication node estimates locally which replica will be the first one reaching that message threshold, and when this will happen (e.g. after 3500 seconds).

As explained in Section 7.1, consumers and producers enable a bit in the FLAGS field of the GET and PUT messages respectively when a new epoch starts. This flag indicates that they do not know the epoch duration or the number of replicas in the next epoch yet. They keep setting this flag until they get a response. When the replication nodes have calculated both values, they include them in the GET REPLY or PUT ACK messages when answering consumers and producers, respectively. In particular, replication nodes include in those messages the time (in seconds) until the epoch expires, measured from the moment they received the GET or PUT message. Therefore, when receiving the GET REPLY and PUT ACK messages, consumers and producers establish a timer that expires at the time indicated by the replication node and disables the flag until the next epoch. In this way, consumers and producers are synchronized over the epoch deadline and are able to locally calculate the replicas in the new epoch and keep operating using them.

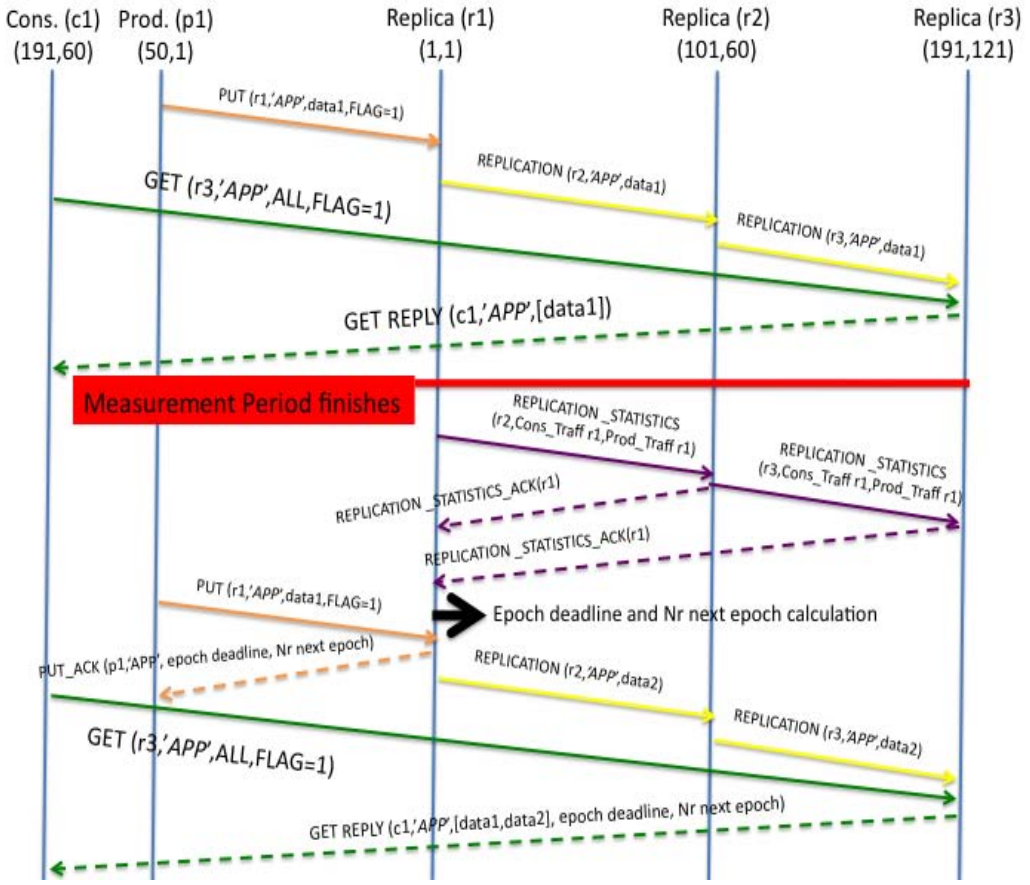


Figure 8.3: *Example of messages exchanged in a Dynamic Random Multi-Replication scenario with three replication nodes.*

It is important to notice that ACKs are required for REPLICATION STATISTIC messages. In our implementation a replication node can only perform the estimation of the number of replicas in the next epoch and the current epoch duration when: (i) it has obtained the traffic statistics from all the other replicas, and (ii) it is sure that all the other replicas have received its local traffic statistics. Otherwise, if we do not force both conditions, a replica could receive information from all the remaining ones so it could perform the parameters estimation, but its REPLICATION STATISTIC could be lost for some reason and all the other replicas could not preform the computation or in case they do it, they would reach different estimations.

Figure 8.3 shows a message exchange diagram covering a production event, two consumption queries and the messages exchanged among the replication nodes once the measurement period has expired. In addition, it is also shown how, after the measurement period, consumers and producers are updated with the current epoch deadline and the number of replicas to be used in the next epoch. Moreover, the diagram only shows one replication node sending REPLICATION messages and receiving REPLICATION ACK messages for a

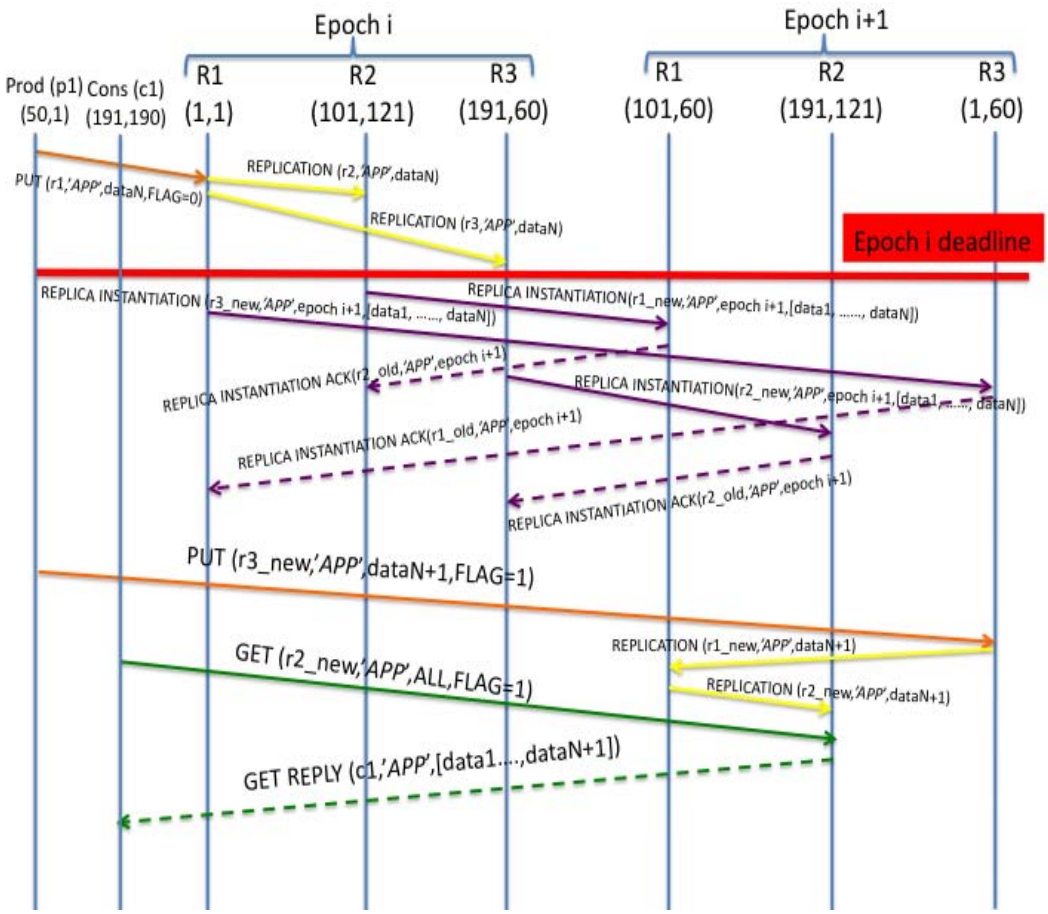


Figure 8.4: *Example of messages exchanged during an epoch transition in a Dynamic Random Multi-Replication scenario.*

better understanding, but the other two replicas would follow a similar messages exchange.

In addition, we faced a especial situation that could happen in real networks and thus needs to be solved. In case of using geographic coordinates the rendezvous points obtained by hashing functions rarely maps the exact position of a node. Moreover, it could happen that the same node is the closest one to more than one of the hashed positions, thus taking responsibility of more than one replica. This corner case appears more frequently in smaller networks that have few nodes, whereas it is very rare in large networks with hundreds of nodes.

Furthermore, when there are N_r' replicas, they would expect to receive $N_r - 1$ REPLICATION STATISTIC messages (one per each remaining replication node). However, if a situation like the one explained above happens this implementation could face some problems. In order to solve it, a replica includes in the REPLICATION STATISTIC messages how many replication points it is actually representing. In this way, we assure that, when performing the estimation of the number of replicas in the next epoch and the current epoch deadline, all the nodes are operating with the same information.

Finally, when the current epoch expires, old replicas calculate who are the new ones and send a REPLICA INSTANTIATION message that informs the nodes receiving it that they are the replicas for the new epoch. In addition this message contains the information stored in the old replication nodes. REPLICA INSTANTIATION messages must be acknowledged in order to assure that the new replication nodes are set up. Figure 8.4 shows an epoch transition in which old replicas notify the new ones, which reply with a REPLICA INSTANTIATION ACK.

8.5 Implementation of Meta-Information Service

We have also implemented the Meta-Information service, described in Section 7.1.3, which offers a simple mechanism for nodes willing to participate in an application to synchronize with the timeline of that application and start producing or consuming data related to the desired application. In addition, it also serves as a fault-tolerance mechanism in case a node fails and needs to re-synchronize again with its application timeline. For simplicity we have implemented the Meta-Information service using a single static node. This node initially realizes that it is the Meta-Information service node by computing the hash coordinates, so then it processes all the messages related to this service.

It is very important to notice that the Meta-Information Service is jut another application within the network so it also works using PUT and GET messages. Therefore, no new type of messages are required:

- GET: When a consumer or producer of an application are initialized, they first contact the Meta-Information service node. They refer to the Meta-Information Service in the APPLICATION ID field within the framework header, whereas the application they are interested in is included in the message payload.
- GET REPLY: The Meta-Information service node replies to the requests with the current epoch and number of replicas for the requested application.
- PUT: Once the replicas have agreed on the epoch expiration time and the number of replicas for the next epoch, the one with ID 1 in the hash operation sends a PUT message to the Meta-Information service node, indicating the next epoch ID, the number of replicas in the next epoch and the time in seconds until the current epoch expires (similarly as it is done with consumers and producers). Therefore, the Meta-Information Service node sets up a timer that expires at the end of the current epoch, until that moment it still serves the information for the current epoch, after it will start serving the information of the next epoch.
- PUT ACK: The PUT message needs to be acknowledged because the information that it contains should not be lost, otherwise new nodes will not be able to synchronize with the applications.

Figure 8.5 shows an example of the messages exchanged when producers and consumers

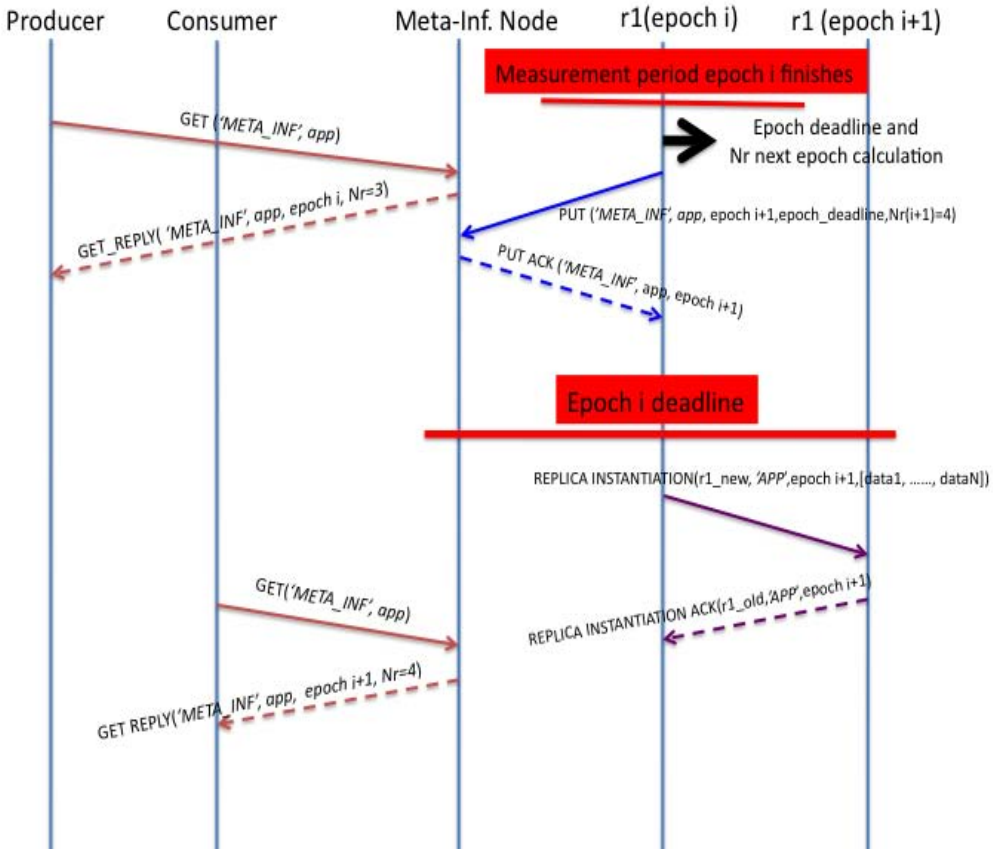


Figure 8.5: *Example of Meta-Information Service functionality.*

use the Meta-Information Service, as well as how the Meta-Information Service updates the information of a particular application.

We have tested the Meta-Information service implementation, and all nodes starting in the network are able to operate after using the Meta-Information service. We have checked both cases nodes that start and nodes that were disconnected and connected again when the network had been running for some time. In both cases, the Meta-Information service allows the nodes to operate with the desired application.

The Meta-Information Service node instantiates an application after receiving the first GET/PUT message containing an APP ID that had not been previously instantiated. When that happens it sends a REPLICA INSTANTIATION message to all the suitable nodes using a pre-defined number of initial replication nodes. From that moment, that particular application is instantiated and starts its normal operation.

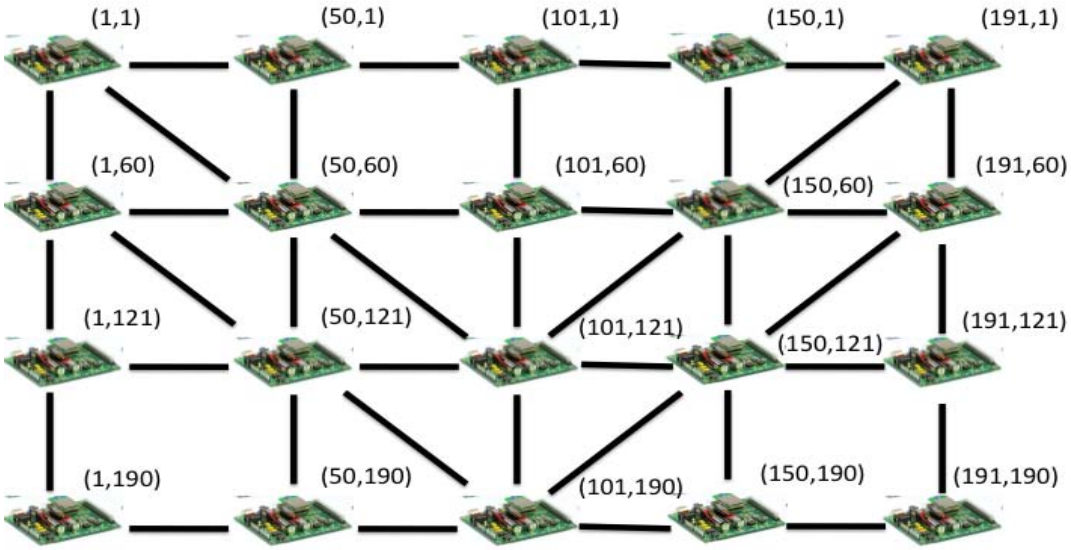


Figure 8.6: *Real testbed using 20 motes emulating a 200x200 square meter network in a two dimensions plane*

.

8.6 Implementation of Multi-Application Framework

Our last step in the implementation was to create a multi-application framework. Since, the Jennic motes do not support on-demand memory allocation, we had to pre-define the maximum number of applications running on the network, and pre-allocate memory in each sensor to store the data events in case it becomes a replica of one of these applications. In addition, each node has to store its 'role' for each of the different applications.

The major complexity was the case in which a node was participating in several applications since it had to keep state of each application in order to correctly update the information. For instance, in case a node is replica for two different applications it needs to store two different timers for the epoch deadline of each application.

Even with these limitations, we have managed to run 4 applications on parallel, since we only have 20 motes, but we strongly believe that in a larger network more applications could be run in parallel without problems.

8.7 Performance Evaluation

We have considered three different aspects in our testbed evaluation: (i) we have checked that the optimal number of replicas provided by Eq. 6.1 is useful in a real implementation, (ii) we have evaluated that changing the replicas over the time balances the energy consumption, and (iii) we have evaluated that changing the replicas over the time also extends the sensornet lifetime.

In all cases we have used a scenario with N=20 motes located in a4 rows by 5 columns grid emulating an area A=200x200 m^2 network. Each mote was programmed with its own coordinates and its neighbours coordinates in order to easily limit its connectivity. Morover, we did not enable a full-mesh topology, but a more irregular one in order to create longer communications paths within the network.

Figure 8.6 shows the testbed used to evaluate our framework implementation. It must be noted that our framework is focused in large networks, for instance our network traffic model based on distance and assumes an infinite plane, but in this real scenario we are working with a small network with just 20 sensors.

Our performance metric was the overall number of messages sent and received within the network[1]. We demonstrate in this section that the main contributions obtained from the simulations of large WSANs are also valid in a small testbed.

8.7.1 Optimal Number of Replicas

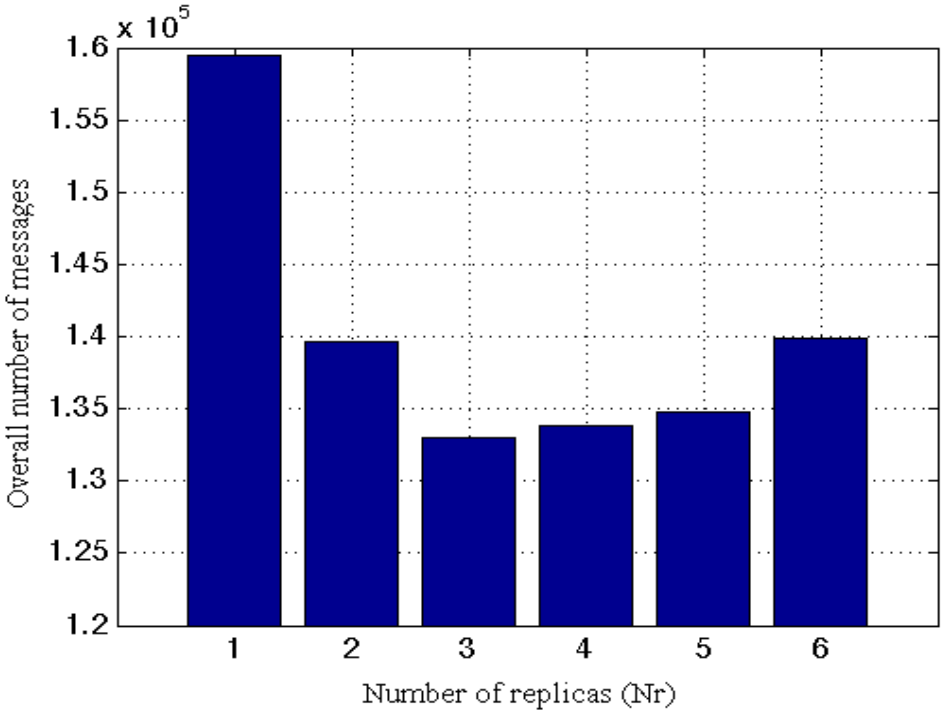


Figure 8.7: *Overall number of messages when using from 1 to 6 replicas in a Dynamic Random Multi-Replication testbed scenario*

.

In this test we always use the same number of replicas in the scenario. It does not matter what are the traffic statistics collected by the replication nodes, we always force the

[1]Jennic 5139 and 5121 motes do not allow measuring the remaining battery in the mote, so we could not use energy to directly analyze the network performance.

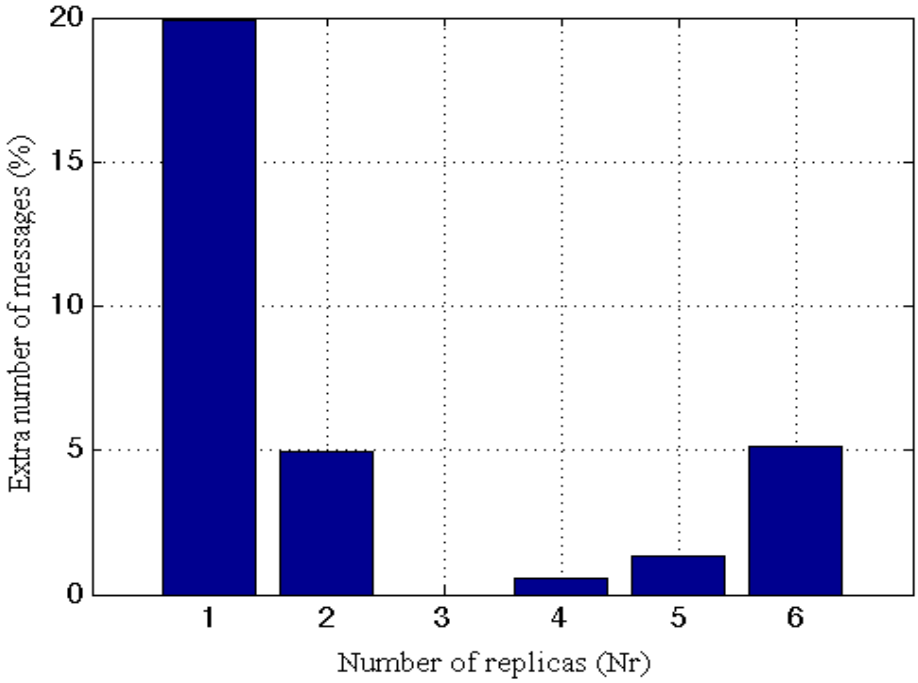


Figure 8.8: *Extra number of messages with respect to the best value for the number of replicas that is 3 in a Dynamic Random Multi-Replication testbed scenario*

.

same number of replicas. We have tested the cases of using from 1 to 6 replication nodes and measured the total number of messages (sent and received) to account all the traffic generated in the network, which is also valid as a rough estimation of the network's energy expenditure. We have used a single application in the test, which was run for 3 hours. All the nodes were producers and consumers at the same time, however the consumption and production rates were different. The nodes generated a production event every 45 seconds, while they generated a query every 15 seconds. Following, the formula of Equation 6.1, the optimal number of replicas should be 4.2. Figure 8.7 shows the overall number of messages in the network when forcing the framework to use 1, 2, 3, 4, 5 and 6 replicas. The number of replicas that actually minimizes the overall traffic is 3.

Figure 8.8 shows the extra traffic generated when using a number of replicas different than 3. Then if 4 replicas are selected, which is the value chosen by our model, only a 0.5% extra traffic is generated. The other close value to the one obtained from the model is 5 replicas. In this case, the overhead traffic is an 1%. Finally, by choosing those values far from the one provided by our model the extra traffic grows up to a 5% for 2 and 6 replication nodes and up to a 20% when a single replication node is selected. This demonstrates in a real testbed that always using a single replication node as proposed in the seminal DCS work [SRK+03] could generate a lot of extra traffic.

This test demonstrates that, even if we are far from the model assumptions (i.e. infinite

field, distance-based model instead of hop-based model, etc), the optimal number of replicas that the model provides is still giving good results in terms of minimizing the network traffic (thus it also reduces the network energy consumption).

8.7.2 Balancing Energy Consumption

In this second test we want to verify whether changing the replication nodes over the time balances the energy consumption per node, even though it generates extra messages. For that purpose we compare a static scenario with dynamic ones, which were configured with different message thresholds to change the epoch. We remember that the message threshold is the maximum number of messages that a node can send and receive while playing the role of a replica.

Again the test ran for 3 hours and all the nodes were consumers and producers at the same time. The production rate was 1 message every 45 seconds and the consumption rate was 1 message every 15 seconds. As in the previous example this leads to an optimal number of replicas of 4.2, so we decided to use 4 replicas in the static case, whereas the dynamic case also started with 4 replicas. After the first epoch, our dynamic implementation autonomously computed the number of replicas in the next epoch based on the traffic statistic captured by the replicas during the measurement period, which lasts 1 minute. It must be noted that usually the selected number of replicas was 4, but sometimes one node was taking the responsibility of two hashed locations, therefore in these cases the real number of replicas was actually 3. Finally, we have used different values for the message threshold in order to manage shorter and longer epochs. Those particular values were 240 and 360 messages respectively. We remember that shorter epochs are expected to provide a better fairness at the price of a higher overhead since more messages are generated due to more frequent epoch transitions.

Hence, in order to compute the network fairness we have used the Jain's Fairness Index (FI) [JCH84] over the number of messages sent and received by each node. The FI provides a value between 0 (lowest fairness) and 1(highest fairness). In addition, we also take into account the extra number of messages generated by the dynamic scenarios in comparison with the static case.

	Static	**Dynamic-240**	**Dynamic-360**
Fairness Index	0.59	0.83	0.81
Overhead (%)	-	10.53	7.04

Table 8.1: *Evaluation of energy distribution in the testbed. Fairness Index (FI) of messages sent and received by network nodes and extra number of messages generated in the dynamic scenarios compared to the static scenario.*

Table 8.1 shows the expected results. On the one hand, changing the replication set over the time leads to a much more fair energy distribution in the network than using static replicas. In addition, as expected, the shorter the epoch (using a lower message threshold) the better the energy distribution. That is the reason why the scenario with a 240 message

threshold presents a better FI than the one with 360 messages. On the other hand, changing the replication nodes over the time leads to some traffic overhead, that in our test was 10% for the dynamic scenario with the shorter epoch (240 messages' threshold) and a 7% in case of using a longer epoch (360 messages' threshold).

Moreover, it must be noted that we have used a quite low message thresholds to generate quite a lot epoch changes to better appreciate a distribution of the energy consumption within the network. For instance, using a 240 message threshold in a 3 hours testbed generates 30 epochs (i.e. 9-10 changes per hour), while a 360 message threshold leads to 20 epochs (i.e. 6-7 epochs per hour) .

Therefore we have confirmed with our testbed that changing the set of replication nodes over the time leads to a much more fair energy consumption distribution within the network. It is also true that it creates some traffic overhead (although, even in cases with short epochs, this is less than 10%). However, as we have shown in Section 7.3, in the long term the routing holes created in the static scenario lead to a traffic overhead (due to the use of longer paths) that overpass the overhead due to the extra traffic generated for changing the replication set over the time. However, we were very limited in our testbed since it was only formed by 20 motes and suffering disconnection due to battery expiration would generate useless scenarios.

8.7.3 First sensor dead lifetime

Due to the reduced number of sensors we were using, and due the fact that Jennic 5139 and 5121 motes do not allow to monitor the remaining battery in a node, we decided to emulate the sensor battery lifetime with a limit to the maximum number of messages sent and received by the nodes. For that purpose we have evaluated: a static scenario, a dynamic one with an epoch message threshold of 240 and another dynamic one with a threshold of 360 messages. For each of these scenarios we measured the time when the first node reached 5000, 6000, 7000, 8000, 9000 and 10000 messages in order to demonstrate that changing the replication set over the time allows an effective lifetime extension.

Lifetime (# msg.)	**Static (sec)**	**Dynamic 240 (sec)**	**Dynamic 360 (sec)**	**Dynamic 240 improv. (%)**	**Dynamic 360 improv. (%)**
5000	3054	4199	3506	37%	15%
6000	3659	5142	4556	41%	25%
7000	4242	6268	5456	48%	29%
8000	4830	7203	6301	49%	30%
9000	5419	8362	7187	54%	33%
10000	6003	8945	7863	49%	31%

Table 8.2: *Evaluation of testbed lifetime extension when adopting the dynamic solution instead of the static one. Times when the first node reaches different message thresholds.*

Table 8.2 shows the effective time extension in percentage of the dynamic solution com-

pared to the static one. The first conclusion is that changing the replication set over the time truly reduces the load of the most saturated node in the network. This is translated into a longer period to reach the messages limit, which implies a longer lifetime before running out of battery. In particular, when analyzing the dynamic scenario with an epoch message threshold of 240, we check that in all cases the time extension is over 35% even overpassing 50% when the messages limit is established in 9000 messages. The time extension for an epoch message threshold of 360 is reduced to values between 15% and 33% depending on the different messages limits. That happens because a lower message threshold implies a shorter epoch, thus a better distribution of energy per node. This reduces the number of messages in the node with more messages, at the price of increasing the overhead in the network.

Finally, it must be noted that the trend in the results is the higher the limit of messages, the longer the time difference between the static and the dynamic solutions. That means that if we were able to run very long testbeds (e.g. 1000 epochs) like the ones evaluated in the simulations (see Section 7.3) the lifetime extension would be much higher, as Table 7.1 shows for the simulation results.

8.8 Final Implementation remarks

First of all, it has been demonstrated that the proposed Dynamic Random Multi-Replication DCS framework is feasible, and it can be implemented in real motes. Still, some implementation hurdles need to be solved first.

The main problems that we found with the employed devices (Jennic 5139 and 5121 motes) were: (i) they do not allow on-demand memory allocation. It was solved by pre-allocating all the potentially needed memory during the node initialization phase, which was the only possible solution, even though nodes were not using that memory most of the time. (ii) They do not allow float number operations. This was specially problematic when calculating the optimal number of replicas in the next epoch. But this could be easily solved by multiplying by 10 the obtained values. For instance in a case where the optimal number of replicas was the integer division 28/10=2.8, that was rounded to 2, but if we multiply the numerator by 10 and then perform the integer division 280/10=28, we could later check if the was closer to 20 or 30, in order to calculate better the number of replicas to be used.

Finally, it must be noted that some minor de-synchronization among the framework participant nodes could occur since the messages containing synchronization values (e.g. PUT ACK) take some time to travel from the source to the destination node. However, these delays are usually in the order of hundreds of milliseconds. Thus if nodes production and/or consumption rates are above that threshold, the de-synchronization does not have any practical impact on epoch transitions.

Chapter 9

Adaptation to Spatially Heterogeneous Traffic Distributions

9.1 Spatially Heterogeneous Traffic discussion

All the proposals presented in the Data Centric Storage (DCS) literature assume that the consumption queries and production events are homogeneously distributed within the network. All analytical traffic models have been developed based on that assumption. In addition, all the structured replication mechanism take advantage of this assumption. Otherwise, it would not make sense to spread replication nodes all over the network when all the traffic is concentrated just in a corner of the network. For instance, let us assume a mutli-replication grid-structured DCS network with 16 replication nodes in a square area, but all consumption queries and production events are concentrated in just 1/4 of the network, let us say the south-east part. On the one hand, if we think in the case where Consumption-dominates-Production, production events will be forwarded to replicas in the north-east, north-west and south-west areas where there are no nodes to consume that information. Therefore, a lot of resources are wasted in useless replication. On the other hand, if we think in the case where Production-dominates-Consumption all the queries are forwarded to the no producers areas, and again a lot of energy is wasted in a useless operation.

Next, let us analyse the heterogeneous Consumption-dominates-Production case in more detail. On the one hand, the production traffic is seen as global by the replicas, since all replication nodes receive all production events. Therefore, it does not matter where producers are located, since the highest cost in production is to replicate their events across the replication tree. Then, if replicas are close to them, the produced events use short paths to reach the replica but long ones in the replication tree. Therefore, it does not make a big difference if the replicas are concentrated in a particular area, because in that case the production events will use longer paths until they reach the replication node, but then the replication tree will be shorter. On the other hand, consumers, whose query intensity overpass the production event intensity, are interested on having replication nodes as close as possible, and in case

all the consumption intensity is concentrated in a particular area, it is better to place there most of the replicas. Otherwise there will be some replicas very saturated whereas the others are useless since no consumer queries them. Therefore, replication nodes are more worthy if they are allocated in the area with high consumption query intensity.

If we think in the opposite case, Production-dominates-Consumption with spatially heterogeneous traffic, we could make the reasoning in a symmetric way. Now consumption queries have to reach all the replication nodes by using the replication tree. Therefore, if there are replicas that do not have any attached producer, a lot of resources and energy are wasted in order to send the query to that region. However in case that the replication nodes are deployed in the high-intensity production area, even if some consumers are far away from a replication node (located in the area with most of the production) that would not mean a big increment in the consumption cost since their queries are already going to that area anyway. In addition, most of the replicas are close to the producers (whose events intensity overpass the consumption queries) reducing a lot the production cost and alleviating the possible saturation of replicas.

Therefore, intuitively it seems a good idea to allocate replication nodes in those network areas: (i) presenting more consumption traffic when Consumption-dominates-Production, or (ii) presenting more production traffic when Production-dominates-Consumption. That would be feasible if production and consumption patterns were well known before deploying the WSAN. However, it becomes a complex problem in case those patterns are unknown, and even more complex if the consumption and production distributions also change over the time, since the replicas allocation will need to adapt to the moving traffic distributions.

In this chapter we propose an heuristic that adapts the Dynamic Random Multi-Replication DCS framework to heterogeneous consumption and production traffic distributions. In addition, it also adapts to changes over the time, since it re-evaluates the consumption and production traffic conditions in each epoch. We demonstrate that applying the proposed heuristic leads to better results in terms of traffic reduction and nodes' lifetime extension. In addition, the proposed heuristic takes advantage of the protocol and operations already defined for our framework, so that the new solution introduces very few overhead and low complexity. We strongly believe that our implementation could be easily extended with this heuristic. To the best of our knowledge this is the first DCS proposal that is adaptive to heterogeneous network consumption queries and production events distributions.

9.2 Heuristic to adapt the DCS framework to heterogeneous traffic

We have presented a framework designed for scenarios where consumption and production traffic are roughly homogeneously distributed within the network. However, in this section we consider a different scenario where production and/or consumption intensities present a spatially-heterogeneous distribution across the network.

The rationale of this heuristic is that the total number of replicas could still be computed

based on the global consumption and production traffic, but then the set of replicas could be placed in the areas with higher traffic intensities. Then, our Random Multi-Replication framework could be extended for this new scenario, by using the measurement period in which all replicas exchange local traffic information. That information could be used to spawn new replicas in those areas with more traffic and prune those replication nodes that are located in areas where there is little or none traffic.

In particular, in this chapter we consider the case in which Consumption-dominates-Production (it would be similar for the opposite case). In this case the production traffic is global since each production event is received by all the replication nodes. However, consumption traffic remains local between consumers and its closest replica. When the replication nodes exchange traffic statistics they are able to know what is the consumption traffic of each replication node. Therefore, each node could compute how many replicas should be assigned to each current replica depending on the measured consumption traffic.

Next, we present an example for better understanding. Let us assume a scenario with 5 replicas in the current epoch, which have exchanged their accounted consumption and production traffic during the measurement period. Then each replica receives the following percentage of total consumption traffic:

- Replica 1 $\rightarrow$ 0%
- Replica 2 $\rightarrow$ 10%
- Replica 3 $\rightarrow$ 40%
- Replica 4 $\rightarrow$ 0%
- Replica 5 $\rightarrow$ 50%

Under this traffic pattern we could think in a new distribution for the 5 replicas in the current epoch:

- Replicas 1: It does not perceive any consumption traffic, therefore it must be pruned.
- Replica 2: It receives a 10% of the overall consumption traffic. That means that its area of influence should be assigned with 0.5 replicas. If we round that number, we obtain 1 replica in that area. Since replica 2 is already placed there, no further action is taken.
- Replica 3: It receives a 40% of the total consumption traffic. That means that the area under the responsibility of replica 3 should be assigned with $0.4 * 5 = 2$ replicas. Therefore, a new replica should be spawned in that area.
- Replica 4: It does not perceive any consumption traffic therefore it must be pruned.
- Replica 5: It receives half of the total consumption traffic. Therefore, half of the total number of replicas should be assigned to its Voronoi cell, which is 2.5 replicas. That number is rounded, and 3 replicas are then assigned to that area. Hence, two new replicas need to be generated in the area of replica 5.

It must be noted than in the example above, we have decided to round the deserved number of replicas, that means that if a node is being assigned with a value ≥ 0.5 replicas we decide to assign 1 replica (in the case of 3.5 we assign 4 replicas), thus if the value is < 0.5 we assign 0 replicas. In many cases, the number of replicas after the heterogeneous adaptation could be different to the initial value (e.g. in the previous example after the adaptation there would be 6 replicas in the scenario instead of 5). A more sophisticated algorithm could be used in order to better fit the initial number of replicas.

Now, a mechanism to spawn new replicas within the area of influence of an existing replica is needed. A requirement for this algorithm is that each replica must be able to calculate the position of all the new replicas locally, as well as, to know which replicas must be pruned. First of all, each of the initial replicas receive the local consumption traffic from all the other replication nodes. Therefore, they can easily compute which is the new number of replicas that must be assigned to each area, as well as which replication nodes need to be pruned. Then, we propose an easy mechanism to assure that the new spawned replicas are located in the regions with higher traffic. In the above example, replica 3 has to spawn a new replica within its area of responsibility. For that purpose new hash locations are generated until the output coordinates are closer to replica 3 than to any other of the initial replicas. This guarantees that the new replica is located in the suitable network zone. Therefore, if the first 5 replicas were generated using $hash(app \oplus epoch \oplus i) \ \ \forall i \in [1, 5]$, to generate the new locations we use the same hash function, starting from $i = 6$, and keep increasing i until a suitable location for the spawned replica is found. This mechanism also fulfils the requirement that all replication nodes can locally calculate the new replication nodes adapted to the heterogeneous consumption distribution, since all have the same information and calculate the same has operations in the same order

Therefore, by just using some few extra messages (those that notify the new spawned replicas its new role), we obtain an adaptive algorithm to heterogeneous conditions. However, consumers and producers need to be notified with the location of the new replicas because now its closest replica could be one of the spawned ones, or it was one of the pruned replicas. In the homogeneous framework, producers and consumers are notified with the duration of the current epoch and the number of replicas in the next epoch after the measurement period. Then, that notification could be also used to re-direct the suitable consumers and producers to those new spawned replicas that now are the closest ones to them. Therefore, redirecting consumers and producers to the new spawned replicas should not add any overhead.

Therefore, this simple heuristic extends our dynamic multi-replication DCS framework to adapt the replication nodes allocation to the heterogeneous traffic distribution in each epoch. This heuristic produces little overhead because only some few extra messages to instantiate the new replicas are generated compared with the standard framework. Moreover, since this algorithm is executed in every epoch, the framework is able to adapt to changes in the traffic distributions both in time and space.

Next, we evaluate this spatial-temporal adaptive solution for scenarios with heterogeneous traffic distribution in front of the original framework assuming homogeneity. We refer to the former as *adaptive* solution and the latter as *non-adaptive* solution.

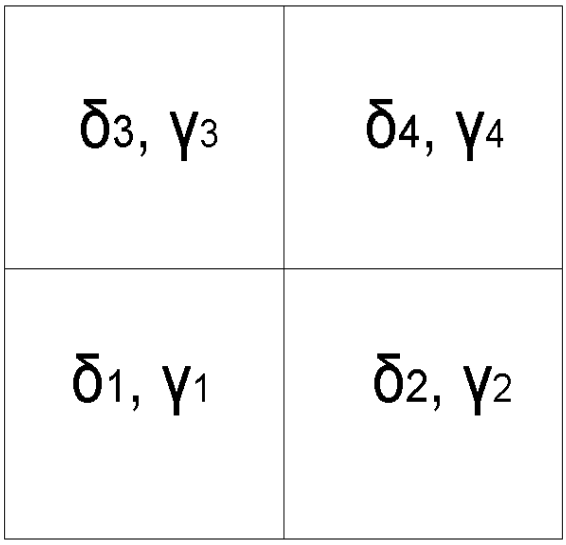


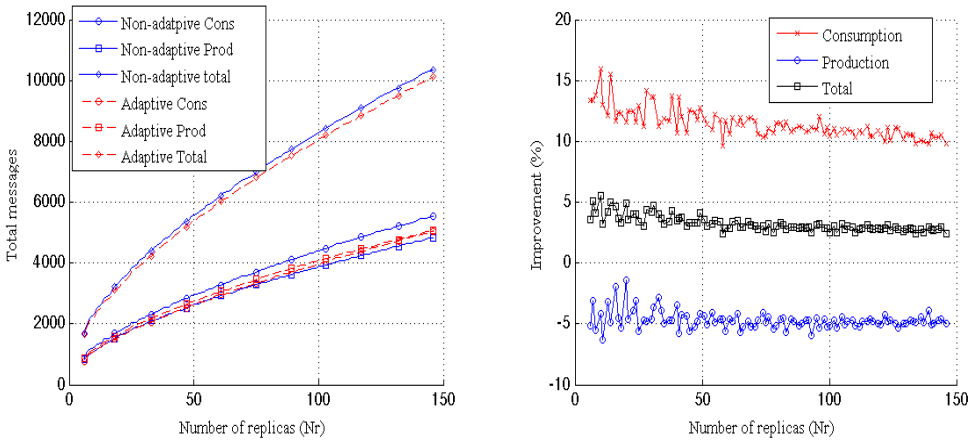
Figure 9.1: *Generation of spatially-heterogeneous traffic*

9.3 Performance Evaluation

This section evaluates the goodness of the adaptive approach in cases of heterogeneous consumption and/or production traffic distributions. For that purpose, we compare the adaptive and non-adaptive solutions in terms of overall network traffic. Thus, we compare both approaches under the same traffic generation pattern, and then get the overall network traffic in one simulation cycle. In addition, we also measure the network lifetime with long term simulations by showing the number of dead nodes over the time for both approaches.

9.3.1 Comparison of overall network traffic

In order to create a heterogeneous traffic distribution, we divide a square sensornet into 4 equal zones and assign a portion of the total consumption queries and production events to each of them as shown in Figure 9.1. Then, we define two different set of values, $\delta = [\delta_1, \delta_2, \delta_3, \delta_4]$ that refers to the portion of consumption traffic associated to each region and $\gamma = [\gamma_1, \gamma_2, \gamma_3, \gamma_4]$, that refers to the intensity of production traffic assigned to each small square. We evaluate via simulation which are the savings of using the adaptive approach instead of the non-adaptive one. It must be noted that in this section we just compare the traffic results in one cycle, that is, when each producer and consumer node generates one event or a query. We consider different heterogeneity degrees in our simulations:



(a) Overall Traffic (b) Improvement of adaptive approach

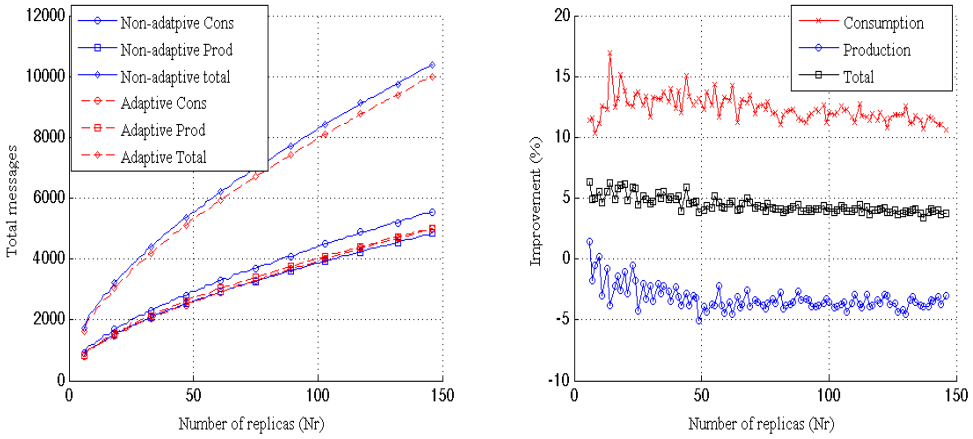
Figure 9.2: *Comparison of adaptive vs. non-adatpive approaches with* $\delta_a = [0.25, 0.25, 0.25, 0.25]$ *(A=1000x1000* m^2*, N=5000 nodes, Tx=50 m.)*

- $\delta_a = [0.25, 0.25, 0.25, 0.25]$ (see Figure 9.2)
- $\delta_b = [0.2, 0.2, 0.4, 0.2]$ (see Figure 9.3)
- $\delta_c = [0.15, 0.15, 0.55, 0.15]$ (see Figure 9.4)
- $\delta_d = [0.1, 0.1, 0.7, 0.1]$ (see Figure 9.5)
- $\delta_e = [0.05, 0.05, 0.85, 0.05]$ (see Figure 9.6)
- $\delta_f = [0, 0, 1, 0]$ (see Figure 9.7)

The first case actually refers to an homogeneous consumption traffic distribution since all the zones are assigned with the same portion of consumption traffic. Later we increase the heterogeneity degree until the case in which all the consumption traffic is concentrated in one of the four quadrants. In addition, we keep the production intensity homogeneous across the network with $\gamma = [0.25, 0.25, 0.25, 0.25]$.

We simulate an scenario with an area A=1000x1000 m^2, N=5000 nodes, Tx=50 m, a λ_c/λ_p ratio variation from 4 to 100, measuring the overall traffic with the overall number of hops generated in the network. We have run 50 iterations for each ratio value and represent the average overall network traffic (confidence intervals are not shown since they would be undistinguishable).

In order to run the simulation we consider periodic traffic generation by each consumer and each producer. That is, in one cycle (time unit) every consumer sends one query (and receives the reply) and every producer generates one event. Therefore, we get the different λ_c/λ_p ratios by keeping constant the number of producers and increasing the number of consumers.



(a) Overall Traffic (b) Improvement of adaptive approach

Figure 9.3: *Comparison of adaptive vs. non-adatpive approaches with* $\delta_b = [0.2, 0.2, 0.4, 0.2]$ *(A=1000x1000* m^2*, N=5000 nodes, Tx=50 m.)*

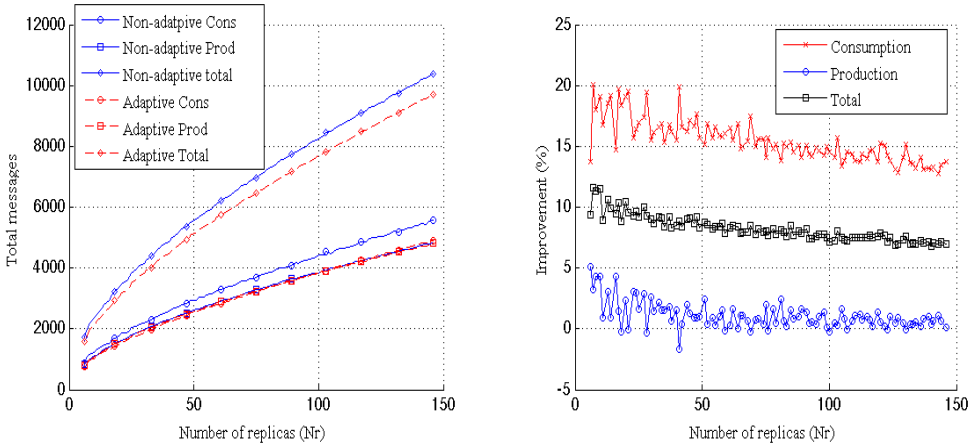
All the graphs show in the X axis the optimal number of replicas (N_r) provided by the model for a particular ratio λ_c/λ_p, and in the Y axis: subfigure (a), the consumption, production and overall traffic for each mechanism measured in number of messages, and subfigure (b), the relative improvement for consumption, production and overall traffic, when the adaptive approach is applied.

First of all, it must be highlighted that even in the homogeneous traffic scenario (δ_a), the adaptive proposal outperforms the results of the original solution. The reason is that even in an homogeneous deployment, since the replicas are located at random, some of them could receive a higher consumption rate while others fewer or none traffic (e.g. replicas located at the borders of the network or close together). In such cases, applying the adaptive solution allows to prune useless replicas and/or spawn new ones in those areas experiencing more consumption load.

In addition, when the spatial consumption distribution becomes more and more heterogeneous the adaptive approach provides a more relevant improvement for the overall network traffic, reaching peaks of a 60% in the most extreme case (δ_f).

Other conclusion is that in medium/high heterogeneity cases, the improvement is better for a low number of replicas and decreases when the number of replicas grows up. This happens because the replication is more sensitive to the cases with less number of replicas, because a bad allocation of these few replicas leads to a very bad results, whereas if these few replicas are located taking into account the consumption distribution the overall traffic is reduced a lot.

Table 9.1 shows the average number of hops for consumption queries and production events (that also include the replication tree). The results shown in the table explain the reason of the traffic reduction when applying the adaptive solution. Consumers' communication paths are much shorter when applying the adaptive solution to heterogeneous traffic condi-



(a) Overall Traffic (b) Improvement of adaptive approach

Figure 9.4: *Comparison of adaptive vs. non-adatpive approaches with* $\delta_c = [0.15, 0.15, 0.55, 0.15]$ *(A=1000x1000* m^2*, N=5000 nodes, Tx=50 m.)*

δ	**Consumption Queries Avg. Path Length Non-Adaptive — Adaptive**	**Production Events Avg. Path Length Non-Adaptive — Adaptive**	**Consumption Difference**	**Production Difference**
δ_a	5.63 — 5.11	27.08 — 29.23	9.81%	-6.72%
δ_b	5.72 — 5.22	27.33 — 28.36	9.5 %	-3.63%
δ_c	5.95 — 5.23	27.77 — 25.94	13.74%	7.07%
δ_d	5.77 — 5.15	28.07 — 24.31	12.1 %	15.47%
δ_e	5.57 — 4.39	27.29 — 21.90	27.3 %	24.63%
δ_f	6.79 — 4.85	28.6 — 24.77	40.06%	15.46%

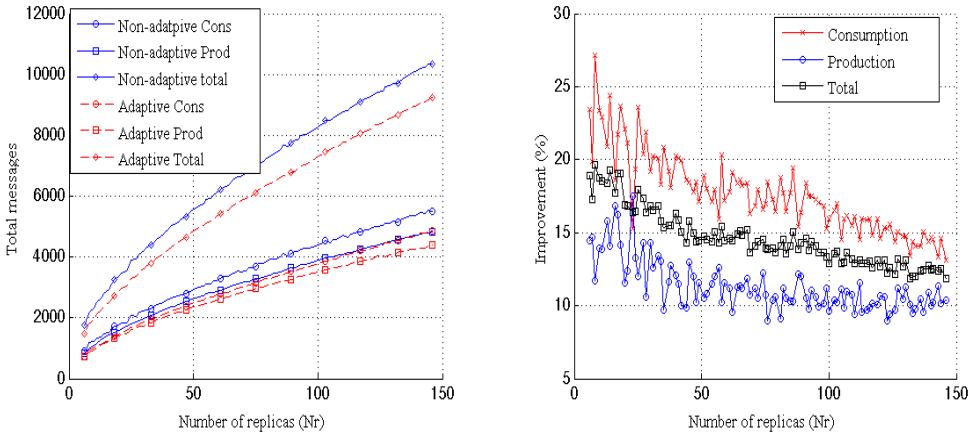
Table 9.1: *Consumption query and Production event path lengths in number of hops for adaptive and non-adaptive approaches.*

tions. In particular, the higher the consumption heterogeneity, the larger the improvement when applying our adaptive proposal. This occurs because we are moving the replication nodes near the consumers. Although the adaptive solution does not pay attention to the producers in order to re-allocate the replication nodes, in medium/high heterogeneity scenarios the production paths are also shorter. However, in low heterogeneity scenarios production paths are slightly shorter in the non-adaptive approach, but still the overall traffic present a clear reduction.

9.3.2 Network Lifetime

Additionally, we compare the adaptive solution to the non-adaptive one in terms of network lifetime when there is a heterogeneous traffic distribution within the network. For that purpose, we have measured the number of dead nodes over the time for both approaches when utilizing the same traffic pattern.

We have first evaluated a static traffic scenario in which the spatially-heterogeneous



(a) Overall Traffic (b) Improvement of adaptive approach

Figure 9.5: *Comparison of adaptive vs. non-adatpive approaches with* $\delta_d = [0.1, 0.1, 0.7, 0.1]$ *(A=1000x1000* m^2*, N=5000 nodes, Tx=50 m.).*

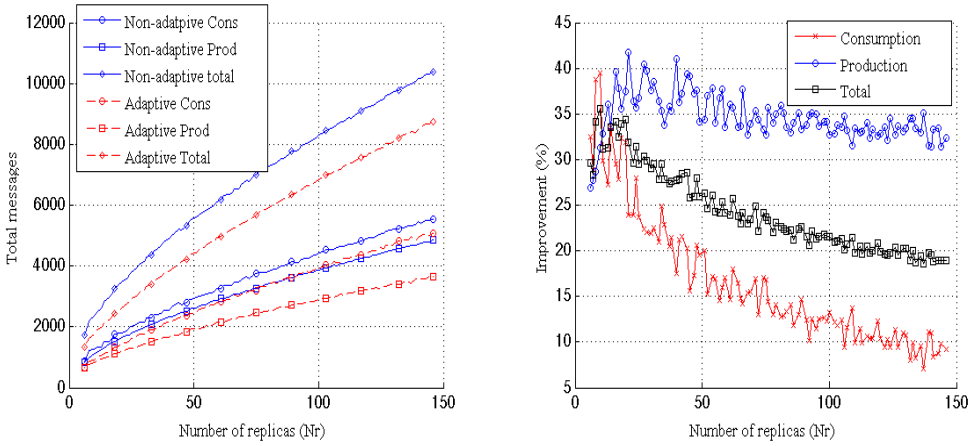
traffic pattern remains constant across the whole simulation. In this case, the evolution of dead sensors is quite similar in both approaches because, if most of the traffic is always localized in a particular area of the network, the nodes in that area run out of battery quicker than the rest of the nodes anyway. It does not matter if we move the replicas' location to that region, it still is the area with more traffic and the nodes there will be the ones exhausting their batteries earlier.

Therefore, what is interesting is to compare both solutions when the spatial-heterogeneous traffic is dynamic, that is, the zone with heavier consumption and/or production intensities changes over the time. In this case, being adaptive to heterogeneous traffic distribution has a noticeable impact on extending the sensors lifetime.

We have run a long-term simulation under different heterogeneity conditions. In all the cases we simulate a sensornet with a grid sensor deployment of area A=500x500 m^2, N=2500 nodes with a transmission range Tx=30 m. We deploy 300 consumers and 100 producers. The epoch message threshold that starts an epoch transition is 200000 messages. The measurement period at the beginning of an epoch to get traffic statistics lasts 10 cycles. We generate dynamic consumption traffic by changing the consumers location every 5000 cycles. The simulation duration is 150000 cycles. Finally, it must be noted that a sensor runs out of battery after 1 million messages (sent+received).

For the consumption traffic we have used the heterogeneity distributions defined before as δ_a ([0.25,0.25,0.25,0.25]), δ_c ([0.15,0.15,0.55,0.15]), δ_d ([0.1,0.1,0.7,0.1]), δ_e ([0.05,0.05,0.85,0.05]]) and δ_f([0,0,1,0]), but randomly selecting one of the sectors to apply the heavier traffic portion every 5000 simulation cycles. As in the previous case, we have simulated and homogeneous distribution for the production traffic (γ =[0.25,0.25,0.25,0.25]).

Figures 9.8, 9.9, 9.10, 9.11 and 9.12 show the number of dead sensors over the time for

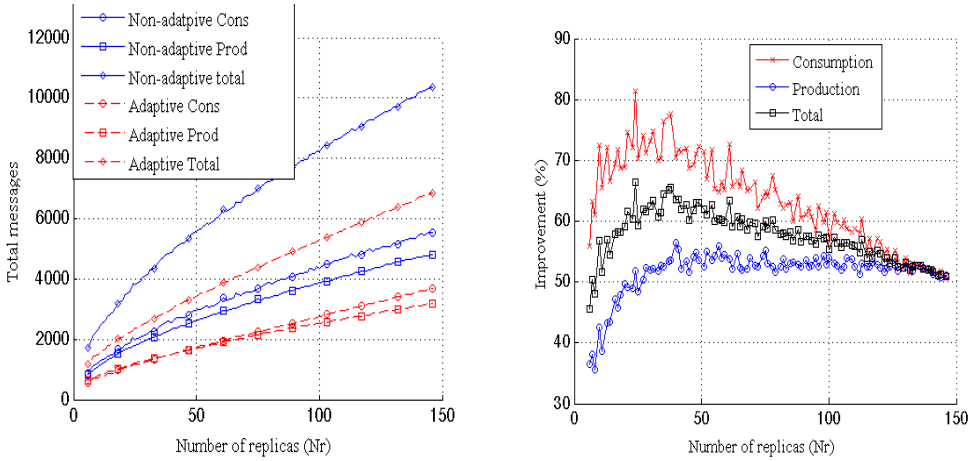


(a) Overall Traffic (b) Improvement of adaptive approach

Figure 9.6: *Comparison of adaptive vs. non-adatpive approaches with* $\delta_e = [0.05, 0.05, 0.85, 0.05]$ *(A=1000x1000* m^2*, N=5000 nodes, Tx=50 m.)*

adaptive and non-adaptive solutions for δ_a, δ_c, δ_d, δ_e and δ_f, respectively. The first conclusion is that the adaptive approach allows sensor to extend their lifetime even when it is applied in a scenario with an homogeneous traffic pattern (see Figure 9.8). In addition, the higher the heterogeneity, the lesser number of sensors die when using the adaptive solution. Conversely, the non-adaptive solution behaves worse under high heterogeneous traffic patterns. Therefore, the adaptive solution presents better performance with higher consumption heterogeneity, and it even extends the nodes lifetime under homogeneous conditions.

Finally, we want to mention that similar results to those obtained by a single application with dynamic traffic scenario would be obtained in the case when multiple static applications with spatially-heterogeneous traffic were running over a WSAN.



(a) Overall Traffic (b) Improvement of adaptive approach

Figure 9.7: *Comparison of adaptive vs. non-adatpive approaches with* $\delta_f = [0, 0, 1, 0]$ *(A=1000x1000* m^2*, N=5000 nodes, Tx=50 m.).*

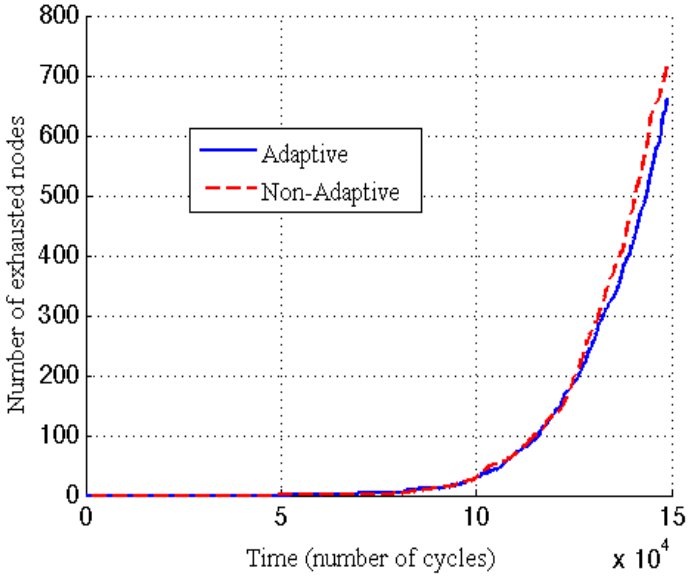


Figure 9.8: *Number of dead sensors over the time when comparing the adaptive and non-adaptive approaches for* δ_a=$[0.25, 0.25, 0.25, 0.25]$ *(A=1000x1000* m^2*, N=2500 nodes, Tx=30 m, Battery=* 10^6 *message,* $E_{th} = 2{*}10^5$ *messages).*

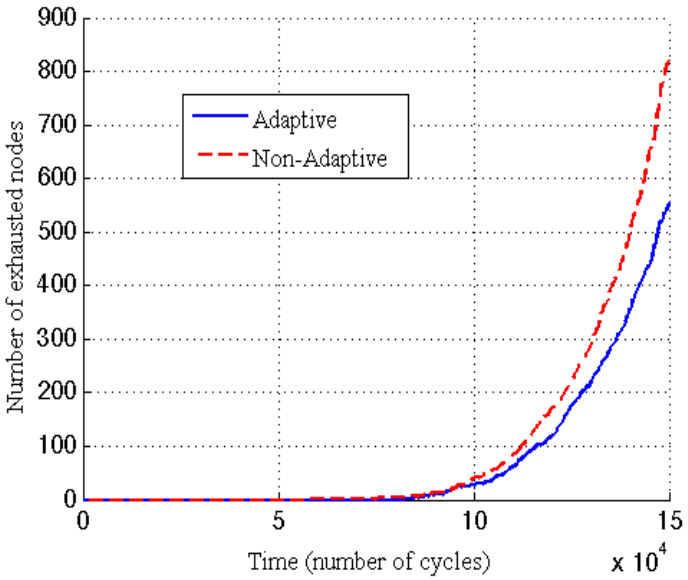


Figure 9.9: *Number of dead sensors over the time when comparing the adaptive and non-adaptive approaches for* δ_c=$[0.15, 0.15, 0.55, 0.15]$ *(A=1000x1000* m^2*, N=2500 nodes, Tx=30 m, Battery=* 10^6 *message,* E_{th} $=$ 2*10^5 *messages).*

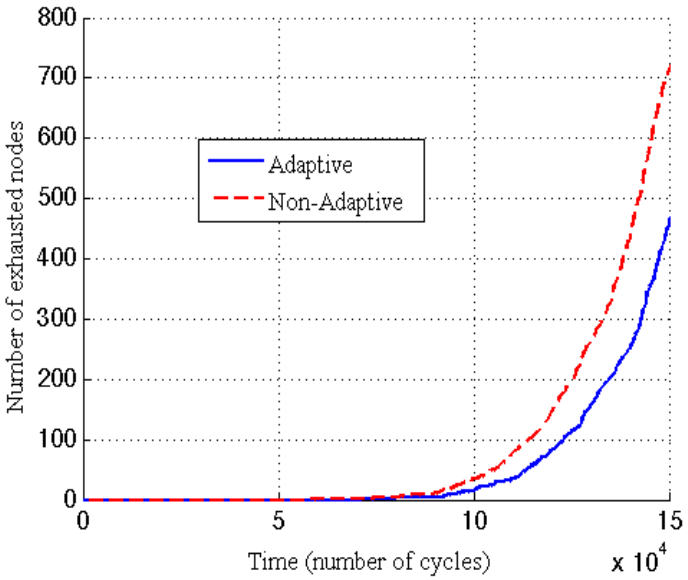


Figure 9.10: *Number of dead sensors over the time when comparing the adaptive and non-adaptive approaches for* δ_d=$[0.1, 0.1, 0.7, 0.1]$ *(A=1000x1000* m^2*, N=2500 nodes, Tx=30 m, Battery=* 10^6 *message,* E_{th} $=$ 2*10^5 *messages).*

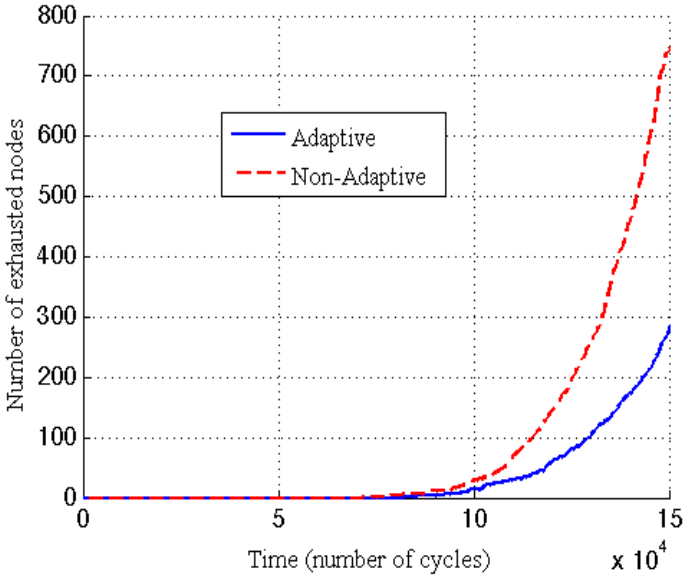


Figure 9.11: *Number of dead sensors over the time when comparing the adaptive and non-adaptive approaches for* δ_e=$[0.05, 0.05, 0.85, 0.05]$ *(A=1000x1000* m^2*, N=2500 nodes, Tx=30 m, Battery=* 10^6 *message,* $E_{th} = 2{*}10^5$ *messages).*

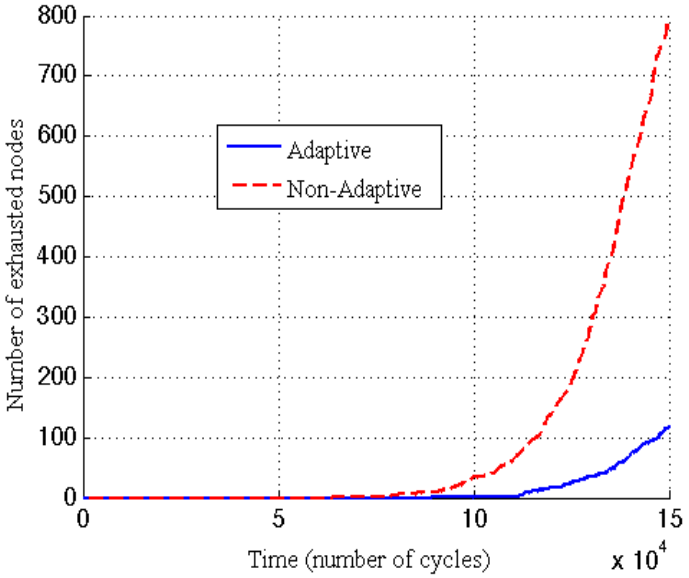


Figure 9.12: *Number of dead sensors over the time when comparing the adaptive and non-adaptive approaches for* δ_f=$[0, 0, 1, 0]$ *(A=1000x1000* m^2*, N=2500 nodes, Tx=30 m, Battery=* 10^6 *message,* $E_{th} = 2{*}10^5$ *messages).*

Part IV

Conclusions, Future Work and Contributions

Chapter 10

Conclusions and Future Work

10.1 Conclusions

First of all, after realizing this PhD Thesis, we strongly believe that Data Centric Storage (DCS) would be a very useful mechanism for Wireless Sensor and Actor Networks (WSANs), although a deep theoretical study, but also aware of practical issues, was necessary.

We realized that using a single home node is not a suitable solution for DCS networks since it creates a hot-spot in the network. Furthermore, those works in the literature proposing multi-replication solutions were not adaptive enough. Then, we have first proposed Quadratic Adaptive Replication (QAR) that improves the adaptivity in front of previous proposals. Furthermore, we have proposed a second Random Multi-Replication mechanism that has been demonstrated to be the best replication DCS scheme proposed so far.

In addition, from the results of this Thesis we can conclude that DCS systems must be adaptive to changing network conditions, being able to tune the number of replication nodes allocated in the network in order to minimize the traffic since this has a direct impact on energy consumption. Therefore, solutions proposing to use a fixed number of static replication nodes are suboptimal and lead to increase the energy consumption within the network.

However, designing such adaptive solution is only possible with a solid theoretical ground that has analyzed the costs of essential DCS operations in order to discover what are the trade-offs that could lead to optimize the solution, in this case by minimizing the overall network traffic. It is very important that those mathematical models are based on measurable parameters that can be really obtained in a distributed way. For instance, it is feasible to measure the network consumption and production intensities in a distributed way with a very low overhead. Furthermore, the mathematical models need to finalize in a simple formula that could be evaluated with low computational effort by a simple device such as a sensor node. In this sense, our two multi-replication proposals, QAR and Random Multi-

Replication, are based on mathematical models that fulfils all the previous requirements: (i) they are based on measurable parameters such as the consumption and production traffic intensities, and (ii) they lead to very simple formulas to compute the optimal number of replicas that minimizes the overall network traffic by the replication nodes.

Furthermore, the obtained results clearly show that a static DCS network is a very unfair system in terms of energy consumption. These systems have a real impact in sensornets, because of the generation of routing holes that, as it has been demonstrated, have an important effect in the network energy consumption, since they lead to use longer communication paths that consume more network resources. Then, one of the main conclusions of this work is that some mechanisms to balance the energy consumption among all the nodes in the network is required. We propose to change the set of replication nodes over the time in order to obtain a much fairer energy distribution within the network. For that purpose, we have designed a network protocol and a set of algorithms that permit to change the replication set over the time. In addition, our proposal keeps all consumer, producer and replication nodes synchronized across the time. For that purpose, we rely on a special application called Meta-Information service that offers bootstrapping and fault-tolerance to the nodes participating in an application, plus a stateless flag mechanism that enables to notify sensitive timeline information. Our solution is totally distributed since it does not rely on any central entity. Finally, it must be noted that the overhead of our solution is demonstrated to be lower than the one generated by the static approaches. The reason is that static solutions tend to generate large routing holes within the network that lead to longer communication paths to overpass those holes. Then the overhead due to those long communication paths is higher that the overhead introduced by our dynamic solution in the long term. Furthermore, the experiments carried out with long-term simulations show that our proposal to balance the energy consumption could extend the network lifetime in more than a 60%, and even reaching lifetime values of up to 10 times compared to a static approach. Further experiments realized in a small testbed also demonstrates that the lifetime of the first node running out of battery is extended over a 30% in most of the cases when our dynamic framework is applied instead of a static DCS solution.

Therefore, based on: (i) a multi-replication proposal that allocates replicas at random locations, (ii) a simple mathematical model that provides the optimal number of replicas in order to reduce the overall traffic, and (iii) a protocol design that allows to change replication nodes over the time at a low cost, we have presented in this Thesis a Spatio-Temporal Adaptive Random Multi-Replication DCS framework that is, to the best of the authors knowledge, the most complete effort done so far in order to bring multi-replication DCS networks closer to practice.

Furthermore, we have demonstrated that our framework can be implemented in commercial sensor motes, which is a further step towards using DCS networks in real applications. It must be noted that although the theoretical work was developed for large WSAN (i.e. our mathematical models assume networks in an infinite plane), the main conclusions obtained from simulation results for large WSANs are also applicable to small networks. For that purpose, we have run experiments whose results show: (i) the optimal number of replicas provided by the model used in our framework also leads to very good results even in a small network. (ii) The energy is effectively balanced when the replication set of nodes changes

over the time. (iii) If we do not change the replication nodes over the time, the most saturated node dies much quicker compared to the case in which the replicas change over the time. Therefore, we conclude that our framework could be also applied in small WSANs.

Finally, we show in this Thesis that the spatially-homogeneous traffic requirement, which is one of the main assumptions in most of DCS proposals, can be also removed by adapting the proposed framework with a simple heuristic, and thus it could be applied to more realistic scenarios. In this case we have presented a heuristic that adapts our framework to applications where consumption query and production events intensities are not homogeneous across the network anymore, but some locality is present. Therefore, by slightly modifying our framework proposal we are able to move the replication nodes to those parts of the network that are experiencing a higher production and/or consumption traffic. This leads to further reduce the overall network traffic under heterogeneous traffic conditions. Furthermore, we also have shown that applying the adaptive heuristic extends the sensornet lifetime when compared to the original, non-adaptive framework.

10.2 Future Work

The first extension to this work is to adapt our implementation for the case when production intensity dominates consumption one, as well as to integrate the adaptive heuristic to spatially-heterogeneous traffic into our code.

In addition, some additional mechanism could be thought for cases in which production and consumption intensities alternate as the dominant one over the time, so that both models could be integrated together to cover all the spectrum of possible traffic conditions. This could be easily done since replicas can compute the total traffic generated by both models. and then choose what is the most appropriate mode for the next epoch.

Furthermore, it would be also interesting to propose some mathematical model that, based on current traffic conditions establishes what would be the optimal epoch duration (or message threshold) in order to further extend the network lifetime. Still the proposed message threshold mechanism seems to have wide operational range.

Finally, we are investigating for how long data could be stored in a multi-replication DCS network before it is overwritten by newer information. If we think in a sensor network with hundreds or thousands of nodes, we could enable a big distributed storage system that in a near future could reach Gigabytes of capacity. For this case, we could employ fixed epoch durations to allow temporal-driven queries, that is, a given consumer could try to obtain information stored some time ago (e.g. 1 week, 1 month, 1 year, etc).

Chapter 11

Main Contributions

Most of the main contributions of this PhD Thesis have been already published in an international conference, as well as in the special issue of a journal devoted to Wireless Sensor Networks:

- *Data Centric Storage Technologies: Analysis and Enhancement*
  A. Cuevas, M. Urueña, R. Romeral and D. Larrabeiti
  Sensors 2010 (Special Issue on Wireless Sensor Network and Its Application in Advanced Computer Science).
  March 2010. 10(4), 3023-3056.
  ISSN:1424-8220
  JCR Indexed

- *Dynamic Random Replication for Data Centric Storage*
  A. Cuevas, M. Urueña and G. Veciana
  ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, (MSWiM' 10)
  October 17-21, 2010 - Bodrum (Turkey)
  BEST PAPER AWARD

The first paper introduces our Quadratic Adaptive Replication (QAR) proposal. In addition, it surveys the main works on Data Centric Storage (DCS) and describes what are the main research challenges.

The second paper introduces the Dynamic Random Multi-Replication DCS framework that is the main contribution of this Thesis. In this paper, we cover the traffic model for random replication as well as a deep evaluation via simulation of the framework. This work was awarded with the Best Paper Award in the ACM MSWiM '10 conference. Selected best papers from this conference are being selected for publication in the Elsevier Ad-Hoc Networks or Elsevier Computer Communications. Therefore, we are preparing a light extended version of our paper for one of those journals.

Furthemore, there is a third paper submitted that has been accepted under major revision:

- *Modelling Data-Aggregation in Multi-Replication Data Centric Storage Systems for Wireless Sensor and Actor Networks*
  A. Cuevas, M. Urueña, R. Cuevas and R. Romeral.
  IET Communications (Special Issue on Distributed Intelligence and Data Fusion for Sensor Systems)
  Accepted for publication under major revision.
  ISSN:1751-8628
  JCR Indexed

In this paper we propose to use our QAR multi-replication DCS proposal as a suitable solution for WSN data aggregation. Furthermore, we present an application taxonomy depending on two main parameters: whether the application supports data aggregation or require all the raw data, and if the consumption traffic dominates the production one or the other way around. We present a number of mathematical models to obtain the optimal number of replicas in all the possible cases.

Finally, we have a fourth paper under preparation that completes the Spatio-Temporal Adaptive Random Multi-Replication DCS framework by including the framework implementation and the adaptive heuristic to heterogeneous traffic conditions. Therefore, this paper will summarize the main contributions presented in this Thesis. We plan to submit this paper to the journal ACM Transactions on Sensor Networks, the most relevant journal of the Wireless Sensor Network community.

Glossary

- DCS: Data Centric Storage
- DHT: Distributed Hash Table
- GHT: Geographic Hash Table
- GPSR: Greedy Perimeter Stateless Routing
- HVGR: Hierarchical Voronoi Graph Routing
- IEEE: Institute of Electrical and Electronic Engineers
- IETF: Internet Engineering Task Force
- MWSN: Multimedia Wireless Sensor Networks
- PRP: Perimeter Refresh Protocol
- QAR: Quadratic Adaptive Replication
- ToW: Tug of War
- UWSN: Unattended Wireless Sensor Networks
- WSN: Wireless Sensor Network
- WSAN: WIreless Sensor and Actor Network

References

[AC09] Michele Albano and Stefano Chessa. Distributed erasure coding in data centric storage for wireless sensor networks. In *Proceedings of IEEE Symposium on Computers and Communications, ISCC' 09*, pages 22–27, 2009.

[ACNP07a] Michele Albano, Stefano Chessa, Francesco Nidito, and Susanna Pelagatti. Data centric storage in non-uniform sensor networks. In *Proceedings of the 2nd International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID 2007)*, Santa Margherita Ligure (Genova), Italy, 2007.

[ACNP07b] Michele Albano, Stefano Chessa, Francesco Nidito, and Susanna Pelagatti. Q-NiGHT: Adding qos to data centric storage in non-uniform sensor networks. In *proceedings of the 8th International Conference on Mobile Data Management, MDM'07*, Mannheim, Germany, 2007.

[AK04] Ian F. Akyildiz and Ismail H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad Hoc Networks*, 2(4):351 – 367, 2004.

[AK06] Joon Ahn and Bhaskar Krishnamachari. 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*, pages 334–343, New York, NY, USA, 2006. ACM.

[AKK04] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, Dec. 2004.

[All05] Zigbee Alliance. Zigbee specification version 1.0, April 2005.

[AMC07] Ian F. Akyildiz, Tommaso Melodia, and Kaushik R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4):921–960, 2007.

[APC06] Mohamed Aly, Kirk Pruhs, and Panos K. Chrysanthis. Kddcs: a load-balanced in-network data-centric storage scheme for sensor networks. In

Proceedings of the 15th ACM international conference on Information and knowledge management, CIKM' 06, pages 317–326, New York, NY, USA, 2006. ACM.

[APM05] Ian F. Akyildiz, Dario Pompili, and Tommaso Melodia. Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257 – 279, 2005.

[APM06] Ian F. Akyildiz, Dario Pompili, and Tommaso Melodia. State-of-the-art in protocol research for underwater acoustic sensor networks. In *Proceedings of the 1st ACM international workshop on Underwater networks, WUWNet '06*, pages 7–16, New York, NY, USA, 2006. ACM.

[ASSC02] Ian. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.

[AY05] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325 – 349, 2005.

[BdV07] Seung Jun Baek and Gustavo de Veciana. Spatial model for energy burden balancing and data fusion in sensor networks detecting bursty events. *IEEE Transactions on Information Theory*, 53(10):3615–29, October 2007.

[BdVS04] Seung Jun Baek, Gustavo de Veciana, and Xun Su. Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation. *IEEE Journal on Selected Areas of Communications*, 22(6):1130–1140, August 2004.

[BMSU99] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications, DIALM' 99*, pages 48–55, New York, NY, USA, 1999. ACM.

[BMSU01] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[BYAH06] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems, SenSys '06*, pages 307–320, New York, NY, USA, 2006. ACM.

[cBB07] François Baccelli and Charles Bordenave. The radial spanning tree of a poisson point process. *Annals of Applied Probability*, 17:305, 2007.

[cBKLZ97] François Baccelli, Maurice Klein, Marc Lebourges, and Sergei Zuyev. Stochastic geometry and architecture of communication networks. *J. Telecommunication Systems*, 7:209–227, 1997.

[cBZ96] François Baccelli and Sergei Zuyev. Poisson-voronoi spanning trees with applications to the optimization of communication networks. *Operations Research*, 47:619–631, 1996.

[CCAC+09] Rubén Cuevas, Albert Cabellos-Aparicio, Ángel Cuevas, Jordi Domingo-Pascual, and Arturo Azcorra. fp2p-hn: A p2p-based route optimization architecture for mobile ip-based community networks. *Computer Networks*, 53(4):528–540, 2009.

[CCCA+10] Rubén Cuevas, Ángel Cuevas, Albert Cabellos-Aparicio, Loránd Jakab, and Carmen Guerrero. A collaborative p2p scheme for nat traversal server discovery based on topological information. *Computer Networks*, 54(12):2071–2085, 2010.

[CCNR06] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, and Antony Rowstron. Virtual ring routing: Network routing inspired by DHTs. In *Proceedings of ACM SIGCOMM Conference, SIGCOMM '06*, pages 351–362, New York, NY, USA, 2006. ACM.

[CCUnL07] Ángel Cuevas, Rubén Cuevas, Manuel Urueña, and David Larrabeiti. A proposal for zigbee cluster interconnection based on zigbee extension devices. In *First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN 07)*, pages 227–238. IFIP, 2007.

[CUnB09] Rubén Cuevas, Manuel Urueña, and Albert Banchs. Routing fairness in chord: Analysis and enhancement. In *Proceedings IEEE International Conference on Computer Communications, INFOCOM '09*, April 2009.

[CUndV10] Ángel Cuevas, Manuel Urueña, and Gustavo de Veciana. 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, pages 393–402, New York, NY, USA, 2010. ACM.

[CUnRL10] Ángel Cuevas, Manuel Urueña, Ricardo Romeral, and David Larrabeiti. Data centric storage technologies: Analysis and enhancement. *Sensors*, 10(4):3023–3056, 2010.

[CvH04] Supriyo Chatterjea, Lodewijk F.W. Hoesel van, and Paul J.M. Havinga. Ailmac: An adaptive, informationcentric and lightweight mac protocol for wireless sensor networks. In *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 381–388. IEEE, 2004.

[DCABM05] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[DD09] Isabel Dietrich and Falko Dressler. On the lifetime of wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(1):1–39, 2009.

[DDHV03] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod Varshney. A witness-based approach for data fusion assurance in wireless sensor networks. In *In Proceedings of the IEEE Global Telecommunications Conference*, pages 1435–1439, 2003.

[DEA06] Ilker Demirkol, Cem Ersoy, and Fatih Alagoz. "mac protocols for wireless sensor networks: a survey". *IEEE Communications Magazine*, 44(4):115–121, April 2006.

[EGHK99] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, MobiCom '99*, pages 263–270, New York, NY, USA, 1999. ACM.

[ERS06] Cheng Tien Ee, Sylvia Ratnasamy, and Scott Shenker. Practical data-centric storage. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation, NSDI'06*, Berkeley, CA, USA, 2006. USENIX Association.

[FJL+97] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking.*, 5(6):784–803, 1997.

[FRZ+05] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng Tien Ee, David Culler, Scott Shenker, and Ion Stoica. Beacon vector routing: scalable point-to-point routing in wireless sensornets. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, NSDI'05*, pages 329–342, Berkeley, CA, USA, 2005. USENIX Association.

[GFJ+09] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systemsm, SenSys '09*, November 2009.

[GGC03] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings of the 4th International Conference on Mobile Data Management, MDM '03*, pages 45–62, London, UK, 2003. Springer-Verlag.

[GhIgGhPd] Carlos F. García-hernández, Pablo H. Ibargengoytia-gonzález, Joaquín García-hernández, and Jesús A. Pérez-díaz. "wireless sensor networks and applications: a survey". *International Journal of Computer Science and Network Security,*.

[GMPK10] Damianos Gavalas, Aristides Mpitziopoulos, Grammati Pantziou, and Charalampos Konstantopoulos. An approach for near-optimal distributed data fusion in wireless sensor networks. *Wireless Networks*, 16(5):1407–1425, 2010.

[HCM05] Fei Hu, Xiaojun Cao, and Carter May. Optimized scheduling for data aggregation in wireless sensor networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing, ITCC'05*, pages 557–561, Washington, DC, USA, 2005. IEEE Computer Society.

[HL07] Gertjan Pieter Halkes and Koen Langendoen. Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks. Delft, The Netherlands, January 2007.

[IEGH02] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22 nd International Conference on Distributed Computing Systems, ICDCS'02*, page 457, Washington, DC, USA, 2002. IEEE Computer Society.

[IGE00] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking, Mobicom '00*, pages 56–67, Boston, MA USA, 2000.

[IGE+03] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[JCH84] Rajendra K. Jain, Dah-Ming W. Chiu, and William R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, Digital Equipment Corporation, September 1984.

[JH08] Yuh-Jzer Joung and Shih-Hsiang Huang. 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*, pages 237–251, Berlin, Heidelberg, 2008. Springer-Verlag.

[JX05] Shanping Li Qing Gao Gang Peng Jian Xu1, Jianliang Xu. Adaptive data dissemination in wireless sensor networks. In *Parallel and Distributed Processing and Applications*, pages 156–168. Springer Berlin / Heidelberg, 2005.

[KEW02] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems, ICDCS '02*, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[KK00] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *"Proceedings of the sixth annual international conference on Mobile computing and networking, Mobicom '00"*, pages 243–254, New York, NY, USA, 2000. ACM.

[KL05] Naoto Kimura and Shahram Latifi. A survey on data compression in wireless sensor networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing, ITCC'05*, pages 8–13, Washington, DC, USA, 2005. IEEE Computer Society.

[KSC08] Youngmin Kim, Hyojeong Shin, and Hojung Cha. Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks, IPSN '08*, pages 53–63, Washington, DC, USA, 2008. IEEE Computer Society.

[LHZ04] Xin Liu, Qingfeng Huang, and Ying Zhang. Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*, pages 122–133, New York, NY, USA, 2004. ACM.

[LKGH03] Xin Li, Young Jin Kim, Ramesh Govindan, and Wei Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*, pages 63–75, New York, NY, USA, 2003. ACM.

[LKR04] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium, IPDPS '04*, page 224, April 2004.

[LLL06] Guilin Li, Jianzhong Li, and Jinbao Li. Adaptive data transmission algorithm for event-based ad hoc query. In *APWeb Workshops*, pages 249–256, 2006.

[LLLD06] Hong Luo, Jun Luo, Yonghe Liu, and Sajal K. Das. Adaptive data fusion for energy efficient routing in wireless sensor networks. *IEEE Transactions on Computing*, 55:12861299, 2006.

[LSJ06] Dandan Liu, Ivan Stojmenovic, and Xiaohua Jia. A scalable quorum based location service in ad hoc and sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, MASS '06*, pages 489–492, 2006.

[LSS07] Xu Li, Nicola Santoro, and Ivan Stojmenovic. Mesh-based sensor relocation for coverage maintenance in mobile sensor networks. In *Proceedings International Conference on Ubiquitous Intelligence and Computing, UIC '07*, pages 696–708, 2007.

[LSS08] Xu Li, Nicola Santoro, and Ivan Stojmenovic. Localized distance-sensitive service discovery in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing, FOWANC '08*, pages 85–92, New York, NY, USA, 2008. ACM.

[LSW10] Wen-Hwa Liao, Kuei-PIng Shih, and Wan-Chi Wu. A grid-based dynamic load balancing approach for data-centric storage in wireless sensor networks. *Computer and Electrical Engineering*, 36(1):19–30, 2010.

[LXY05] Thang Nam Le, Dong Xuan, and Wei Yu. An adaptive zone-based storage architecture for wireless sensor networks. In *IEEE Global Telecommunications Conference, GLOBECOM'05*, New York, NY, USA, 2005. IEEE.

[MKC07] G. Montenegro, N Kushalnagar, and David Culler. RFC 4944. transmission of ipv6 packets over ieee 802.15.4 networks, 2007.

[MM02] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *First International Workshop on Peer-to-Peer Systems, IPTPS '01*, pages 53–65, London, UK, 2002. Springer-Verlag.

[Mol94] Jesper Moller. *Lectures on random Voronoi tesselations*. Springer-Verlag, 1994.

[Mul09] The multihop LQI protocol, http://www.tinyos.net/tinyos-2.x/tos/lib/net/lqi, 2009.

[Neu51] John Von Neuman. *Various techniques used in connection with random digits*, volume 5. In Taub, A.H., ed: John von Neuman, Collected Works., 1951.

[NS03] James Newsome and Dawn Song. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*, pages 76–88, New York, NY, USA, 2003. ACM.

[oEE06] Institute of Electrical and Electronics Engineers. IEEE std. 802.15.4-2006: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (lr-wpans), 2006.

[PBRD03] Charles Perkins, Elizabeth M. Belding-Royer, and Samir Das. Rfc 3561. ad hoc on-demand distance vector (aodv) routing, 2003.

[PHC04] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*, pages 95–107, New York, NY, USA, 2004. ACM.

[PKL07] Jim Partan, Jim Kurose, and Brian Neil Levine. A survey of practical issues in underwater networks. *SIGMOBILE Mobile Computer Communications Review*, 11(4):23–33, 2007.

[PSW04] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Communications of the. ACM*, 47(6):53–57, 2004.

[Rab89] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Lecture Notes in Computer Science*, pages 329–350, 2001.

[RKS+03] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Network Applications*, 8(4):427–442, 2003.

[RKY+02] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. 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*, pages 78–87, New York, NY, USA, 2002. ACM.

[RV06] Ramesh Rajagopalan and Pramod K. Varshney. Data aggregation techniques in sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, 8:48–63, 2006.

[RV08] Gianluca Mazzini Andrea Conti Roberto Verdone, Davide Dardari. *Wireless Sensor and Actuator Networks*. Elsevier, 2008.

[RWAM05] Injong Rhee, Ajit Warrier, Mahesh Aia, and Jeongki Min. Z-mac: a hybrid mac for wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems, SenSys '05*, pages 90–101, New York, NY, USA, 2005. ACM.

[scr07] Tymo source code repository. Tymo: Dymo implementation for tinyos. 2007.

[SEP99] Ivan Stojmenovic, Pedro Eduardo, and Pedro Eduardo Villaneuva Pea. A scalable quorum based location update scheme for routing in ad hoc wireless networks, 1999.

[SH04] Karim Seada and Ahmed Helmy. Rendezvous regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks. *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04) - Workshop 12*, 13, 2004.

[SKM95] Dietrich Stoyan, Wilfrid S. Kendall, and Joseph Mecke. *Stochastic Geometry and its Applications*. J. Wiley & Sons, Chichester, 1995.

[SLJ08] Ivan Stojmenovic, Dandan Liu, and Xiaohua Jia. A scalable quorum-based location service in ad hoc and sensor networks. *International Journal on Communication Network and Distributed Systems*, 1(1):71–94, 2008.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM Conference, SIGCOMM '01*, San Diego, California, August 2001.

[SRK+03] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *SIGCOMM Computer Communication Review*, 33(1):137–142, 2003.

[SZG06] Rik Sarkar, Xianjin Zhu, and Jie Gao. Double rulings for information brokerage in sensor networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking, Mobicom '06*, pages 286–297, New York, NY, USA, 2006. ACM.

[TNK04] R. Tanushetty, Lek Heng Ngoh, and Pung Hung Keng. An efficient resiliency scheme for data centric storage in wireless sensor networks. In *IEEE Vehicular Technology Conference, 2004. VTC2004-Fall*, New York, NY, USA, 2004. IEEE.

[TWXD06] Nam Le Thang, Yu Wei, Bai Xiaole, and Xuan Dong. A dynamic geographic hash table for data-centric storage in sensor networks. In *IEEE Wireless Communications and Networking Conference, WCNC '06.*, pages 2168 – 2174, New York, NY, USA, 2006. IEEE.

[vDL03] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*, pages 171–180, New York, NY, USA, 2003. ACM.

[WGA06] Dirk Westhoff, Joao Girao, and Mithun Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Transactions on Mobile Computing*, 5(10):1417–1431, 2006.

[WSL+06] Chonggang Wang, Kazem Sohraby, Bo Li, Mahmoud Daneshmand, and Yueming Hu. A survey of transport protocols for wireless sensor networks. *IEEE Network*, 20(3):34–40, May-June 2006.

[WTC03] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*, pages 14–27, New York, NY, USA, 2003. ACM.

[XCCX08] Lei Xie, Lijun Chen, Daoxu Chen, and Li Xie. Eebass: Energy-efficient balanced storage scheme for sensor networks. In *GLOBECOM*, pages 719–724, New York, NY, USA, 2008. IEEE.

[YCL+10] Chia-Mu Yu, Chi-Yuan Chen, Chun-Shien Lu, Sy-Yen Kuo, and Han-Chieh Chao. Acquiring authentic data in unattended wireless sensor networks. *Sensors*, 10(4):2770–2792, 2010.

[YHE02] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 21st IEEE Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '02*, volume 3, pages 1567 – 1576, 2002.

[YKT03] Wei Yuan, Srikanth V. Krishnamurthy, and Satish K. Tripathi. Synchronization of multiple levels of data fusion in wireless sensor networks. In *Proceedings of the Global Telecommunications Conference, GLOBECOM '03*, volume 1, pages 221 – 225, December 2003.

[Yu09] Gwo-Jong Yu. Adaptive storage policy switching for wireless sensor networks. *Wireless Personal Communication*, 48(3):327–346, 2009.

[ZCLP03] Wensheng Zhang, Guohong Cao, and Tom La Porta. Data dissemination with ring-based index for wireless sensor networks. In *Proceedings of the 11th IEEE International Conference on Network Protocols, ICNP '03*, page 305, Washington, DC, USA, 2003. IEEE Computer Society.

[ZCR08] Yao Zhao, Yan Chen, and Sylvia Ratnasamy. Load balanced and efficient hierarchical data-centric storage in sensor networks. In *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08.*, pages 560–568, New York, NY, USA, 2008. IEEE.

[ZWR+09] Jinsong Zhang, Malaka Walpola, David Roelant, Hao Zhu, and Kang Yen. Self-organization of unattended wireless acoustic sensor networks for ground target tracking. *Pervasive Mobile Computing*, 5(2):148–164, 2009.