UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

YAKOV NAE

"MODELO DISTRIBUÍDO PARA AGREGAÇÃO DE ARMAZENAMENTO EM REDES DE SENSORES SEM FIO"

"DISTRIBUTED MODEL FOR STORAGE AGGREGATION IN WIRELESS SENSOR NETWORKS"

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Telecomunicações e Telemática.

A Master dissertation submitted to the school of Electrical and Computer Engineering as part of the requirements for the Master degree in Electrical Engineering. Focus area: Telecommunications and Telematics

Orientador: Prof. Dr. Lee Luan Ling
Tutor: Prof. Dr. Lee Luan Ling

Campinas, SP
2011

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Yakov Nae
F149u Modelo Distribuído para Agregação de
Armazenamento em Redes de Sensores Sem Fio
Yakov Nae. – Campinas, SP:
[s.n.], 2011.

Instructor: Lee Luan Ling.
Tese (mestrado) - Universidade Estadual de Campinas,
Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de Telecomunicações.
2. RSSF Many - to - Many. 3. Algoritmos Distribuídos.
4. Gráficos típo Expander. 5. Roteamento Geográfica.
6. Passeio Aleatório. 7. Agregação de Recursos.
I. Lee Luan Ling. II. Universidade Estadual de Campinas.
Faculdade de Engenharia Elétrica e de Computação. III.
Título

Título em Inglês:	Distributed Model For Storage Aggregation in Wireless Sensor Networks
Palavras-chave em Inglês:	Telecommunication systems, Many - to - Many WSN, Distributed Algorithms, Expander Graphs, Geographical Routing, Random Walk, Resource Aggregation
Área de concentração:	Engenharia Elétrica e Telecomunicações
Titulação:	Mestre em Engenharia Elétrica
Banca Examinadora:	Celso de Almeida, Flávio Henrique Teles Viera
Data da defesa:	14/12/2011

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Yakov Nae

Data da Defesa: 14 de dezembro de 2011

Título da Tese: "Modelo Distribuído para Agregação de Armazenamento em Redes de Sensores"

Prof. Dr. Lee Luan Ling (Presidente): _____

Prof. Dr. Flávio Henrique Teles Vieira: _____

Prof. Dr. Celso de Almeida: _____

Abstract

Storage management of Wireless Sensor Networks (WSN) is a very critical issue in terms of system's *lifetime*. While WSNs host a vast storage capacity on the aggregate, that capacity cannot be used entirely. Eventually, the entire network may fail when the first sensor has its own storage capacity depleted, leaving behind a large amount of unutilized storage capacity. We suggest that sensors should be able to detect unutilized storage capacity in order to prolong their functionality. However, for large scale WSNs this can be a difficult task, since sensors may not be aware of the existence of others. This work has two main contributions: an optimization of the overall storage capacity for large scale WSNs and a novel routing approach of *deterministic "random" walk*. We present a new storage model by building *"on - demand" Distributed Storage Chains (DSC)*. These chains represent partnership between sensors that share their storage capacity. As a result, sensors are no longer subjected to their own storage limitations but to the total amount of available storage in the WSN. We construct these chains via *deterministic* walks over our suggested topology. However, we show that these walks resemble the behavior of *random* walks and are therefore highly efficient in terms of locating available storage.

Keywords: Telecommunication systems, Many - to - Many WSN, Distributed Algorithms, Expander Graphs, Geographical Routing, Random Walk, Resource Aggregation.

Resumo

Gerência de armazenamento em Redes de Sensores Sem Fio (RSSF) é uma questão muito crítica. Além da RSSFs conter uma vasta quantidade de armazenamento agregada, ela não pode ser usada inteiramente. Portanto, o sistema inteiro falha quando o primeiro sensor tem sua capacidade de armazenamento esgotada, deixando uma grande capacidade de armazenamento inutilizada. Sugere-se que os sensores devem-se ser capazes de detectar as capacidades de armazenamentos inutilizadas, para prolongar as suas funcionalidades. Entretanto, em RSSF de larga escala isso pode ser muito difícil uma vez que os sensores podem não ter conhecimento da existência dos outros. Neste trabalho apresenta-se duas principais contribuições: otimização da capacidade total de armazenamento para RSSF em grande escala e uma nova abordagem de roteamento - *Deterministic "Random" Walk* (*Passeio "Aleatório" Determinístico*). Apresenta-se um novo modelo de armazenamento via construção *"sob demanda"* de Cadeias de Armazenamento Distribuídas (*Distributed Storage Chains (DSC)*). Estas cadeias representam parcerias entrem os sensores que podem compartilhar suas capacidades de armazenamento. Resultando, os sensores não estão sujeitos às suas limitações de armazenamento, mas para à capacidade total de armazenamento disponível no sistema. Constrói-se estas cadeia via passeio *determinístico* sobre a topologia sugerida. Todavia, mostra-se que estes passeios apresentam um comportamento *aleatório* que é muito eficiente em termos de localização de capacidade de armazenamento disponível.

Palavras-chave: Sistemas de telecomunicações, RSSF Many - to - Many, Algoritmos distribuídos, Grafos tipo Expander, Roteamento Geográfico, Passeio Aleatório, Agregação de Recursos.

To my entire family

Acknowledgements

Gostaria de expressar minha sincera gratidão para minha família inteira e os meus amigos por seu sopurte durante esta jornada, especialmente para a minha irmã Luciane Costa da Silva, que sem ela nada disto teria sido possível.

Queria agradecer a CAPES pelo apoio financeiro.

I would like to express my sincere gratitude to my entire family and friends for the support during this journey, especially to my sister Luciane Costa da Silva, who without her none of this would have been possible.

I would like to thank CAPES Scholarship for the financial support

Contents

List of Figures

List of Tables

Abbreviations

DSC	-	**D**istributed **S**torage **C**hains
EOC	-	**E**nd **O**f **C**hain
LSC	-	**L**ocal **S**torage **C**onsumption
WSN	-	**W**ireless **S**ensor **N**etwork
SNR	-	**S**ignal (to) **N**oise **R**atio
GPS	-	**G**lobal **P**oisition **S**ystem
GPSR	-	**G**reedy **P**erimeter **S**tateless **R**outing
TTL	-	**T**ime **T**o **L**ive
P2P	-	**P**eer **T**o **P**eer

Symbols

n	-	Network size.
M	-	Number of storage units per sensor.
d	-	Expansion parameter.
HS	-	Number of hot-spot sensors.
γ	-	Percentage of the overall WSN memory being used.
p	-	Probability for an event to be generated on a hot-spot region.
STORE	-	Refers to the operation of storing events outside of the sensor who sensed them.
RETRIEVE	-	Refers to the operation of retrieving events that where stored by STORE.
PUT	-	A packet that carry events out of the sensor who stored them in order to be stored on an available sensor.
ACK	-	A packet that confirms that events where stored outside the sensor who sensed them. Also indicates the new EOC (End Of Chain).
GET	-	A packet that carries a query along the chain.
$E_{elec}\ [J/bit]$	-	Transceiver radio dissipate parameter.
$\varepsilon_{amp}\ [J/bit/m^2]$	-	Transmission amplification parameter.

Trabalhos Publicados Pelo Autor

1. Y. Nae and L.L. Ling. “Modelo Distribuído para Otimização do Armazenamento em Redes de Sensores Sem Fio”. *XXIX Simpósio Brasileiro de Telecomunicações* (SBT’11), Curitiba, Paraná, Brasil, Outubro 2011.

Chapter 1

Introduction

"...The problem in the world today is communication. Too much communication..."

Homer Simpson

1.1 Scope

Wireless Sensor Networks (WSNs or SensorNets) are distributed sensing systems. They are used to visualize, through numerical measurements, physical or environmental phenomena that cannot be observed through conventional images from cameras and satellites e.g.,. WSNs are comprised of large numbers of small autonomous sensing devices that are *densely* deployed over a target area (Akyildiz et al., 2002). These devices observe and collect measurements from different regions. Then, all measurements are processed in order to study and analyze the monitored phenomena on the target area. WSN is an evolving and increasing trend on science today, producing a high volume of scientific papers and investigating various aspects (Culler, Estrin, & Srivastava, 2004).

Following the Moore's law, the number of transistors on a minimum component cost chip doubles every year or two (Culler et al., 2004). Thus, on every passing year, WSN technology is powered by smaller, cheaper and more intelligent sensors. Moreover, researchers are now applying WSN technology in ways that enable a new role for computing in science. WSN technology can provide the state of the art solutions for monitoring extensive observations such as seismographic activity, weather, surveillance, radioactivity, habitat monitoring, etc.

WSNs are distinguished from other *ad-hoc* networks because of their limitations in resource capacities: they have limited processing speed, energy, memory and communications. Since a WSN may contain millions of sensors (Akyildiz et al., 2002), the cost of a single sensor becomes a very

critical factor. In addition, a sensor size can vary from the size of a shoe box to the size of a grain of dust. Therefore, size and cost constraints of sensor nodes result in limitations on their available resource capacities. In other words, reduction in cost and size result in reduction of resource capacities such as: energy, storage, computational capacity and communication range. These limitations have serious implications on the design of application and protocols.

WSNs are classified into *real-time* and *non-real-time* networks, depending on their application and the type of the monitored phenomena. Real-time applications (e.g. surveillance) require that the observed event be immediately transmitted to an end user. On the other hand, significant numbers of applications that do not require real-time information, allow the measurements to be stored within the sensors (Luo, Huang, Abdelzaher, & Stankovic, 2007). Therefore, information can be collected after a defined period of time, when the experiment is over or even by mobile base station (data mules (Luo et al., 2007)) while visiting the target area. WSN in general and especially non-real-time WSN architecture must take under serious considerations the storage aspect i.e., how the events should be stored, when and where do events should be processed into observations and how observations are retrieved.

In this work, we propose a Distributed Model for Storage Aggregation in Wireless Sensor Networks. Basically, our model addresses a non-real-time WSN architecture and deal with its storage and retrieval aspects. In brief, we wish to maximize the storage utilization on WSN by allowing each sensor to access the entire wsn's aggregated storage capacity. In other words, our model suggests that sensors who had their own *local* storage capacity depleted can use unutilized reserves from sensors that are located on low activity regions.

1.2 Problem Statement

The problem with WSN lies within the limitations in *local* resource capacities of each individual sensor. While WSNs host vast resource capacities on the aggregate, each sensor is subjected to its own poor local resource limitations. Therefore, the entire system is subjected to local resource constraints, derived from every individual sensor, rather than being subjected to its *global* aggregated resource capacities.

In this work, we focus into the WSN resource capacity problem and handle only the limitations on the storage capacity resource. We consider a non-real-time WSN that is uniformly deployed over a target area of interest. Sensor nodes are static and communicate between themselves in a multi-hop fashion. Each sensor periodically sense different events that occur on its own region and stores their measurements. When a sensor node had its storage capacity depleted, it transmits its additional events to some other sensor with available storage capacity. In addition, sensors should be able to retrieve

all the events that they have sensed.

In this work, our problem is how to maximize the storage utilization of the entire WSN. We increase the WSN storage limitations by allowing sensors to share their storage capacities. We do so, by designing a distributed network protocol for WSN.

1.3 Approach

We address the problem of overflowed events, which cannot be locally stored within those sensors who had their storage capacity depleted. Generally, our model aim to enable sensors, which deplete their storage capacity, to store their overflowed events on available host-sensors. As a result, sensors that produce more data than they can store locally, have a global storage solution. Moreover, sensors that produce less data than their storage capacity can maximize their storage utilization by hosting overflowed events. As a result, the WSN becomes more robust as we maximize its overall storage utilization. Our approach towards designing our model in this work is built of four phases: network topology design, discovery algorithms on sensor networks, modeling events generation on the target area and protocol design.

Network topology design. Our main motivation for designing this model began with a work on the field of expander graphs (Gabber & Galil, 1979). These kinds of graphs are mainly characterized by strong connectivity, while maintaining a low number of edges. Therefore, expander graphs suggest a very efficient network topology for communication. In this work, we study an expander graph implementation for a random deployment of points on a geographical target area. Particularly, although that geographic routing and expander graphs already exist, the construction that we suggest for a geographic expander graph is a new concept. One of the main challenges of this work is to prove that our suggested construction is indeed an expander graph.

Discovery algorithm. One of the most critical issues on large scale WSN is the communication scheme. Normally, on large scale sensor network it is impossible for each sensor to know all the other participants. Logically, every two sensors must "know" each other in order to communicate. Our new construction allows a *fast* discovery of the network from every sensor location. In other words, each sensor who implements a simple walk over the suggested topology, discovers new sensors at each step with high probability. With this fast discovery, we allow localization of sensors with available storage capacity. Therefore, sensors can use that capacity in order to host overflowed events.

Modeling event generation on the target area. When a WSN is deployed over a target area that is perfectly balanced, events occur uniformly on all regions. Moreover, all sensors observe the same amount of events at any given time. Therefore, all sensors have their storage capacity depleted together. Naturally, the WSN maximizes its storage utilization up to its global limit. However, in

real WSN environment, this may not be practical. A target field that is not balanced contains regions that generate more events than others. Sensors that are deployed on those regions may rapidly deplete their storage capacity and form coverage holes on the target area. In order to evaluate our storage aggregation model, we had to develop a traffic model that represent a non-balanced target area. Eventually, we used this traffic model as a control group for our storage aggregation model.

Protocol Design. The product of our work is a distributed protocol for WSN. We use our non-uniform event generation model in order to evaluate protocol performances. We evaluated the performance of our protocol according to the maximization of the storage utilization within the WSN.

1.4 Contributions

To summarize, we have succeeded to maximize the storage utilization on WSNs. Moreover, we will demonstrate that our model maintains fairness conditions within the WSN. In other words, all sensors participate with (more or less) the same intensity. While developing our solution we came upon three main contributions and innovations:

First. We developed a *distributed geographic expander overlay network* for sensors that are uniformly distributed over a target area. We evaluated the network performances through extensive simulations and found that its behavior is similar to that of an expander graph. Thus, we have established a geographic expander network topology for sensor network, which can be constructed on a distributed manner. Our next contribution is based on this topology.

Second. We developed a geographical *Deterministic "Random" Walk*. This walk is consisted of a deterministic sequence of routing instructions that *"randomly samples"* the geographic target area. This novel geographic walk allows sensors to select partners from the WSN network and share their resources. We show that this selection have the same probability as a uniform selection of sensors from the WSN. However, that selection is, definitely, a deterministic selection that depends only on the location of the sensor. This walk gains its uniqueness for the ability to generate "random" partnerships that can both be predicted and reconstructed.

Third. We developed a traffic model for non-uniform event generation on WSN. Very little is mentioned on the literature about non-uniform generation models. Most works in the literature assumes a uniform event distribution (Heinzelman et al., 2000; Ratnasamy et al., 2003; Silberstein & Yang, 2007) although the reality is different (Hung-Yu, Gwo-Jong, & Jang-Ping, 2005). Our model is more oriented towards real sensor networks, where different regions have different rates for generation of events.

Forth. We developed a WSN protocol that maximizes the overall storage utilization of the WSN. This protocol uses the geographical walk, described on the second achievement, for generating inte-

ractions between sensors. Sensors use the Deterministic "Random" walk to share their storage capacities. This walk is of great impact, because no information needs to be kept in order to reconstruct it. In other words, no routing tables or any kind of instructions are needed to be maintained in order to retrieve data. Moreover, the effectiveness of these walks is derived from the uniform "sampling" characteristic. Hence, when conducted from different sensors (different locations) it would not cause starvation on specific regions.

1.5 Outline

This work suggests a novel WSN application for the former Gabber-Galil expander graph (Gabber & Galil, 1979). We have developed a search function based on this graph that can be used by WSN. Moreover, although the original graph was designed for a discrete grid network, we implemented it for non-grid WSNs. Also, based on that implementation, we suggest a distributed WSN protocol for storage management and analyze its performance. We organize this work as follows.

In Chapter 3 We begin with a brief summary on WSN and their applications. We give some important details about different WSN architectures and explain the question of energy consumption on WSN. We conclude this chapter with a scenario example for WSN. That scenario is an example that can be very helpful in understanding the need in our distributed storage aggregation model.

In Chapter 4 we supply some background about expander graphs and introduce the original Gabber-Galil expander graph for a *discrete* grid network. Then, we develop our *continuous* expander graph above a deployment of sensors in field. We define a 2-D *continuous discrete approach* (Naor & Wieder, 2007) that relates sensor nodes with our continuous expander graph.

In Chapter 5 we begin by defining our network layer (GPSR - Greedy Perimeter Stateless Routing protocol) (Karp & Kung, 2000) and the continuous geographical address space (Voronoi diagrams (Berg, 2000)). Then, we define the main innovation of our work, a *Deterministic "Random" Walk*, which is based on the geographic expander graph (Chapter 4), GPSR and the Voronoi diagrams. We use our walk as a network discovery algorithm that allows individual sensors to interact with others.

In Chapter 6 we deal with non-uniform traffic model for WSN. We explain the importance of this model for the performance analysis of our Distributed Storage Chains (DSC). We define the term WSN lifetime with respect to the data generated on the target area. We develop the probability function that predict the expected lifetime of a WSN, based on the traffic model that we suggest.

In Chapter 7 we formulate a distributed protocol for storage management on WSN. That is, a protocol that maximizes the storage utilization on WSNs that are deployed over non-uniform environments (Chapter 6) and , therefore, prolong the WSN lifetime. Our protocol makes use of our Distributed Storage Chains (DSC) (Chapter 5) in order to locate sensors with available storage capac-

ity.

Chapter 8 Throughout this work we show specific simulations results that concerns the topic of each chapter. However, in Chapter 8 we include the major part of our simulations as we estimate the performances of our protocol. Finally in **Chapter 9** we provide our conclusions and discuss about potential future work.

Chapter 2

Introdução

"As coisas valem pelas idéias que nos sugerem"

Machado de Assis (1839 - 1908)

2.1 Escopo

Rede de Sensores Sem Fio (RSSF), em inglês Wireless Sensor Networks (WSNs), são sistemas sensoriais distribuídos. Eles visualizam, através de medidas numéricas, fenômenos físicos ou ambientais que não podem ser observados em imagens convencionais de câmeras ou satélites por exemplo. RSSF são sistemas que são compostos de um grande número de pequenos dispositivos sensoriais autônomos, que são densamente implantados numa área - (Akyildiz et al., 2002). Estes dispositivos observam e coletam medições de diferentes regiões. Assim, todas as medições são processadas para que os fenômenos monitorados na área - alvos possam ser estudados e analisados. RSSF é uma tendência crescente e em constante aprimoramento na ciência atual.

De acordo com a Lei do Moore, a quantidade de transistores em um chip, de custo mínimo, dobraria a cada um ano ou dois (Culler et al., 2004). Assim, a cada ano que passa, a tecnologia da RSSF é incrementada por sensores cada vez mais inteligentes, menores e mais baratos. Isso tem levado a pesquisas em diferentes campos de aplicação para a RSSF permitindo a ciência da computação chaga a níveis cada vez mais altos. A tecnologia RSSF fornece soluções de estados da arte para monitoramento extensivo de observações como: atividade sísmica, climáticas, vigilância, radioatividade, monitoramento de habitat, etc.

As RSSF são diferenciadas de outras redes *ad-hoc* por causa de suas limitações de recursos. Dado que o tamanho de uma RSSF varie de centenas até milhões de sensores, o custo de cada sensor torna-

se um fator crítico. Adicionalmente, o tamanho de cada sensor pode variar entre o tamanho de uma caixa de sapato a um grão de poeira. Assim, as restrições de tamanho e custo de um sensor influenciam correspondentemente em suas limitações de recursos. A fim de terem seus custos reduzidos, os sensores sem fio sofrem limitações em recursos como: energia, memória, comunicações, etc... Estas restrições trazem sérias implicações no design de aplicações e de protocolos.

As RSSF são classificadas em sistemas de *tempo-real* (*real-time*) e sistemas de *tempo-não-real* (*non-real- time*) conforme sua aplicação, ou conforme o tipo de fenômeno monitorado. As aplicações de tempo real (por exemplo, vigilância) requerem que os eventos observados sejam imediatamente transmitidos para um usuário final. Por outro lado, um número significativo de aplicações que não requerem informações em tempo real permite que as medições sejam armazenadas pelos sensores (Luo et al., 2007). Assim, a informação pode ser coletada após um período predefinido de tempo ou ate mesmo após o término do experimento. A arquitetura de RSSF de tempo não real precisa considerar seriamente os aspectos de armazenamento. Em outras palavras, onde e quando os eventos observados devem ser armazenados, onde e quando os eventos devem ser processados para observações e onde e como as observações devem ser recuperadas.

Neste trabalho é sugerido um Modelo Distribuído para Agregação do Armazenamento em RSSF. Este modelo propõe uma arquitetura da RSSF de tempo não real para gerenciar seu armazenamento e a recuperação de eventos. Em resumo, maximiza-se a utilização de armazenamento na RSSF, possibilitando que cada sensor use a capacidade total disponível do sistema. De acordo com este modelo, quando os sensores têm suas capacidades de armazenamento *local* esgotadas, eles podem usar armazenamento dos outros sensores com capacidade livre.

2.2 Definição do Problema

Um dos problemas com os sistemas RSSF encontra-se nas limitações *locais* de recursos de cada sensor. Enquanto o sistema RSSF conteria uma vasta capacidade dos recursos, caso fossem agregados, cada sensor do sistema, por sua vez, estaria sujeito às limitações locais dos recursos. Desta forma, o sistema inteiro estaria sujeito às restrições locais dos recursos de cada sensor, ao invés de estar sujeito aos recursos *globais* agregados.

Neste trabalho, pretende-se focar na capacidade de armazenamento do sistema e em suas limitações, como um exemplo para os recursos que podem ser agregados. Considera-se um sistema RSSF de tempo não real distribuído sobre um campo alvo de interesse. Os nós de sensores são estáticos e comunicam-se entre si atraves do modo salto múltiplo (*multi - hop*). Periodicamente, cada sensor percebe atividades em sua região e precisa armazenar suas medições. Quando possível o sensor armazena as medições localmente. No caso do sensor já tem sua capacidade de armazenamento es-

gotada, ele fará o armazenamento de forma global. Ou seja, ele pode usar os sensores disponíveis que ainda têm capacidade de armazenamento disponível. Adicionalmente, as atividades que têm sido armazenadas globalmente precisam ser recuperadas pela sistema quando for necessário.

Este trabalho confronta-se com o problema de maximização de utilização de armazenamento total dA RSSF. Em outras palavras, pretende-se incrementar as limitações locais de armazenamento habilitando os sensores para compartilhamento de suas capacidades de armazenamento. Para atingir este objetivo foi desenvolvido um novo protocolo distribuído para RSSF.

2.3 Abordagem adotada

Considera-se o problema das atividades sobrecarregadas, que não podem ser armazenadas localmente num sensor que já tem a sua capacidade de armazenamento esgotada. De forma geral, os sensores que já têm suas capacidades locais de armazenamento esgotadas, devem transferir seus registros de atividades para os sensores que ainda têm capacidade de armazenamento disponível. Assim, sensores que produzem mais dados do que conseguem armazenar, passam a ter uma nova solução de armazenamento global. Além disso, sensores que produzem quantidade de dados menor de que suas capacidades de armazenamento podem maximizar a utilização de armazenamento do sistema enquanto hospedam as atividades excessivas de outros sensores. Como resultado, o sistema torna-se mais robusto e sua eficácia será prolongada através da maximização de sua utilização de armazenamento global. Neste trabalho, o modelo sugerido se constitui por quatro etapas correspondentes: Definição da topologia da rede, algoritmos de descoberta em RSSF, modelagem de geração de eventos num campo alvo e desenvolvimento de protocolo.

Definição da topologia da rede. A maior motivação para a topologia da rede sugerida nesse trabalho partiu de uma pesquisa no campo de grafos tipo *expander*. Estes tipos de grafos são caracterizados principalmente por manter uma alta conectividade apesar de terem poucos enlaces. Assim, grafos tipo *expander* sugerem uma topologia de rede muito eficiente para comunicações. Neste trabalho, estuda-se a implementação do grafo tipo *expander* em cima de uma implantação aleatória dos pontos num campo alvo geográfico. Reforçando, apesar de já existir grafos geográficos e grafos tipo *expander*, a apresentação de uma topologia de rede que envolve os dois juntos é um conceito inovador. Um dos objetivos deste trabalho é provar que esta nova topologia geográfica é realmente um grafo tipo *expander*.

Algoritmos de descoberta em uma RSSF. Um dos tópicos mais críticos no desenvolvimento da RSSF em larga escala, é a forma de comunicações entre sensores. Normalmente, em redes de larga escala, é impossível requerer que os sensores identifiquem todos os demais sensores da rede. De forma simples, para dois sensores se comunicarem é necessário que um “conheça” o outro. Esta nova

topologia apresentada neste trabalho possibilita uma descoberta *rápida* da rede a partir da localização de cada sensor. Em outras palavras, cada sensor que realiza um passeio simples em cima da topologia sugerida, consegue descobrir com probabilidade alta novos sensores em cada roteamento. Com esta descoberta rápida, possibilita-se a localização de sensores com capacidade de armazenamento disponível. Assim, quando necessário, sensores podem explorar esta capacidade de armazenamento e armazenar sua sobrecarga de dados.

Modelagem de geração de eventos num campo alvo. Quando uma RSSF é implantada sobre um campo alvo que é perfeitamente balanceado, as atividades ocorrem uniformemente nas diferentes regiões. Assim, todos os sensores observam a mesma quantidade de atividades. Em função disso, todos os sensores juntos têm suas capacidades de armazenamento esgotadas. Naturalmente, o sistema maximiza sua utilização de armazenamento até seu limite global. Entretanto, se o campo não for balanceado ele terá algumas regiões que geram mais atividades que outros. Sensores que são implantados nestas regiões hiperativas tenham suas capacidades de armazenamento esgotadas mais rapidamente que os outros. Assim, eles formam buracos de cobertura no campo alvo. A fim de avaliar este novo modelo de agregação de armazenamento, desenvolveu-se um modelo de tráfego que simula um campo alvo não balanceado que é mais realista. Eventualmente, usa-se este modelo de tráfego como um grupo de controle para o modelo de agregação de armazenamento proposto.

Desenvolvimento de protocolo. O produto deste trabalho é entregue na forma de um protocolo distribuído para a RSSF. O desempenho deste protocolo é avaliado com simulações que implementam o modelo de geração de eventos acima citado. O sucesso deste protocolo é medido de acordo com a otimização da utilização da capacidade de armazenamento do sistema.

2.4 Contribuições do Autor

Em resumo, este trabalho foi concluído com sucesso. Em particular, obteve-se a maximização da utilização do armazenamento para RSSF escalável. Tudo isso foi possível ainda mantendo custo baixo dos enlaces e balanceamento de carga estável entre os sensores. Além disso, durante o desenvolvimento desta solução chegou-se a três contribuições importantes e realmente inovadoras:

Primeiro. Desenvolveu-se uma rede *overlay* geográfica distribuída tipo *expander* para sensores, que são uniformemente distribuídos sobre uma área alvo. Os desempenhos da rede foram medidos através de simulações extensivas. Assim, descobriu-se que seu comportamento é similar a um grafo tipo *expander*. Desta forma, estabeleceu-se uma topologia de rede geográfica tipo *expander* para rede de sensores. Além disso, esta rede poderia ser construída de maneira distribuída. A próxima contribuição que será apresentada é baseada nesta topologia

Segundo. Desenvolveu-se um Passeio "Aleatório" Determinístico que consegue amostrar uni-

formemente o campo alvo geográfico. Este novo passeio geográfico permite que os sensores selecionem uniformemente seus parceiros da mesma rede (assim como em uma seleção aleatória), para compartilhar suas reservas. Entretanto, esta seleção é definitivamente uma seleção determinística que depende apenas da localização do sensor. Este passeio ganha seu caráter exclusivo devido à sua habilidade de gerar parceiros *aleatórios* que podem ser pré-definidos e recuperados.

Terceiro. Modelos de geração não uniforme são pouco mencionados na literatura. Neste trabalho desenvolveu-se um modelo de tráfego para geração não uniforme de eventos para RSSF *many - to - many*. O modelo desenvolvido é mais orientado para redes reais de sensores, onde regiões diferentes apresentam diferentes níveis de geração de atividades.

2.5 Outline

Neste trabalho, será proposta uma nova aplicação RSSF para o já existente grafo *expander* (Gabber & Galil, 1979). Conseguiu-se desenvolver uma implementação distribuída para este tipo de grafo que poderá ser usada numa RSSF. Além disso, apesar do grafo original ter sido desenhado para uma rede *grid* discreta, conseguiu-se implementá-la numa rede RSSF que é contínua e não é discreta. Com base nesta implementação, sugera-se um protocolo para gerenciamento de armazenamento. Este trabalho é organizado da seguinte forma:

O Capítulo 3 inicia com um breve resumo das aplicações em RSSF. Apresentam-se detalhes importantes de diferentes arquiteturas de RSSF e explicam-se as questões de consumo de energia nas RSSF. Este capítulo é concluído com um exemplo de cenário para RSSF. Este exemplo é importante para a compreensão das necessidades do modelo de agregação sugerido neste trabalho.

O Capítulo 4 fornece embasamento sobre grafos *expander* e introduz-se o grafo original do Gabber e Galil, que foi desenhado para uma rede *grid* que é *discreta*. A seguir, desenvolveu-se o grafo tipo *expander contínuo* para a implantação geográfica de sensores. Definiu-se uma *abordagem contínua discreta* 2-D (Naor & Wieder, 2007) que relaciona os nós de sensores com o grafo contínuos tipo *expander*.

O Capítulo 5 inicia com a definção da camada da rede (protocolo GPSR - *Greedy Perimeter Stateless Routing* (Karp & Kung, 2000)) e o espaço contínuo dos endereços geográficos (diagramas de Voronoi (Berg, 2000)). A seguir, definiu-se a maior inovação deste trabalho, um Passeio Determinístico "Aleatório" (*Deterministic "Random" Walk*) que se baseia neste grafo geográfico tipo *expander* sugerido (Chapter 4), o protocolo GPSR e os diagramas de Voronoi. Este passeio é usado como um algoritmo que descobre a RSSF, e que possibilita com que sensores individuais interajam com outros.

Capítulo 6 lida com modelos de tráfego para RSSF que não são uniformes. Explica-se aqui, a

importância de um modelo como este para a análise de desempenho da Cadeia de Armazenamento distribuído (Distributed Storage Chain DSC) sugerida neste trabalho. Definiu-se o termo *lifetime* da RSSF considerando aos números dos eventos gerados no campo alvo. Desenvolveu-se a função de probabilidade que prediz a vida útil de uma RSSF, baseado no modelo de tráfego sugerido.

No Capítulo 7 formulou-se o protocolo distribuído para otimização de armazenamento em RSSF. Isso é, um protocolo que consegue maximizar a utilização do armazenamento total da RSSF que implantada no ambiente que gera as atividades na forma não-uniforme (Capitulo 6). A ideia é usar Cadeias de Armazenamento Distribuídas (*Distributed Storage Chains (DSC)*) para localizar sensores com capacidades de armazenamento disponíveis.

Capítulo 8. Através deste trabalho são apresentadas os resultados de nossas simulações. Em cada capítulo, são demonstrados os resultados específicos sobre os seus tópicos. Entretanto, no Capítulo 8 foram incluídas a maior parte de nossos resultados enquanto foram estimados os desempenhos dos nossos protocolo. Finalmente, no **Capítulo 9** são apresentadas nossas conclusões e trabalhos futuros em potencial.

Chapter 3

Wireless Sensor Networks

"Any fool can make things bigger, more complex, and more violent. It takes a touch of genius - and a lot of courage - to move in the opposite direction"

Albert Einstein (1879 - 1955)

3.1 Introduction to WSN

Today we have the telescope, which enables a deeper understanding of astronomy, the microscope, which brings bacteria into view, and the satellites that survey the Earth's surface, expanding what we can perceive and measure (Culler et al., 2004). Wireless Sensor Networks (WSNs) have made a major breakthrough on the concept of visualization. Instead of simply observing some phenomena through an unique source (e.g., telescope, satellite etc.), sensory data is combined from disparate sources and processed into richer and more accurate report for the monitored phenomenon. WSN are composed of small computing devices, which collect numerical measurements, in order to map physical phenomena that cannot be observed through trivial means. In other words, WSNs use sensors measurement to produce a richer and more enhanced picture that is invisible to the "naked eye".

WSN have been intensively studied on the last decade and became an active research topic (Jardak, Riihijärvi, & Mähönen, 2010). Advances in hardware technology have led to a dramatic reduction in size, power consumption and cost of WSN technology. The goal is to make sensor nodes as small, inexpensive and easy to deploy as possible. Good examples, that may also spark the imagination, are Berkeley's *"Smart Dust"* and *"Micromechanical Flying Insect" (MFI)* projects, led by Professors Pister and Kahn (Kahn, Katz, & Pister, 1999). It is easy to understand by their titles that these projects explore the limits on size and power consumption (Figure 3.1 (A)) for autonomous sensor

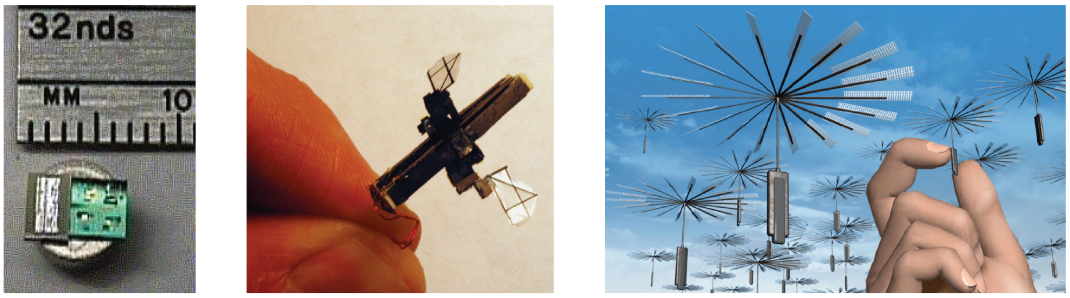
nodes on large scale WSN . Their idea is to be able to implement as small as millimeter-scale nodes that can remain suspended in air (MFI project) and even be self-powered using solar receptors (Figure 3.1).

Most of the existing works on WSN in the literature is focused on design questions such as: protocols (Karp & Kung, 2000; Dimakis, Sarwate, & Wainwright, 2006), architecture (Hung-Yu et al., 2005; Dietrich & Dressler, 2009; Wu, Chen, & Das, 2008) and how data from a single sensor node should be dealt with (Ratnasamy et al., 2003; Silberstein & Yang, 2007; Liao & Wu, 2008). As we will show in this chapter, the design of WSN consists of many parameters. Moreover, since WSN and in particular large scale WSN are still immature and under heavy development, there is no formal model or unique architecture to describe them. In this chapter we give a brief survey to WSN and their challenges in order to position our work within the WSN scientific community. We also provide some technical background on WSN that connect between our work and WSN on the architecture level.

3.2 WSN Applications

3.2.1 Characteristics of Sensor nodes

A sensor node is an independent component that can record and transmit detailed information about its surroundings. Actually, this independent component can be thought as a transducer that transforms one form of energy into another in order to quantify it. When a drastic change is measured, a sensor node can indicate events within its own region on a specific time. When events that were sensed on different regions are combined and processed together, we can analyze and study phenomena on the target area where the WSN was deployed. There are various WSN applications



(A) (B) (C)

Fig. 3.1: Micro-sized sensors (A) Smart Dust sensor (1999) (B) Micromechanical Flying Insect Sensor (MFI) and (C) Illustration of MFI.
Images taken from Prof. Piste's web site (Pister, 1999) and (Bermeo, 2010).

and they are dependent on various parameters. We list the most important parameters bellow:

Large Scale and Low-cost. Large scale WSNs consist of thousands sensor nodes. Therefore, reducing the cost of each sensor node to the minimum is very important. Due to their large scale, the reduction of every cent on the cost of a single sensor has a significant implication on the price of the entire WSN.

Small Size node. Sensor nodes are generally small in size. The idea is to enable them to merge with the monitored environment. There are some applications (e.g., military) that even demand that the WSN be camouflaged (Michael Winkler & Barclay, 2008). Due to their low cost and small size, the resources of sensor nodes are very limited i.e., the quality and the size of their components suffer from their restrictions in size and cost.

Energy efficiency. Energy is probably the scarcest resource for sensor nodes. This resource must be utilized properly in order to allow the functionality of sensors for long period of time. For that reason, sensor nodes are designed to be able to sleep and awake independently (Lewis, 2005; Ajay Jangra & Priyanka, 2010). Moreover, depending on their application and costs, sensor nodes can even be self-powered (solar cells) (Kahn et al., 1999). Even in that case, sensors should efficiently use their energy, consuming only a limited volume of energy during a day cycle (estimated by an order of 1 Joule or power consumption of less than $10\mu W$ (Kahn et al., 1999)).

Distributed Components. Since each sensor is in charge of an autonomous region, the WSN can be thought as a distributed sensing mechanism. On large scale WSN it is very important that sensors could work distributedly in order to reduce the communication volume. Therefore, network algorithms should be designed on a completely distributed manner, without counting on any central node or server. Moreover, on large scale networks it is inefficient and unnecessary that all sensors be familiar with each other (infrastructureless networks). In order to route data on infrastructure-less WSN, sensors discover their neighbors by exchanging information. Then, using distributed algorithms (e.g., GPSR (Karp & Kung, 2000)) they can determine how to route the data.

Wireless and Multi-Hop. Sensor node radio consumption is about 20 mW on average and their range typically is measured in tens of meters (Culler et al., 2004). Sensor nodes can route data to wider ranges using a multi-hop routing. Moreover, it is not feasible for each node to reach an end user or a base station (sink). There are several different approaches for delivering packets to a base station (Hung-Yu et al., 2005; Wu et al., 2008; Heinzelman et al., 2000).

3.2.2 Classification of WSN by the Volume of their Traffic

Depending on the type of events that sensor nodes sense and measure (hardware), WSN can be used to monitor almost every possible desired phenomena. Generally, sensor configurations are defined by the application, implemented by the WSN. These configurations should be designed care-

fully, since each parameter has its implications on a large scale. Generally, one have to consider that all sensor nodes are subjected to the tradeoff between their size and cost. We begin by categorizing WSNs into three groups of applications according to their traffic scale (Jardak et al., 2010):

- **Vehicular Sensor Networks.** Vehicular sensors will be used in the future to collect real-time data on road conditions. Their applications would be safety, collision detection, traffic control (reduce traffic jams), etc. Generally, vehicular sensors are located inside the car, enjoying an (almost) unlimited energy supply. Their data packets are estimated to be routed through three hops before reaching a sink station that is located on the side of the highway. In order to calculate the potential traffic load of such WSNs, we use the following estimations. The total number of vehicles in 2009 was estimated as 744×10^6 vehicles. In the literature, a node traffic volume is estimated to be 524 b/s. Therefore, the overall potential traffic of vehicular sensor networks is estimated to be 390Gb/s.

- **Patient Monitoring.**: This kind of networks is based on sensor devices that are planted inside patients and measure his/ her health condition (for example heart bit, body temperature etc.). The design of such sensors should take under series considerations the energy consumption since recharging these sensors may be extremely difficult. However, transmission to a sink station is estimated to be one hop. Therefore, the tradeoff in this case is in a higher number of sink stations. We estimate the traffic load of such WSN as follows. According to the world population (2009) the number of patient that may benefit from that kind of monitoring is estimated by 32.3×10^6 patients. Each patient is estimated to have a traffic volume of 23 b/s with a total WSN traffic volume of 750 Mb/s.

- **Environmental and Wildlife Monitoring.** Maybe the most famous environmental monitoring example is the one of storm chasing. In this case, sensors are dropped into a hurricane (for example) and record its characteristics by means of thermal, humidity wind and speed. When information is sent to the sink station we may study and analyze that meteorological phenomenon.

  The most critical problem on environmental monitoring is the size of the target area that we wish to monitor. On contrast to vehicular sensor networks that are deployed within narrow highways and on contrast to patient monitoring that is deployed on urban regions, the most complicated problem on environmental monitoring is the deployment of sink station. On environmental monitoring we deploy sensors on environments that it is difficult or impossible to reach. Therefore, environmental monitoring WSN suffers both from the lack in sink stations and from the distance between the sink station and the sensors. In addition sensor nodes have limited battery sources and suffer both from the limit in energy and a long multi-hop routing.

We roughly estimate the traffic volume as follows. In total, land covers $149 \times 10^12\ m^2$ from the total surface of the earth. If we roughly assume coverage range of $200\ m$ in radius for each sensor node, we get a total number of 596×10^6 sensor nodes. We assume the presence of an access point for a region $2.6 \times 10^3\ m$ in size. Therefore, we can estimate the number of hops that a packet has to travel before it reaches the sink station as 13 hops. In addition, we consider two different sampling frequencies: a quick sampling reading happens every 5 sec with rate 13.4 b/s per sensor node and a slow sampling rate of a single reading in every 15 minutes with rate 0.07 b/s per sensor node. Therefore, we obtain total traffic volume of 8 Gb/s and 42 Mb/s respectively.

Environmental and wildlife monitoring WSN design is of the higher complexity because of the limitations described above. In the literature, a common solution for large scale WSN is using distributed algorithms in order to coordinate sensor nodes and allow them to share their resources. Environmental and wildlife monitoring applications are varied. Since our work is classified in this category, we will describe it more into details. In the next section we will present some specific applications for environmental and wildlife monitoring. At the end of this chapter we will describe a specific scenario where storage aggregation solution is needed. This is very useful for the understanding of this work and our proposed protocol.

3.2.3 Generations of Sensor Nodes

One area, commonly cited as a primary use of sensor networks, is for military benefits (Michael Winkler & Barclay, 2008). Like other scientific areas, WSN was funded and motivated by the military research. Military applications such as detection, identification, analysis of enemy movements, are the basis for the academic research that enables civil applications. Similarly to the evolution of mobile cellular technologies, we describe the evolution of military sensor devices in terms of generations:

First generation WSN (1GSN) consist of individual sensor devices. Deployment is via manual placement. First generation networks are fully pre-configured, which means zero tolerance to environment changes after deployment. Access to information is via manual retrieval of the device itself, or long-range point-to-point communication links. Industrial manufacture of 1GSN is available by companies such as SenTech, Textron and Lockheed Martin (Michael Winkler & Barclay, 2008). They have systems with variety of sensors (including seismic, acoustic, infrared) which transfer their data directly to a sink station.

Second generation WSN (2GSN). On 2GSN sensors work in collaboration to cover an area. That is, in contrast to 1GSN, where sensors are individual components. Typically, the network contains a small number of sensors (3 or 4), communicating with a sink station. Most 2GSN are manually de-

ployed and are strongly dependent on pre-configuration. Generally, there are very few 2GSN systems on the market. WSNs, up to second generation, do not truly implement multi-hop routing in nature and cannot be deployed on a large scale.

Third generation sensor networks (3GSN). These networks are the subject of our work. These networks are distributed, flexible and scalable. Sensors communicate between themselves for two main reasons: communications with a user or base station and multi-hop routing and in-network processing (data aggregation data fusion and multi-hop routing (Ratnasamy et al., 2003)). 3GSN may contain as many as tens, hundreds or even thousands of sensor nodes (depending on the application, the number may reach an extreme value of millions (Akyildiz et al., 2002)). Deployment can be hand-emplaced (like previous generations) or remotely air-dropped (generally, on a uniform distribution). Also, since these networks are designed to be deployed on a large target area, 3GSN sensors are usually equipped with GPS or utilize some kind of localization technique (Khan, Kar, & Moura, 2009). Therefore, a sensor node can associate its events with a geographic location. As we will demonstrate in this work, on large scale WSN even the sensor ID is dependent on its geographic location (geographical routing (Karp & Kung, 2000)).

Although the literature (including this work) intensively deal with different aspects of 3GSN, there are only few known networks and many of them appear to be immature (Michael Winkler & Barclay, 2008). As to military applications, we speculate that large sensor network may already exist and remain confidential for different reasons (types of applications, high costs etc.). On the next section we will describe some known application.

3.2.4 Examples for Environmental and Wildlife monitoring

Sensors are being integrated into structures, machinery, environment, wildlife research and military. WSN could provide tremendous benefit to society (Lewis, 2005). Their goal is to improve our lives by trying to predict some phenomena that could have been handled better when we are prepared (e.g., volcanic activity) or when we can interfere (the condition of a bridge). Their advantages include: reduce environmental catastrophes, conservation of natural resources, wildlife research, vehicular monitoring, improved manufacturing productivity, pollution monitoring en enforcement, improved emergency response, and enhanced homeland security. On this section we give some specific examples of WSN (generally 3GSN) with environmental and wildlife applications.

Agriculture. WSN can be involved on the agriculture: sensing of pesticide, soil moisture, PH levels, temperature and humidity measurement, etc. Thus, agriculture WSN may offer habitat exploration of animals and insects, forest fire and flood detection. Marine WSN includes monitoring of fish for agriculture. AgroSens (http://agrosense.org/) is an example for a WSN that monitors agriculture using ZigBee (Lewis, 2005) based sensors. Sensors collect information about changes in whether,

disease, insects and optimize irrigation based on soli conditions.

Seismological and geophysics activities. Maybe one of the most famous application of sensor networks in in seismological activities. Sensors are used to measure and monitor seismic noise in order to predict earthquakes or volcanic eruption on known regions (regularly wired sensors). In the industry WSN can be deployed over a target area in order to monitor controlled seismical noises and map the geophysics layers of the earth for the exploration of minerals.

Heavy industrial monitoring. Industrial applications require highly reliable operation in harsh environment, in warehousing, industrial applications, manufacturing monitoring, industrial automation and factory process control (Ajay Jangra & Priyanka, 2010). WSN may monitor structural changes for large industrial factories, temperature changes for heavy machines and industrial systems, pollution monitoring for chemical factories, nuclear activities (also for military services), etc.

Wildlife monitoring - Combination of acoustic, thermal, mechanical and motion sensors can map animal activity on bushy or deserted environments where human interference is difficult, impossible or unwanted. Moreover, WSN can be used constantly, continuously and simultaneously on many different location, resulting in a substantial data collection, which can be processed to monitor wildlife activities.

Fig. 3.2: ZebraNet project (Image taken from the ZebraNet project on Princeton University).

An example for wildlife monitoring WSN is the ZebraNet project (Luo et al., 2007) conducted by Princeton University in Kenya. This project implemented many algorithms and techniques that exist in the literature such as multi-hop routing, data fusion, power management, GPS (or localization such as (Khan et al., 2009)) etc. Their goal (apart of the engineering challenge) was to study about the migration and lifestyle of zebras. Previously, scientists were using a transmission collar (Figure

3.2). Then, data was sent to a flying sink station that was traveling above the zebra territories. In this project, intelligent tracking collars, equipped with GPS, solar cell and short range transceivers, where were placed on a sampled set of zebras. These collars where capable of recording GPS position every 3 minutes, with information about speed and Sun/ Shade details.

The main contribution of this project was in the area of Telecommunications. These short range transceivers where not designed to send information to a sink link. Instead, they were designed to communicate between themselves. Whenever two zebras where encountered, data was exchanged between their collars (sensors) and processed into events. Then, data was gathered by data mules (mobile devices that come in contact with the sensor - with the zebras). Notice that since data is simply fused and exchanged between sensor nodes, information about all zebras can be withdraw from a single (or only few) ZebraNet collars.

3.3 Network Architecture

Wireless networks in general are divided into infrastructured and infrastructure-less networks. An infrastructured network consists of wireless nodes with a network backbone (e.g., cellular networks). An infrastructure-less network consist of independent wireless nodes, which distributedly manage to establish a dynamic topology (Ajay Jangra & Priyanka, 2010). Due to their infrastructure-less nature and due to the limitations in resources of each sensor, WSN architecture is very complex. In this section we will describe the most important aspects of the architectural design of WSNs.

Homogeneous Vs. Heterogeneous WSN. As we mentioned, sensor nodes differentiate from each other by the type of events that they can sense. Heterogeneous WSNs consist of the same sensor types but with different resource capacities. For example, heterogeneous WSN may contain sensors with different energy levels and battery supply. Therefore, sensors with higher energy levels could be used within the WSN as routers or data aggregations (Dietrich & Dressler, 2009). Logically, the tradeoff for developing sensors with higher resource capacity is between their size and costs. Thus, if we fix the resource capacities, the smaller the sensor node the more expensive it will be. Homogeneous WSN are networks in which all nodes are identical and have same resource capacities. Their challenge is to resolve resource capacity limitations via distributed algorithms, for example (Ratnasamy et al., 2003; Heinzelman et al., 2000; Luo et al., 2007) and as we describe in this work.

Many-to-Many Vs. Many-to-One. A wireless sensor network is composed of a few sink stations and a large number of sensor nodes that are densely deployed in a sensing environment (Liao & Wu, 2008). In the literature we distinguish between many-to-may and many-to-one network architectures. Many-to-one WSN describe a paradigm where sensors communicate with a single sink station (Hung-Yu et al., 2005). Generally, the idea is that events that are observed by sensor regions are immediately

transmitted to the sink station and processed by it. Their advantage is in real-time monitoring (e. g., sensitive military application). On many-to-many WSN, sensors communicate between themselves in order to allow *data fusion* (Akyildiz et al., 2002; Ratnasamy et al., 2003) within the network. Therefore, the data is stored and processed within sensor nodes and then, accessed by end users. Since the data is fused and processed within the WSN, an end user receives smaller data packets, which contain processed information (in terms of bits). Therefore, each retrieval is cheaper (in terms of energy) than on the many-to-one paradigm where every piece of information is sent back to the sink station (Luo et al., 2007; Kahn et al., 1999).

Distributed Algorithms. Since the sensor nodes themselves are physically distributed, it is not unnatural to design the WSN with distributed algorithms (Estrin, Govindan, & Heidemann, 1999). On many-to-many WSN we economize energy by processing and keeping the information inside the WSNs. However, it is very difficult for a single (or few) sensor nodes to administrate the network. In particular, in homogenous and large scale WSN it is even more difficult for a single sensor administrate and take decisions in the network. Therefore, the common solution is to use distributed algorithms that runs simultaneously on every sensor node. For example, LEACH (Low-Energy Adaptive Clustering Hierarchy) (Heinzelman et al., 2000) protocol define clusters of sensor nodes (similarly to a distributed *leader selection*) in order to solve the energy problem. In this work we solve the storage problem by a distributed algorithm.

Geographic. Large scales WSNs face the challenge of handling large number of sensor nodes and coordinating among them. Therefore, sensor nodes may not have global identifications (ID) because of the large amount of overhead (Akyildiz et al., 2002) (e.g., Smart Dust (Kahn et al., 1999)). A more practical solution is to use the geographical addresses (x, y coordinates) of sensor nodes as their identifiers. Accordingly, on a geographic oriented network sensors do not have to be aware of the existence of all sensor locations (or identifiers). Geographical routing techniques such as GPSR (Karp & Kung, 2000) enable routing packets between sensor nodes using a distributed protocol. Moreover, it is possible to exploit the geographical addresses for applications such as: geographical hash table (Karp & Kung, 2000), energy management and clustering (Heinzelman et al., 2000), resource allocation (Liao & Wu, 2008) and in our case, distributed storage.

3.3.1 Deployment of Sensors

An important architectural aspect of WSN is the way that sensor nodes are deployed. Sensor nodes are deployed either inside a monitored phenomenon or very close to it. Regularly, they are deployed on places, where human interference is difficult. They can be either thrown as a mass or placed one by one in the sensor field. Once the WSN is deployed, sensor nodes communicate among themselves and with an end user or a sink station. During a communication session sensors can also

study about the aggregated resources conditions of the WSN. Therefore, sensor nodes can indicate regions that may become coverage holes. Coverage holes are regions whose sensors have reached to their resource limits and, therefore, can not continue monitoring their regions. Akyildiz et al. differentiate three deployment phases (Akyildiz et al., 2002):

Pre-deployment and deployment. Sensor nodes can be either thrown in as a mass or placed one by one in the sensor field. They can be deployed by dropping from a plane, delivered in an artillery shell, rocket, or missile, and placed one by one by either a human or a robot. Heterogenous WSN can be implemented by deploying two (or more) sensor types uniformly over the same target area. Either way, in the design of WSN, the position of sensor nodes need not to be engineered or predetermined (Culler et al., 2004).

Post-deployment. After deployment, topology changes are due to change in sensor nodes position, reachability (due to jamming, noise, moving obstacles, etc.), available energy, malfunctioning, and task details (Hung-Yu et al., 2005).

Redeployment. Additional sensor nodes can be redeployed at any time to replace sensors that malfunctioning energy, malfunctioning, and task details (Sheng, Li, & Mao, 2006).

Large scale WSN are composed of large number of sensor nodes that are *densely* deployed. In this section we would like to define the density of a WSN. On the literature, the number of sensor nodes, their transmission radius and the size of the target area are variables that change from one work to another (see Table 3.1) based on WSN application. Moreover, large scale WSN are not common today and the majority of work in the literature is based on simulations for future use. Therefore, WSN parameters also differentiate since on many cases we compare with systems that we know today and try to estimate the future capacities and needs (for example, comparing radio range with Bluetooth standards (Culler et al., 2004; Lewis, 2005; Akyildiz et al., 2002; Heinzelman et al., 2000)).

3.3.2 The Density of a WSN

In the last section we discussed the deployment of sensors over the target field. However, in order to characterize WSN one should consider more parameters. As we will demonstrate in this section, a uniform deployment of sensors is the base assumption for more comprehensive measure, the WSN density.

In the literature, we find many WSN configurations that assumes uniform distribution of sensors over the target field. However, since both the number of sensors, the target area size and the radio range are parameters that vary, it becomes very difficult to compare between different WSN. Akyildiz et al. (Akyildiz et al., 2002) defines the density measure $\mu(r)$ for WSN. Basically, $\mu(r)$ denotes the number of sensor nodes that falls within the transmission radius of every single sensor in average. Let n be the number of sensors in the WSN, let r denote the transmission range of every sensor node

and let A denote the size of the target area (on our case $A = 1$), the density measure $\mu(r)$ is given be Equation 3.1:

$$\mu(r) = (n\pi r^2)/A \tag{3.1}$$

Note that the measure A/n is the node density, which represents the average region size (1 Node / (A/n) meter). We consider Equation 3.1 as the ratio between the transmission area and the node density ($\pi r^2/(A/n)$). On Table 3.1 we present different WSN configurations from the literature. We calculate their density $\mu(r)$ in order to demonstrate that even when the WSN parameters seems to be different, it is possible to compare between them. For example, GHT (Geographical Hash Table) (Ratnasamy et al., 2003) and GPSR (Greedy Perimeter Stateless Routing) (Karp & Kung, 2000) are two related works. Although their configurations seems to be different (radio ranges of 50 and 250 m), their WSN densities that we have calculated demonstrate that they are similar. In other word, they both represent an average of ~ 20 sensor nodes within a sensor radio range.

Notice that network density $\mu(r)$ takes under considerations all the parameters. Therefore, although the GHT and GPSR examples seems to be different in terms of radio size and Node density, these simulations consider the same network density (of ~ 20 sensor nodes within radio range).

Paper	Node Density (A/n)	Radio	$\mu(r)$
Mang. Sch. (Liao & Wu, 2008)	1 Node / 83 m^2	50 m	94.6
Many Agg.(Silberstein & Yang, 2007)	1 Node / 316 m^2	50 m	25
GPSR (Karp & Kung, 2000)	1 Node / 9000 m^2	250 m	21.8
GHT (Ratnasamy et al., 2003)	1 Node / 256 m^2	40 m	19.6
Energy hole (Hung-Yu et al., 2005)	1 Node / 200 - 100 m^2	20 m	6.3 - 12.6

Tab. 3.1: Examples of different WSN densities from the literature.

To conclude, we encourage using the WSN density measure that is ignored in the literature in order to compare between different simulation results. On Table 3.1 we give few examples of some WSN configurations that we will describe in this work. We calculate their densities and demonstrate how it can be used as a common language between different simulations. As we will show on the next section, the radio range of a sensor node can vary with respect to the transmission technique that it chooses (long range or short range for a multi hop routing). Therefore, the transmission technique is another consideration that we need to take under consideration while designing or comparing between WSNs.

3.4 Energy Consumption on WSN

The energy required for a sensor node to operate, usually come from a limited battery power. This limited power source should support the transceiver, GPS, CPU and the sensing unit modules (Note that GPS can be replaced by localization algorithms such as (Khan et al., 2009)). If sensor nodes run out of their energy, they could become inactive and create coverage holes in the WSN. Therefore, energy management is an important research topic in wireless sensor networks. Since our work deals with routing algorithms and since most energy consumption is due to the transceiver, we dedicate this section to the analysis of energy consumption in WSN. Later (on Chapter 8.9.2) we will use the equations we present on this section to estimate the actual energy consumption of our protocol. Note that our goal is to estimate the energy consumption derived by multi-hop

As we mentioned in Section 3.2.1, sensor nodes on large scale WSN have to communicate in a multi-hop routing manner in order to deliver packets to long distances. Moreover, if we think about the overall energy consumption of the WSN, sometimes it is more efficient to route a packet on a multi-hop manner. That way several messages can be compressed along the route into a single packet.

3.4.1 Transmission Costs

As we already mentioned, energy is a limited resource on WSN, which should be used wisely. Actually, every action in WSN has its costs, from CPU usage to the transmission of data packets. In this section we study and define the energy cost for the transmission of data packet on WSN. We use the simple first order radio model (Liao & Wu, 2008; Heinzelman et al., 2000) to describe the network energy cost for the delivery of data packets. While keeping the radio model as simple as possible, one has to bear in mind that a packet transmitted from sensor A to sensor B (or from some sensor to a sink station) is usually transmitted in a multi-hop fashion. We are interested both in the overall energy consumed by the WSN and in the average case (the reason will be explained into details in Section 5.5, when we will describe how we route messages).

Let E_{elec} $[J/bit]$ be the radio dissipate parameter (Heinzelman et al., 2000) of the transceiver. We use E_{elec} to estimates the energy cost for the usage of the transceivers per transmission/ reception of one data bit (transmit and receive blocks respectively in Figure 3.3). ε_{amp} $[J/bit/m^2]$ represents the energy required by the transmission amplifier for the propagation of one bit to a radius of one meter. That cost ($[J/bit/m^2]$) considers a reasonable SNR (assuming an r^2 energy loss on transmission). Figure 3.3 illustrates the radio model transmitting and receiving k data bits to the distance of d meters:

According to Figure 3.3 we can formulate the energy cost for the transmission E_{Tx} of k bit packet from a source to a receiver node that is placed d meters from the source. Equation 3.2 defines the

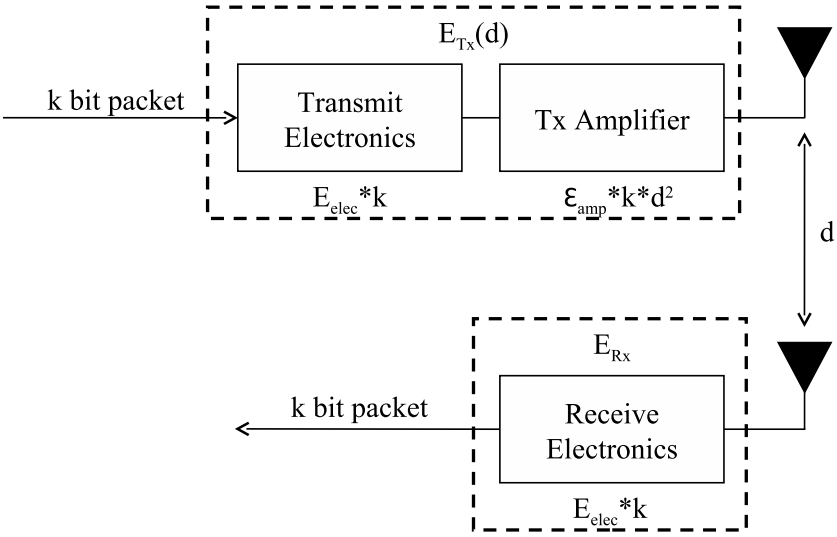


Fig. 3.3: Blocks Diagram for the Radio Model (Image taken from (Heinzelman et al., 2000)).

transmission cost E_{Tx}:

$$E_{Tx}(k,d) = E_{elec} \cdot k + \varepsilon_{amp} \cdot k \cdot d^2 \tag{3.2}$$

and Equation 3.3 defines the receive cost E_{Rx}:

$$E_{Rx}(k) = E_{elec} \cdot k \tag{3.3}$$

Notice that both transmitting and receiving operation have their costs, given by Equations 3.2 and 3.3 respectively. Therefore, WSN models, in general, should reduce the number of data packets as much as possible. Also, multi-hop transmission has even higher received costs since the packet is repeated on several sensor nodes. However, multi-hop routing drastically reduce energy consumption in the transmitter since amplification is of order d^2. Importantly, protocol and communication design should take under considerations the tradeoff between direct and multi-hop transmissions.

Our objective is to estimate the communications energy costs in many-to-many WSN. Therefore, we present a different analysis from (Heinzelman et al., 2000) where sensors communicate with a base station. However, as we will demonstrate, we get the same conclusion. Figure 3.4 shows a simple linear network where the distance between every two sensors is r. Sensor A transmits a packet of size k bits to sensor B. Notice that the distance between these two sensors is $n * r$ (we located $n + 1$ sensors). Therefore, if sensor A transmits a packet directly to sensor B, the total energy cost E_{direct}

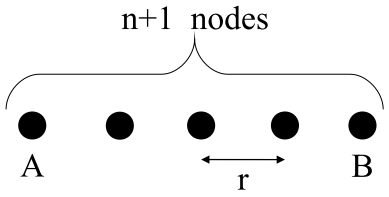


Fig. 3.4: Multi-hop routing in a simple linear network (Image taken from (Heinzelman et al., 2000)).

will be:

$$E_{direct} = E_{Tx}(k, d = n \cdot r) + E_{Rx}(k) =$$

$$E_{elec} \cdot k + \varepsilon_{amp} \cdot k \cdot (nr)^2 + E_{elec} \cdot k = \tag{3.4}$$

$$k(2E_{elec} + \varepsilon_{amp} n^2 r^2)$$

Note that on Equation 3.4 the parameter $2E_{elec}$ refers to one transmission (sensor node A) and one receive (sensor node B). If we chose to transmit the same packet on a multi-routing manner, the packet would be transmitted once by node A and be retransmitted $n-1$ times before being received by sensor node B. On multi-hop routing, every sensor transmit the message to its closest neighbor on the direction of the target (sensor B). In the this case, the Multi Hop energy E_{MH} is calculated as follows. The message would require n transmits and n receives to a distance r (where the last receive is on sensor node B):

$$E_{MH} = n \cdot E_{Tx}(k, d = r) + n \cdot E_{Rx}(k) =$$

$$n(E_{elec} \cdot k + \varepsilon_{amp} \cdot k \cdot r^2) + n \cdot E_{elec} \cdot k = \tag{3.5}$$

$$k(2n \cdot E_{elec} + \varepsilon_{amp} n r^2)$$

As described by (Heinzelman et al., 2000), there is no clear answer to which communication model is better (direct Vs. multi-hop). Moreover, they demonstrate that in some configurations the

directed model requires less energy while in others, multi-hop model is more efficient. Therefore, we intend to compare between direct transmission (Equations 3.4) and multi-hop routing (3.5) in order to study the tradeoff between these two methods. Our goal is to analyze the energy consumption on WSNs that use multi-hop routing. On Equation 3.6 we declare the condition on which direct communication consume more energy than multi-hop routing:

$$E_{direct} > E_{MH}$$

$$2E_{elec} + \varepsilon_{amp} n^2 r^2 > 2n * E_{elec} + \varepsilon_{amp} n r^2 \tag{3.6}$$

$$\frac{r^2 n}{2} > \frac{E_{elec}}{\varepsilon_{amp}}$$

As we can see in Equation 3.6, the packet size does not affect the tradeoff between these two methods. Therefore, the tradeoff is affected by the relations between the parameters: $E_{elec}, \varepsilon_{amp}, n$ and r. Notice that E_{elec} and ε_{amp} are constants that derived from the WSN configurations, while n and r are parameters that we should consider for each specific routing. In other words, for each route we should check the relation described in Equation 3.6 in order to decide the best routing method, then calculate the energy consumption. However, if we would like to change the WSN configuration (the relation between E_{elec} and ε_{amp}) we would have to repeat the simulation and select all the routing methods all over again. Therefore, the solution is to find some upper bound for communication costs.

In Section 8.9 we will estimate communications cost of our protocol, using the direct model energy consumption. However, while we will be using the direct model for energy estimation, many of our communication sessions are conducted via the multi-hop model. Similarly to (Heinzelman et al., 2000), we can vary the parameters E_{elec} and ε_{amp} in order to obtain a relation, for which all direct transmission consume more energy than multi-hop routing. In other words, we will exploit Equation 3.6, in order to estimate an upper limit on the energy consumptions (Liao & Wu, 2008).

3.5 Scenario

In this work, we deal storage capacity of sensor nodes. A sensor that had its storage capacity depleted should look for some other sensor in the WSN who has available storage capacity. In this section we present a scenario where sensor nodes should look for available storage in order to avoid coverage holes.

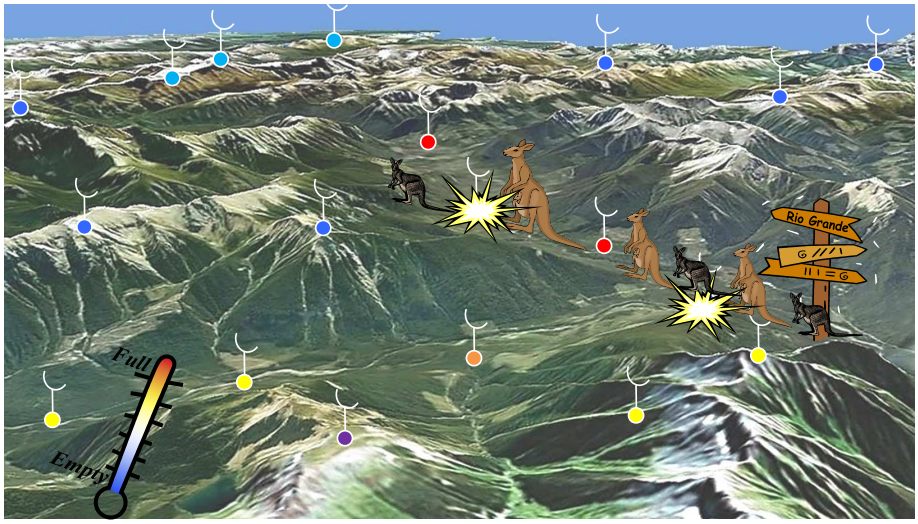


Fig. 3.5: Animal sighting scenario (Background is taken from google maps).

We consider a wildlife habitat monitoring WSN that is uniformly distributed over a large wild terrain (e.g., pre-deployed by an airplane). Sensors observe and record detailed measurements such as: temperature, acoustic, humidity and pressure measurements. A particular combination of several measurements might define some animal sighting, e.g. kangaroo-sighting, as shown by Figure 3.5. We describe a scenario of monitoring the kangaroo wildlife habits on an exotic environment. We use sensors that can detect and record combinations of measurements, referring to kangaroo-sighting. End user could then, send queries to some region on the target area and discover where and when kangaroo events were recorded. Sensors on this network communicate among themselves in order to achieve the higher level task (Culler et al., 2004) of wildlife monitoring. For example, multi-hop transmission (Hung-Yu et al., 2005; Karp & Kung, 2000), data fusion (Silberstein & Yang, 2007; Ratnasamy et al., 2003) etc. Similarly, our work is based on communication between sensor nodes in order to share their resources.

Figure 3.5 presents sensors that are uniformly deployed over a monitored environment. In other words, sensors are responsible for regions of more or less same sizes. We demonstrated a scenario, where most events were recorded by the main river. On Figure 3.5, we painted each sensor with a color that represents its storage consumption. Note that sensors, located by river, are painted red in order to demonstrate that they are about to deplete their storage capacity. Other sensors that had their storage capacity depleted are marked as out of use (exploded) since they cannot store additional events. Our conflict is that although the WSN as has a great amount of available storage (blue sensors), it contain a coverage hole on the main river. Moreover, this coverage hole is located on the most critical location (where kangaroo activity is high). Note that we use this figure as a scenario that gives motivation for

our work, this figure does not imply that we plan to compare our model with some existing wildlife monitoring model (nor that we know about a comparable model).

On this work we will consider the WSN storage capacity as a global resource that can be accessed by all sensors. Therefore, sensors that had their storage capacity depleted would have to communicate within the WSN in order to find available storage capacity (blue sensors). We will present a distributed protocol that defines by simple instructions how each sensor should search for available storage. This imaginary scenario could help to give the reader practical tools to understand our theory.

Chapter 4

Geographic Expander Graph

"...The blood, the sweat, the tears. Attribute to the strength, built through the years..."

Robert Flynn (Machine Head)

As we described in Chapter 1, in this work we develop a WSN protocol that is based on an expander graph topology. In this chapter we discuss expander graphs in theory and define a *static* expander topology for WSN. Later, on Chapter 5 we will use the definitions from this chapter to develop a *dynamic* expander topology that is the basis of our protocol.

4.1 Introduction to Expander Graphs

Expander graphs are graphs with a strong connectivity property, although their total amount of edges is small. Every set of vertices on an expander graph has a large boundary of neighbors. Therefore, in order to disconnect a large part of the graph, one has to sever many edges. In this work, we will demonstrate how a simple walk, or a simple sequence of links, over expander graphs rapidly visit most of their vertices. Since expander graphs are known to offer a good deterministic emulation of random behavior (Hoory, Linial, Wigderson, & Overview, 2006), we will compare our walk with a random selection of vertices from the graph. Due to their deterministic emulation of random behavior, expander graphs have application in cryptography, error corrections, combinatory, mathematics, computer science and obviously telecommunication.

Expander graphs are a growing study in science today (Naor & Wieder, 2007; Hoory et al., 2006; Alon et al., 2008; Camtepe, Yener, & Yung, 2006). In the past four decades, a great amount of research has been done in various fields to study this family of graphs (Hoory et al., 2006). Expander

graphs were first established on the early '70s (Bassalygo and Pinsker). The first construction of an explicit expander graph was published by G. A. Mergulus in 1973 on the Russian journal "Problems Information Transmission". However, his work did not contain a complete analysis. On 1979, Gabber and Galil (Gabber & Galil, 1979) presented an explicit construction of their expander graph. Their construction contained a complete analysis of its performances. Finally, in 1987, S. Jimbo and A. Maruoka used the Fourier theory in order to improve the former analysis.

Naor et al. (Naor & Wieder, 2007) present a novel approach for constructing a P2P (Peer To Peer) network topology, based on a *Distributed Hash Table*. Generally, their model divides the continuous 1-D interval $I = [0, 1)$ into non-overlapping subintervals, where each sub interval represents a process job. Then, they used a *Hash function* that relates between points in I. A link between two subintervals exists iff at least one pair of points between them are connected. These links between sub intervals are an analogy to the relation between processes. They regulate the job load between processors by transferring process jobs from heavy loaded processors to others, based on these links.

In this chapter, we present our overlay network that is based only on the *geographical* locations of sensors. We implement a two dimensional P2P network, as suggested by Naor et al. (Naor & Wieder, 2007). On our network, we have a link between two sensors iff at least one pair of points between their regions are connected. In this section, we will define an *expander graph* (Sec. 4.3) and demonstrate that it can be implemented *geographically* over the target area (Sec. 4.4). Accordingly, we will define our P2P Wireless Sensor Network (Sec. 4.5).

4.2 Definition of an Expander Graph

As we mentioned, expander graphs are graphs with a high connectivity, despite their low number of edges per vertex. Formally, we define an (n, k, d) expander graph, as a graph $G(V, E)$ that has $|V| = n$ vertices, at the most $|E| = k \cdot n$ edges and an *expansion rate* d. An expansion rate d is the ability for a vertex set $S \subseteq V$ to *rapidly* magnify itself to a larger set via their neighbors. Definition 1 (Hoory et al., 2006) specifies the expansion rate measure:

Definition 1 *Let $G(V, E)$ be an (n, k, d) expander graph and let S be a subset of vertices in V. We denote the set of edges that emanate from S to its complement $\overline{S} = V \setminus S$ as $\partial(S) = E(S, \overline{S})$. We define the expansion rate d of graph G as:*

$$d = \min_{0<|S|\leq \frac{n}{2}} \frac{|\partial(S)|}{|S|} \tag{4.1}$$

Notice that there is a restriction on S, $0 < |S| \leq \frac{n}{2}$. According to Definition 1 any subset of vertices S ($|S| \leq \frac{n}{2}$) can magnify (or expand) itself, based only on the edges that emanates from it

(close neighbors of every vertex in S). In other words, there would simply be links between S and $\overline{S}$. Moreover, it is guaranteed that S has a *large* neighborhood. According to the expansion rate d, the size of that neighborhood is at least a portion of size d from itself. Therefore, an (n, k, d) expander graph guarantees a *fast* expansion rate (d) for any subset $S : S \leq \frac{1}{2}$.

4.3 Gabber-Galil Expander Graph

In this section we will describe a specific expander graph that is called Gabber-Galil expander graph (Gabber & Galil, 1979). We begin by describing this graph, as described in their paper, for a grid network application where vertices are located on all the intersection of the grid. Then we will describe a continuous representation of their graph, which they used for their mathematical model. Later, we will use this continuous representation for our WSN application.

4.3.1 Expander Graph for a Grid Network

"Explicit constructions for linear sized super-concentrators" (Gabber & Galil, 1979) is a study, published in 1979 by Ofer Gabber and Zvi Galil of the Tel-Aviv University, Israel. Gabber and Galil describe an explicit construction for an expander graph G_n that is an $(n, 5, d_0)$ expander graph on a grid topology, where n denotes the number of nodes, 5 denotes degree of each node and $d_0 = (2 - \sqrt{3})/2$ is the expansion rate. The degree of a node, refers to the number of edges that emerge from it (Diestel, 2006).

Let $n = m^2$ (n is a natural number) and let $A_n = \{0, 1, \ldots, m-1\} \times \{0, 1, \ldots, m-1\}$ be a grid network of size n. A_n may be thought of as a combinatorial square torus (Figure 4.1), where each vertex is connected with its grid neighbors and whose edges are connected. In Definition 2 we define the Gabber Galil expander, for a set of n vertices that are placed on a $m \times m$ grid.

Definition 2 *A Gabber-Galil expander graph (Gabber & Galil, 1979) is defined as a bipartite Graph $G(U_n, V_n; E_n)$, where the relation between the sets of vertices U_n, V_n and the set of edges E_n are defined as follows:*

- $U_n = V_n = A_n$
- $\{E_n = (u, \sigma(u)) : u \in U_n, \sigma \in \{\sigma_i\}\}$

- *And the permutations σ_i's are given as:*

$$\sigma_0(x, y) = (x, y)$$
$$\sigma_1(x, y) = (x, x + y)$$
$$\sigma_2(x, y) = (x, x + y + 1)$$
$$\sigma_3(x, y) = (x + y, y)$$
$$\sigma_4(x, y) = (x + y + 1, y)$$

where the + operator is modulo m.

The most significant contribution of Gabber and Galil (Gabber & Galil, 1979) was their analysis of this graph. They have managed to prove and calculate the lower bound on the expansion rate d. In the next section we describe our overlay network construction, which is based on Definition 2.

4.3.2 Expander Over the Unit Square

On their work (Gabber & Galil, 1979), Gabber and Galil used a *continuous* graph representation, which is defined over the unit square $[0, 1)^2$. In other words, instead of the $m \times m$ grid, the continuous graph is defined for all the points in $[0, 1)^2$. We will demonstrate how a distributed sensor network can implement the Gabber-Galil expander graph as its network topology. However, since sensors are deployed on specific locations, our topology implements a *discretization* of this continuous graph.

We now introduce the Gabber-Galil expander graph: Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a *continuous graph* that is defined over the unit square $\mathcal{V} = [0, 1)^2$ and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the set of edges in $\mathcal{G}$. $\mathcal{E}$ is defined

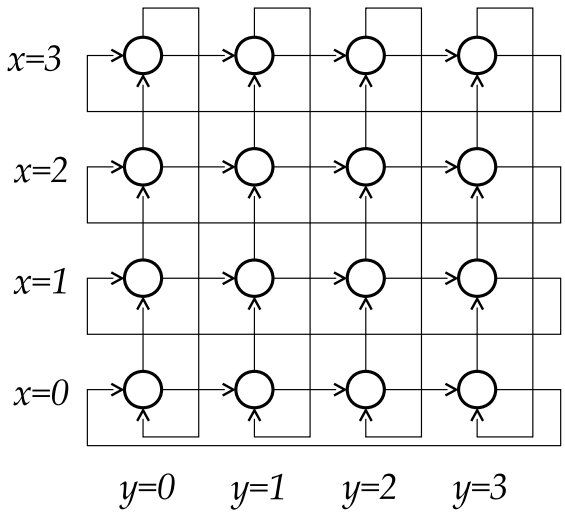


Fig. 4.1: Square torus whose edges are connected.

by the Gabber-Galil transform (Gabber & Galil, 1979) that relates each point in $\mathcal{V}$ with two outgoing links. With respect to their directions, we name those links East link and North link.

Definition 3 *The Gabber-Galil transform for a point* $\{x, y\}$ *in* $[0, 1)^2$ *defines the following two links:*

$$GG(x, y) = \begin{cases} East : \{x + y, y\} & (\text{mod } 1) \\ North : \{x, x + y\} & (\text{mod } 1) \end{cases} \tag{4.2}$$

After a modulo 1 operation (remainder from 1), all destination vertices remain within the unit square. We may consider the unit square as a square torus whose edges are connected (Figure 4.1). Therefore, an edge that exceeds out of the square's margins, spins from the other side of the unit square. An edge $(\{x_s, y_s\} \rightarrow \{x_t, y_t\})$ is in $\mathcal{E}$, iff $\{x_t, y_t\}$ is one of the two transformations of $GG(x_s, y_s)$.

Gabber and Galil were the first to provide a specific bound for the expansion rate d. Theorem 1 (Gabber & Galil, 1979) describes the basic motivation for using expander graph as a network topology for WSN:

Theorem 1 *For every measurable* $A \subset \mathcal{V} = [0, 1)^2$ *such that* $\mu(A) \leq \frac{1}{2}$*, where* $\mu(A)$ *is the Lebesgue measure of* A*, it holds that:*

$$\mu((GG_{\text{East}}(A) \cup GG_{\text{North}}(A)) \setminus A) \geq \frac{(2 - \sqrt{3})}{2}\mu(A) \tag{4.3}$$

According to Theorem 1, the Gabber-Galil expander graph has an *expansion rate* $d = \frac{(2-\sqrt{3})}{2}$. In other words, the two transformations of A (East and North) simply contain a *new* set of points (disjoined from A) that is proportional to A. Therefore, A has a relatively *large* neighborhood with respect to the expansion rate. We posit that a sensor network that construct its links between sensors according to the these two transformations (Definition 3) can also be an expander graph topology. Thus, such topology can be very efficient in terms of *discovering* new areas. As we mentioned, in many cases sensors are not aware of the existence of others. Therefore, that kind of topology could be used by sensors in order to discover partner sensors that can share their resources.

The *continuous graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined over an infinite set of vertices $\mathcal{V} = [0, 1)^2$, and therefore it also has an infinite set of edges $\mathcal{E}$. In fact, each point in $\mathcal{V}$ is associated with two edges from $\mathcal{E}$. We would like to relate the continuous set of edges $\mathcal{E}$, with a discrete and finite set of vertices S that represent sensor locations. Our idea is to design an expander graph topology over the actual geographical target area. Formally, we decompose the continuous $[0, 1)^2$ space into *regions*. Then, we define a discrete set of edges $E \subset \mathcal{E}$ between sensor regions (based on the locations of sensors). In

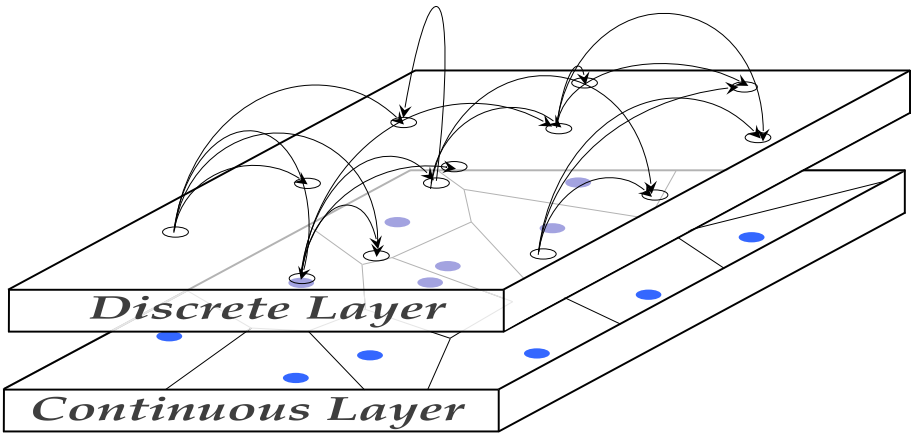


Fig. 4.2: Two dimensional Continuous Discrete Approach.

the next section, we will give a mathematical definition to sensor regions and define a *2-D continuous discrete approach* (Figure 4.2) that relate those regions by the discrete set of edges E.

4.4 Continuous Discrete Approach for a Geographical Target Area

We model our wireless sensor network as a set S of n sensors: $S = \{s_1, s_2, \ldots, s_n\}$, located inside the unit square $[0,1)^2$. Let $\{x(s_i), y(s_i)\}$ denote the coordinates of sensor s_i and we assume that each sensor is aware of its own locations (either using GPS or some kind of localization technique). Moreover, we assume that there are no two identical geographic locations for any pair of sensors. These are reasonable assumptions on sensor networks, because events are associated with the location where they were observed.

We divide the unit square into n non-overlapping *regions* that are based on the location of n sensors in field. Therefore, each point $p \in [0,1)^2$ is associated with its closest sensor $s \in S$. In other words, the region of each sensor $s_i \in S$ contains all the points in $[0,1)^2$ that are closer to s than to any other sensor in S. This kind of division that we described, is a well known concept from computational geometry (Berg, 2000) called *Voronoi Diagrams*. Each region is known as a *Voronoi Cell*. Formally, a Voronoi cell $\mathrm{Vor}(s)$ is defined as follows:

Definition 4 *Let S be a set of n sensors on $[0,1)^2$. $\mathrm{Vor}(s)$ denotes the Voronoi cell of a sensor $s \in S$. This region consists of all points $p \in [0,1)^2$ that satisfy the minimum Euclidean distance $\min_{s_i \in S}(|s_i - p|)$ for sensor s.*

We define our *two dimensional continuous discrete approach* as an approach that maps the continuous set of links $\mathcal{E}$ (between points $p \in [0,1)^2$) into a discrete set E that connects between vertices

$s_i \in S$. In Figure 4.2 we show the continuous layer that contains the continuous set of points in $[0,1)^2$ that are divided in regions. Since we defined two links for each point (Definition 3), there are infinite number of links on the continuous layer. On the discrete layer, we show only the discrete (finite) set of links that connect between two vertices (or sensor regions).

Accordingly, our continuous discrete approach associates the infinite (continuous) geographical address set (al possible x, y coordinates within the target area) with a finite (discrete) set of sensors (x, y coordinates of sensor nodes). The *continuous layer* $[0,1)^2$ represents our *continuous graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$. According to Definition 3 this graph contains an infinite number of vertices and edges. We define $E \subset \mathcal{E}$ to be the set of edges on the discrete graph $G(V, E)$. In other words, links in E connect two sensor nodes iff at least one pair of points between their regions are connected by one of the transformations (Definition 3). Notice that we partitioned the continuous layer into Voronoi cells (Definition 4) in order to demonstrate the region of each sensor. On the *discrete layer* we demonstrate only the links E that connect sensor regions. Therefore, the discrete layer actually represents the *discrete graph* $G(V, E)$ whose edges are the *discretization* of $\mathcal{E}$. We use the coordinates of edges in $\mathcal{E}$ to relate a continuous edge with their discrete form, with respect to the sensor region $Vor(s)$ that they connect. On computational geometry, the problem of finding the Voronoi cell of some coordinate$\{x, y\}$ is also known as the post - office search problem, or Nearest Neighbor search (Berg, 2000). Notice that we have only a finite number of edges on the discrete layer, that is due to the fact that there is only a finite number of sensors. Formally, define the discrete graph $G(V, E)$ as a sub - graph of $\mathcal{G}(\mathcal{V}, \mathcal{E})$:

Definition 5 *Let $S = V \subset \mathcal{V}$ denote a set of sensors and let $E \subset \mathcal{E}$ represent a discrete set of edges between vertices in V. An **undirected edge** $(s_i \leftrightarrow s_j)$ is in E if:*

$$\exists p, p' \; s.t \; (p, p') \in \mathcal{E} \wedge \mathrm{Vor}(p) = s_i \wedge \mathrm{Vor}(p') = s_j$$

where $\wedge$ refers to the logical AND.

In other words, two sensors are linked together if their corresponding Voronoi cells contain at least one adjacent edge in the continuous graph $G(\mathcal{V}, \mathcal{E})$. Our goal is to construct a network topology for WSN that enjoys the attractive characteristics of an expander graph (Definition 1). We wish to allow a *fast* interaction between sensor nodes, with respect to the expansion rate d (Definition 1). In the next section, we use a *continuous discrete approach* to describe our overlay network topology for WSNs. Since the original expander graph (Gabber & Galil, 1979) was designed for a very specific network where all vertex locations where fixed on a grid, our implementation for a random deployment of sensors is not trivial. That is, since the deployment of sensors over the target area is very different from the deployment of a fixed on a grid, our construction is not guaranteed to produce an expander

graph topology. In the following sections we demonstrate, via simulations, that our implementation has indeed the expander graph characteristics.

4.5 Static Overlay Network

As we mentioned in Section 4.3, an (n, k, d) expander graph is known for its expansion rate d, although it has only a linear $n \cdot k$ number of links. Another attractive characteristic of expander graphs is their *relatively low* diameter. Expanders are known to have a logarithmic $log(n)$ *diameter* size (Hoory et al., 2006), where n is the number of vertices in the graph. Diameter is defined as the maximal *distance* (in hops) between any two vertices in the graph (Diestel, 2006), where the measure distance refers to the minimum number of hops between a pair of vertices.

In this section we analyze the diameter of our overlay network construction. We distinguish between a *static* network that is built entirely before any event has occurred and a *dynamic* network that is built "on demand" only when a sensor node has its storage capacity depleted. We analyze the diameter by simulating a static overlay network. Later, in Chapter 5 we will demonstrate the expansion rate d for a *dynamic* overlay network.

Let S be a set of n sensors $s_i \in S$ and let $\{x(s_i), y(s_i)\}$ denote the coordinates of each sensor $s_i \in S$. We define a static network by constructing four links emerging from every sensor location $\{x(s_i), y(s_i)\}$. We define *immediate* links and *expander* as follows:

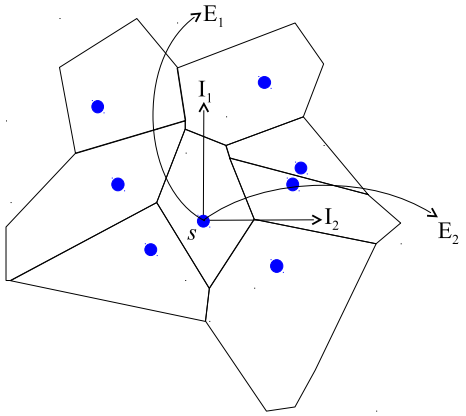


Fig. 4.3: Immediate and Expander links.

1. *Immediate links* are links between sensors that are geographically close to each other. In other words, every pair of sensors whose Voronoi cells are adjacent to each other, share an immediate link. These links are also known as the Delaunay triangulation (Berg, 2000). We use immediate

links in order to avoid isolated components within the overlay network. These links connected the whole network as Manhattan street network (Greenberg & Goodman, 1993). Note that the diameter for a Manhattan street network is of order $O(\sqrt{n})$ where n is the number of sensor nodes. In our case, we only use two immediate links. We connect each sensor with its immediate neighbors on the *East* and *North* directions. In Figure 4.3, edges I_1 and I_2 connect sensor s with its immediate links.

2. *Expander links* are links (or edges - E) from the expander graph $G(V, E)$. We use the two transformations from Definition 3 in order to construct two expander links for every sensor $s_i \in S$. These expander links connect each sensor s_i with two members from S whose Voronoi regions satisfies the two transformations E_1 and E_2. In Figure 4.3, edges $E_1 = \{x, x+y\}, E_2 = \{x + y, y\}$ connect sensor s with its expander links.

Note that although each sensor construct the four links described above, it does not significate that its degree is four. Since these edges are undirected, sensor nodes also "gain" edges that are *directed into their region*. Also, there is the possibility that two sensors would be connected by more than one link (multiple edges (Diestel, 2006)). In the rest of this section we analyze the resulted static topology in terms of diameter and number of links per sensor node.

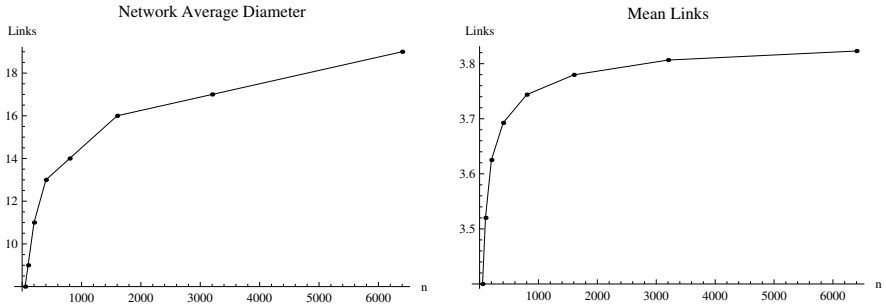


Fig. 4.4: (A) Diameter that scales with the network size. (B) Mean number of links.

Figure 4.4 demonstrates the simulation results for network sizes $50 \leq n \leq 6400$ where sensors are deployed uniformly over the unit surface. First, we establish a static overlay network by constructing the immediate and expander links from each sensor location. Then, we measured the diameter of the resulting topology using the Breadth First Search algorithm (Berg, 2000). Finally, we analyze the number of links per sensor node. As we mentioned, the immediate links guarantee both that the network is connected and that its diameter is of order $O(\sqrt{n})$. However, we expect an improvement in network diameter, since we have also added the expander links.

In Figure 4.4 (A), we show that only by adding the two expander links ($E_1 = \{x, x + y\}$ and $E_2 = \{x + y, y\}$), we achieve an improvement of the diameter size from a $O(\sqrt{n})$ to a $O(Log(n))$.

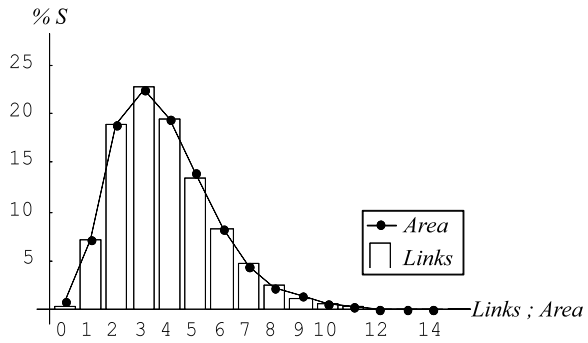


Fig. 4.5: Relation between region sizes and number of links for a 6400 sensor network.

Therefore, we have achieved an overlay network construction whose diameter scales with the network size, although every sensor constructs only four links. In practice, a sensor that had its storage capacity depleted can rapidly detect a sensor with available storage capacity within the network. For example, on a 6400 sensor network, a sensor that broadcasts its message along its 19 hops diameter can reach every sensor node.

Our overlay network contains $4n$ links in total that are directed into sensor regions. Note that the existence of multi-links (two or more links that connect the same pair of sensors) means that a multiple copies of the same message may be routed between pairs of sensors. In WSN, that means a waste of essential energy. Therefore, we would like to measure the number of links that are actually used in our construction. In Figure 4.4 (B) we measure the average number of links that are directed to every sensor node. Four links that are directed to sensor regions on average significate that there are no multiple links. In this figure, we can see that as the network size grows, the number of multi links decreases. The reason for that behavior is the size of sensor regions. For a small n, two links that are directed into *close* locations are a multi link with a high probability. In other words, they are directed into the same sensor region. As n grows, that region is divided into smaller regions. As a result, these two close link locations are directed into two regions of smaller size.

In order to complete our analysis we would like to verify that there are no bottleneck sensors within our construction. We give the *star* topology as a simple example that describes the problem of bottleneck sensors. According to a star topology we can achieve a constant diameter (of two hops) while every sensor constructs one link only. That is, all sensors select the same location (e.g. the region that contains for example the coordinate $\{0.5, 0.5\}$). Although that the resulted topology has a scalable diameter while having a constant number of edges, this topology is not an expander. In this case, the sensor whose region contain the coordinate $\{0.5, 0.5\}$ *"observes"* n links, and therefore forms a bottleneck in the network.

In Figure 4.5 we show the simulation results for a WSN of size $n = 6400$ sensors, on which

all sensors are uniformly distributed over the $[0,1)^2$ target area ($X, Y = \mathcal{U}(0,1)$). We simulated 120 different deployment and constructed a static overlay network for each deployment. As we mentioned there is a relation between a sensor region size and the number of links that are directed to it. In order to demonstrate that relation, we measured both the sizes of all sensor regions and the number of edges that are directed to each sensor region. We have plotted the results in the form of histograms.

In Figure 4.5, we show two measures on the x-axis: number of links and area classes. The bars in that figure represent the distribution of links that are directed into sensor regions. For example, the forth bar (from the left) means that $\sim 23\%$ of the sensors from S are located in regions that are directed by 3 links. We can see that the number of links that a sensor node observe vary between 0 and 14. Also, the majority of senors regions are directed by $2-5$ links. Moreover, the maximal number of links that are directed into some region is 14 which is a small number considering the network size. Therefore, our topology is free of bottleneck sensors.

In order to demonstrate the distribution of area on the same graph, we plotted an histogram of sensor region sizes on top of the linkage histogram (the dots curve). Notice that the number of bars on that histogram matches the maximal number of links (that we described above). Therefore, we measured the area of all sensor regions and categorized them into 15 groups: 0 - the smallest regions and 14 - the largest regions (according to the number of links $0-14$). In Figure 4.5, the relation between the number of directed links and sensor region size is very clear. In other words, the greater the region the more links that are directed to it. In the next section we will demonstrate that this behavior results directly from the transform (Definition 3).

Chapter 5

Distributed Storage Chains (DSC)

"All truths are easy to understand once they are discovered, the point is to discover them"

Galileo Galilei (1564 - 1642)

In this work we present a new approach for sharing the storage capacity of sensors in a WSN. We do so, by constructing links between sensor nodes in the form of an ordered list of sensors. We call that list a storage chain. A sensor s_i, who had its storage capacity depleted, initiates a storage chain $\mathcal{C}_i$ to the first link in its ordered list. That first link (a sensor node) will share its storage with s_i until its storage capacity is depleted also. Only then, s_i will address the next link on its ordered chain. In this chapter, we will explain how exactly a sensor selects its storage chain (or its ordered list).

In the last chapter we have shown how to construct a static network layer for WSN with a low diameter that is based on an expander graph (see the diameter definition in Section 4.5). That network layer was static in the sense that we systematically constructed links for all the sensors. In this chapter we will describe a dynamic network layer for WSN, which is the base of our new approach for storage aggregation in WSN. Instead of constructing links for each sensor, we will construct links by demand. In other words, a link is constructed between two sensors in the WSN, only when a sensor has its storage capacity depleted. This dynamic network layer can be seen as "subgraphs" of the Expander graph that we described in Chapter 4. We say that our storage chains are distributed (Distributed Storage Chains - DSC), since sensors generates them individually and independently. In other words, in contrast to a centralized approach that can map sensors that can share their storage capacity and construct storage chains accordingly, in our model each sensor construct its storage chain in a distributed manner.

To summarize, our DSC are characterized by the following characteristics:

- **Local**: Sensors should be familiar only with their immediate (local) neighborhood. DSC are implemented distributedly based only on local decisions.
- **Deterministic**: We say that DSCs are deterministic since they are based on the explicit Gabber-Galil construction (Section 4.3). Therefore, DSC can be *easily* reconstructed without maintaining routing tables.
- **Individual**: As we mentioned, DSCs are based on an explicit formula. That formula is based on the location of the sensor that initiates the chain. Therefore, different sensors construct different sequences of links.
- **Uniform**: We wish to maintain the WSN balanced in the sense that sensors who share their storage capacity are not concentrated in specific geographical regions. Each individual DSC *"samples"* the target area *uniformly* and avoid *"attacking"* specific regions.

Our storage aggregation model is designed for large scale networks, on which it is extremely inefficient for a sensor to study the whole network topology for a storage solution. Therefore, we suggest a distributed model, on which sensors take only *local* decisions that are based on their immediate environment only. Geographical location can be obtained by a GPS device or by some approximation localization technique (Khan et al., 2009). Moreover, we would also like to be able to reconstruct these chains *easily* without maintaining any information about their structure. Our model should avoid starvation i.e., there are no sensor groups that share their storage capacity more than others. A *non-individual* solution that directs all chains to the same geographical regions might constitute starvation. In order to prevent starvation, each individual chain should also select sensors within the network *uniformly*.

In this Chapter we begin by introducing the network layer that sensors use for communication (GPSR protocol). This introduction is an essential layer for understanding the relation between the storage chains and our expander graph (Chapter 4). Then, we explain the construction of chains that is based on our expander graph. Finally, we present general and experimental evaluations of these chains (Sec. 5.4 and Sec. 5.5 respectively).

5.1 Network layer - Geographical Routing

We model the communication network as a unit disk graph (Clark, Colbourn, & Johnson, 1991), on which two sensors can wirelessly communicate with each other if their distance is less than the

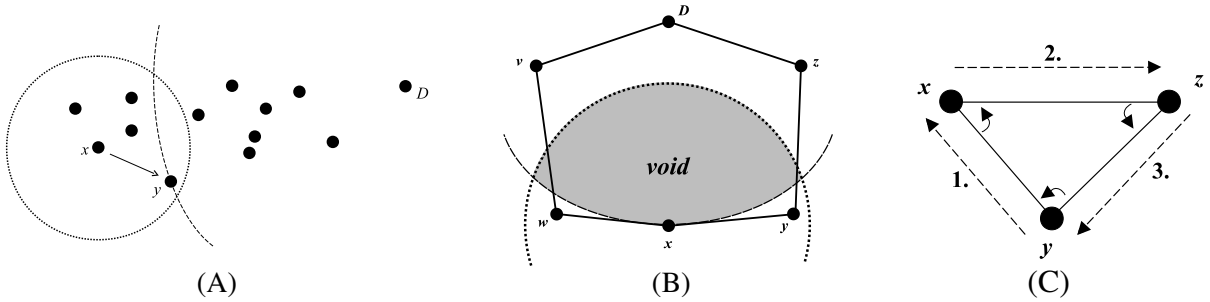


Fig. 5.1: GPSR Protocol (A) Greedy forwarding: x forwards to y, which denotes the minimal distance towards D. (B) Void example: x has no neighbor nearer to D. (C) Right-hand-rule: packets travel around the enclosed region. These images where taken from (Karp & Kung, 2000).

transmission range r. Furthermore, we assume that the network is connected, i.e. that r is large enough to guarantee that there are no isolated sensor components within $[0, 1)^2$.

Since not every two sensors in the network lie within the mutual radio range r, we must specify a network layer protocol that supplies routing instructions between every pair of sensors. For this purpose, we use a *geographical routing protocol*, which is based only on the geographical locations of sensors in field. As we already showed, the continuous nature of $\mathcal{G}(\mathcal{E}, \mathcal{V})$ requires that a message could be routed to a point within some sensor region. When a message is addressed to some destination that is not a specific sensor location, we say that the message was routed to the region of a particular sensor. We will define how a sensor can know that this message was directed to it.

We use the Greedy Perimeter Stateless Routing (GPSR) (Karp & Kung, 2000) geographical routing protocol as our network layer. This protocol can route a packet to any connected destination based on its geographical address. According to that protocol, each sensor must know its own location and the locations of its nearest neighbors (local information that is obtained by beaming (Karp & Kung, 2000)). This protocol operates on two routing modes, *greedy forwarding* and *perimeter forwarding*. In greedy forwarding mode, packets are progressively routed closer to their destinations. In other words, when a sensor receives a packet it forwards that packet to its neighbor with the minimal distance towards the destination (Figure 5.1 (B)). 5.1 (A)). Greedy forwarding mode fails when the packet can not be forward closer to the destination. We call this scenario a *void* or a *local minima*. For example, on Figure 5.1 (B), sensor x cannot forward the packet closer to D. The only sensors within the radio range of sensor x are sensors w and y. Notice that these sensors are more distant from the target D than sensor x. In that case, we call that radio range that cannot forward us closer to the target a void. Therefore, the solution is to use the perimeter forwarding mode, in order to escape from that void. On perimeter forwarding mode we use the right hand rule (Fig. 5.1 (C)) to route a packet along the faces of the void region (the polygon x, w, v, D, z, y in Figure 5.1 (B). Notice that

in this case, sensor node x would route the packet to sensor w although w's distance towards D is longer than x. On perimeter routing, sensors around the void send their distances from the target with for comparison. When the packet reaches a sensor that is closer to the destination (closer than the sensor that initiated the perimeter forwarding mode - x), GPSR returns to the greedy forwarding mode. In this case, routing can succeed, i.e. the packet reaches its destination, or fail, i.e. the packet has completed rotating along the void and did not find a sensor that is closer than x. In that case, GPSR dropped the packet.

An important property of GPSR is that if the packet has failed to reach its destination $d = \{x, y\}$, it either means that the destination sensor is disconnected from the source, or that no sensor is located at $\{x, y\}$. Ratnasamy et al. (Ratnasamy et al., 2003) reached the following conclusion: if the routing fails, it must fail (stop) at a sensor s that is the *Home-Node* (Corollary 1) for the geographical destination address. In Section 4.4 we defined the Voronoi region of sensor s_i to contain all the points p that are closer to s_i than any other sensor in S (Definition 4). Accordingly, we define a Home-Node of some point p as follows.

Corollary 1 *Let $p \in [0,1)^2$ and a set of sensors S located in $[0,1)^2$. $Home(p)$ denotes the unique sensor $s_i \in S$ for which $p \in \mathrm{Vor}(s_i)$. In that case we say that s_i is the Home node of p.*

In particular, when the protocol fails (stops), it must be on its perimeter forwarding mode. In that case, since that packet has already completes a full encirclement around the destination address, the destination address must be located within the Voronoi cell of sensor one of the sensors of the last encirclement. We say that each sensor is a Home node for all points on its Voronoi region. Therefore, the closest sensor to the target on the last perimeter mode must be the Home node for that destination.

Since our model is dependent on multiple routings of packets towards geographical addresses that are located within sensors regions, it is important to characterize the costs of that last encirclement. If those encirclement tend to be very large, that means a waste of important energy. On the next section we demonstrate that due to the characteristics of Voronoi diagrams, the last encirclement has a constant size with an average of six routings. We will demonstrate that characteristic by analyzing the average number of neighbors per sensor node.

5.2 Voronoi Diagrams - Six Neighbors per Sensor Node

On this section we give a brief description of the Voronoi diagrams which is oriented towards WSNs. Voronoi diagram is a special kind of decomposition of the metric space into to regions. It is determined by the distance measures within a specific set of points. A stronger definition of the Voronoi diagrams from computational geometry (Berg, 2000) can be given as: Let s_i, s_j be two points

on the plane. We denote the open half plane that contain the vertex s_i by $h(s_i, s_j)$, and the other half plane that contain the vertex s_j by $h(s_j, s_i)$. Notice that a single point $p \in h(s_i, s_j)$ if and only if $dist(r, s_i) < dist(r, s_j)$.

Definition 6 $\mathrm{Vor}(s_i)$ *is the intersection of n-1 half planes and, hence, convex polygonal region (intersection of convex sets (Berg, 2000)). Notice that some polygonal regions might be unbounded since half plane lines are infinite and beaming to infinity. To get more insight how does a complete Voronoi diagram looks like we will make one more assumption: all infinite lines, beaming to infinity are meet together. Imagine the Voronoi diagram lies on the top of a sphere and all infinite lines meet on its bottom (Figure 5.2).*

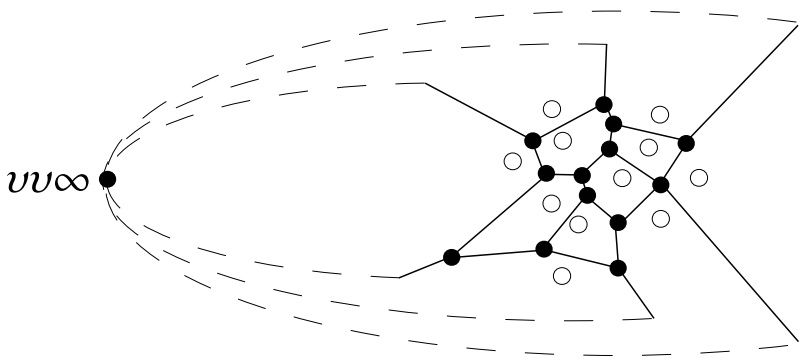


Fig. 5.2: Complete Voronoi Diagram.

Theorem 2 *Let G be a connected plane graph with n vertices, m edges, and F faces (regions). Euler's formula defines the following relation (Diestel, 2006):*

$$n - m + F = 2$$

Corollary 2 ***Every sensor has six close neighbors in average.*** *We fix $\mathcal{VV}_\infty$ to a Voronoi diagram in order to obtain a closed plane graph (Figure 5.2). According to Euler's formula, we have $n = v + 1$ (the extra vertices $\mathcal{VV}_\infty$) (Berg, 2000):*

$$v - m + F = 1 \text{ or } v = m - F + 1$$

Also, if we count the degree of every Voronoi edge we have:

$$\sum_{v_i \in v} deg(v_i) + deg(\mathcal{VV}_\infty) = 2m$$

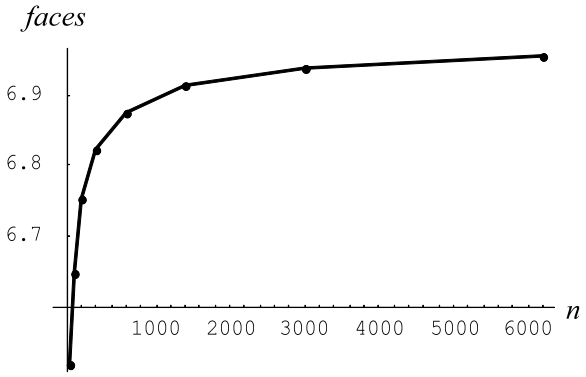


Fig. 5.3: Average number of faces per WSN size.

Explanation: *we count each Voronoi edge twice. Moreover, since the degree of every edge is at least 3 we have:*

$$2m \geq 3(v+1)$$

Substituting v *we get:*

$$2m \geq 3(m-F+2) \text{ or } 3F-6 \geq m$$

When we consider the number of Voronoi edges that encircle a node on the Voronoi diagram we count each Voronoi edges twice. Therefore, we have:

$$6F-12 \geq 2m$$

Therefore, in average if we divide the total number of Voronoi edges by the number of faces (nodes) we get:

$$O(6) \geq \frac{2m}{F}$$

Therefore, an interesting observation is that the number of Voronoi edges per faces is constant ($O(1)$), independent of the network size. In other words, every node shares its Voronoi edges with a constant 6 number of other nodes. In Figure 5.3 we show our analysis for number of neighbors per sensor node on a uniform distributed WSN. The configurations that we analyze in this figure are the same configurations that we will use later for network (Section 5.5) and protocol (Chapter 8) simulations. We can see that the number of faces (and therefore number of close neighbors) reaches the value of 7 sensors as the network size grows. Moreover, it seems that the graph reaches a constant value which matches our observations.

In Section 4.5 we constructed a static network. We did so, by calculating the two *expander links* of each sensor. Actually, we connected each sensor $s_i \in S$ with the two sensors that are Home nodes $Home(GG(s_i))$ for the two transformations of s_i (Definition 3). On the next section we formally define our storage chains and reinforce the relations between the Home node and our expander graph.

5.3 Construction of DSC

In the beginning of this section, we mentioned that our storage aggregation solution is implemented distributedly. We define the Distributed Storage Chain (DSC) as an ordered list of links that is initiated by a sensor s_i who had it storage capacity depleted. The sensors of that list share their storage capacity with s_i. We construct these chains via a *deterministic walk* over the expander graph that we described in Chapter 4. In this section we will define what is a deterministic walk, how do DSC are constructed and what is their relation with the expander graph. Later, we will formally define the construction of DSC by a distributed protocol.

We define a walk over some graph $G(V, E)$ as a sequence of vertices from V that are connected by links from E (Diestel, 2006). For example, if $\{(v_1, v_2), (v_2, v_3)\} \in E$ are two links in E then we can say that $w = \{v_1, v_2, v_3\}$ is a walk of size $|w| = 3$. That walk begins at vertex v_1 and on each *hop* we visit a new vertex. We distinguish between *random* and *deterministic* walks by the way that w is constructed. On Definition 7 below, we describe the deterministic construction of our DSC. According to this construction, these chains can be calculated and deterministically predicted for every sensor in our WSN.

Definition 7 *A **Distributed Storage Chain (DSC)** $\mathcal{C}_k(s)$ is a deterministic walk of $k+1$ geographical addresses over the **continuous** graph $G(\mathcal{V}, \mathcal{E})$. This walk is initiated by sensor $s = g^0$, located at $\{x_s, y_s\}$ and denoted by: $\mathcal{C}_k(s) = (g^0, g^1, g^2, ..., g^k)$ where every consecutive vertices are a link in $\mathcal{E}$. The i^{th} link (g^{i-1}, g^i) is given by:*

$$g^i(x, y) = \begin{cases} East : \{x + y, y\} \pmod 1 & \textit{for i even} \\ North : \{x, x + y\} \pmod 1 & \textit{for i odd} \end{cases} \tag{5.1}$$

note that due to the cyclic behavior of the $\pmod 1$ (modulo 1) operator, the east links might spin from the west also (and the north from the south). That is, depending on the values of the x, y coordinates.

When a sensor s had its storage capacity depleted it initiates a DSC ($s = g^0 = \{x_s, y_s\}$) and searches for new available sensors that can share their storage capacities. Notice that the DSC is dependent only on the location of the sensor who initiated it. We calculate the vertices $\mathcal{C}(s) = (g^0, g^1, g^2, ..., g^k) \in \mathcal{V}$ according to the Definition 7. In other words, every consecutive vertices (g^{i-1}, g^i) are connected by alternatively taking the *East* and *North* links of the Gabber-Galil transform (Definition 3). Since the sequence of vertices $\mathcal{C}_k(s)$ is deterministic, it neither has to be maintained as a routing table nor retrieved in order to be reconstructed. Therefore, in terms of energy, no

transmission is required for obtaining these sequences.

As we mentioned, we may think about this definition as a *continuous* layer construction (Figure 4.2). Therefore, a *continuous DSC* as we described above, specifies geographic locations rather than the location of sensors. As we described, the GPSR protocol can deliver packet to geographic locations within sensor regions. Formally, we define the Distributed Storage Chain (DSC) $DSC_k(s)$ as the discrete form of Definition 7. According to Corollary 1, we obtain a chain of sensor nodes by a discretization of $\mathcal{C}_k(s)$:

$$DSC_k(s) = Home(\mathcal{C}_k(s)) = (s, Home(g^1), Home(g^2), ..., Home(g^k)).$$

5.4 The Smoothness Factor of a DSC

We began this chapter with a definition that a DSC should be local, deterministic, individual and *uniform*. On the rest of this chapter we demonstrate how DSCs are *uniformly* distributed over the target area. As we described on Definition 7, a DSC is an ordered list of geographical addresses. We can think about that ordered list as a "search-function" that a sensor node uses in order to locate available sensors, who can share their storage capacities with it. We will demonstrate that these ordered lists, or "search-functions", are uniformly distributed over the target field in the sense that they do not "attack" in specific geographical regions. Actually, the best way to do so is to prove mathematically that a DSC that begins on any given point $\{x, y\}$ results in an ordered list of points, that are uniformly distributed in $[0, 1)^2$. However, we could not offer such a theoretical proof. Instead, in Appendix C we give the work that we only did so far, in formulating the theoretical problem we wished to prove. In this work, we will characterize the properties of DSC by simulations, instead of proving them theoretically.

In this section, we will study how these chains divide the unit surface. We make a general analysis that is not restricted to a specific WSN topology (or sensors deployment). We begin by separating the DSC from within the WSN. We select a single location (of sensor) in order to generate a DSC and analyze the distribution of its coordinates. We will show that with time, the way that DSC divides the unit surface converges to a constant ratio between regions on the unit surface (do not confuse with sensor regions). Then, we return to a WSN application, where sensors are deployed over the unit surface, and deduce that a DSC can serve as a uniform selection of sensors from the WSN. In other words, since we consider a uniform deployment of sensors, we can say that a DSC that uniformly selects geographical location, also selects their Home nodes uniformly.

We begin by selecting a sensor location $s = \{x, y\}$ and calculate its DSC $\mathcal{C}_k(s)$ of length k according to Definition 7. Then we calculate the Voronoi diagram of $\mathcal{C}_k(s)$ (see Figure 5.4). This diagram shows the way the Continuous DSC divides the unit surface. We can think about this kind of

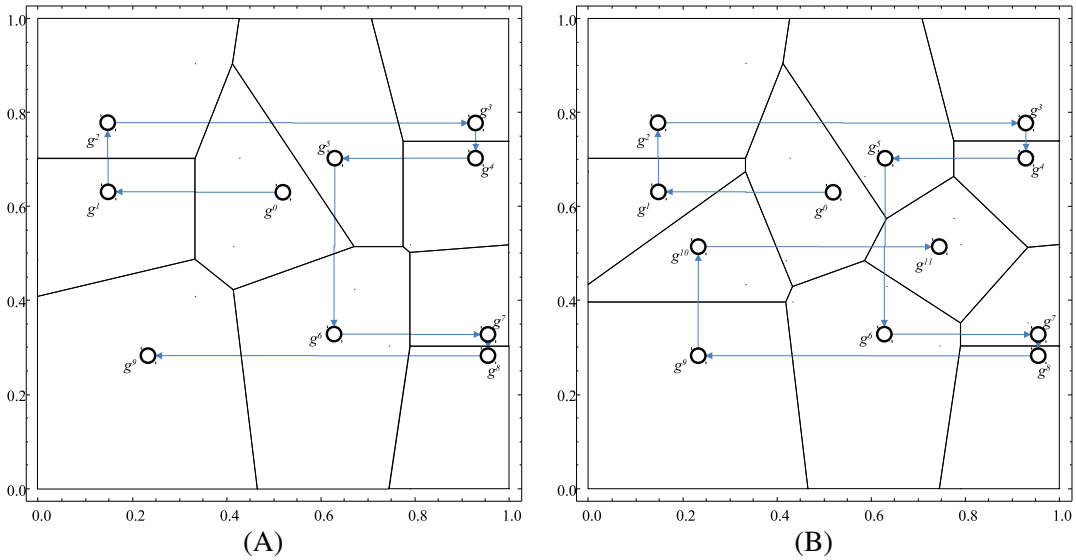


Fig. 5.4: Voronoi diagram for a Continuous DSC $\mathcal{C}_k(s)$ of size (A) $k = 9$ (B) $k = 11$.

division as the regions that the DSC visits. Notice that these are not sensors regions. We consider the ratio between the size of the largest and the smallest Voronoi cells as the *smoothness factor* (Naor & Wieder, 2007) of $\mathcal{C}_k(s)$. Formally, we define the smoothness factor as follows.

Definition 8 *Let $\mathcal{C}(s) = (g^0, g^1, g^2, \ldots, g^k)$ denote a set of vertices on $[0,1)^2$, we define its smoothness factor to be $\rho(\mathcal{C}(s))$, which is given by:*

$$\rho(\mathcal{C}(s)) = \ Max|\frac{\mathrm{Vor}(g^i)}{\mathrm{Vor}(g^j)}|, \ \ \forall i,j; \ \ g^i, g^j \in \mathcal{C} \tag{5.2}$$

where $|\mathrm{Vor}(g^i)|$ is the area of the Voronoi cell of point $g^i \in \mathcal{C}(s)$.

Figure 5.4 illustrates a DSC that begins on a sensor location $s = g^0 = \{0.5176, 0.6296\}$ that we chose randomly from $[0,1)^2$. We demonstrate $\mathcal{C}_k(s)$ for $k = 9$ (A) and $k = 11$ (B). We calculate the smoothness factor for each k by dividing the greatest Voronoi area by the smallest.

We may say that a continuous DSC $\mathcal{C}_k(s)$ that tend to a constant value of the smoothness factor, divides the unit surface *uniformly*. A constant smoothness factor ρ describes the bound $\frac{1}{\rho \cdot k}$ for the smallest region visited by the DSC. In other words, the smallest region visited by $\mathcal{C}_k(s)$ is at least of size $\frac{1}{\rho \cdot k}$. Therefore, a continuous DSC $C_k(s)$ that tend to some constant smoothness factor ρ, implies on the ability of sensor s to *rapidly* interact with new sensors from S. By rapidly, we mean that every hop in $C_k(s)$ would visit a new sensor with high probability if $|S| \geq \rho \cdot k$.

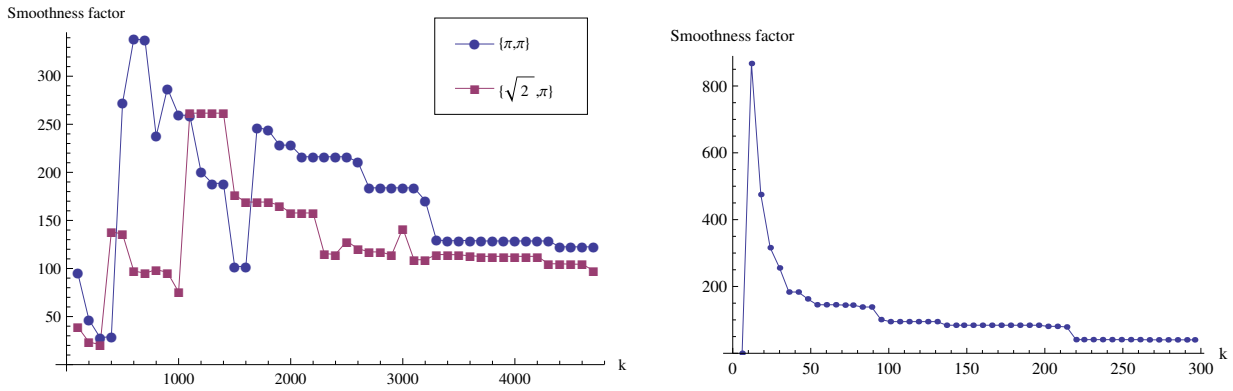


Fig. 5.5: Smoothness factors for: (A) irrational numbers $\{\pi, \pi\}$ and $\{\sqrt{2}, \pi\}$ (modulo 1) and (B) $\{0.01, 0.01\}$.

Figure 5.5 demonstrates the smoothness factor results for three points that we chose on the unit square. On the right (B), we demonstrate the smoothness factors for a chain that begins with point $g^0 = \{0.01, 0.01\}$. However, this chain begins to repeat itself for $k \geq 296$ (due to the coordinates of g^0). On the left (A), we present two examples of sensors that are located at coordinates of irrational numbers ($\{\pi, \pi\}$ and $\{\sqrt{2}, \pi\}$ - modulo 1). These chains do not repeat themselves due to the spacial characteristics of the irrational numbers (infinite numbers after the decimal point). However, their smoothness factors only converge for larger k values than the analysis in (B).

The convergence of the smoothness factors, as shown in Figure 5.5, have repeated for any g^0 that we chose. We gave these two examples (rational and irrational) in order to show that the smoothness factors converge to a constant value, regardless the size of the DSC (for example, irrational numbers are more likely to produce infinite chains). While for rational numbers the convergence is fast ($\sim k = 100$), their smoothness factors vary between 0 and 800 (compared with 0 and 300 for irrational).

In this section we analyzed the DSC generally. Our analysis was based on the distribution of vertices $v \in \mathcal{V}$ that DSC produces. On the next section our analysis will be based on a *discretization* of $CW^i(s)$ (Home nodes of $v \in \mathcal{V}$). We will analyze the number of sensor nodes that where visited by DSC. Note that rather than the general results we have presented, a specific analysis would have be based on specific simulations.

5.5 DSC as a Deterministic "Random" Walk

While DSCs allow sensors accesses to a greater storage capacity, they increase the costs of storage and retrieval. In other words, as the number of DSC links increases, so does the number of packets that travel along them. Also, sometimes a DSC cannot locate a sensor with an available storage capacity on

its first hop. Therefore, as the majority of sensor nodes get their storage capacity depleted, it becomes more and more difficult to locate sensors with available storage capacity. Intuitively, we would like to allow a fast allocation of sensors with available storage capacity, in order to reduce storage costs. In the following simulation, we aim to evaluate the *expansion rate* of DSC and compare it with some intuitive bounds. We define the expansion rate of a WSN as the rate that a DSC visit sensor nodes in S. In other words, we measure the portion of sensors from the WSN that are covered ($\%Cov$) by the DSCs. As far as we know there are no similar researches with whom we can compare our results.

As we mentioned, not all sensors on the WSN are familiar with each other. Therefore, a sensor that had its storage capacity depleted should begin exploring the WSN and "search" for other sensors that can share their storage capacities with it. When the DSC "selects" a sensor that already had its capacity depleted, we calculate the next link of that chain. In Chapter 4 we discussed the Gabber-Galil expander graph and the possibilities of implementing it over a geographical target area. In this section, we describe three different discovery algorithms that we developed, based on that expander graph. We will demonstrate their performances by simulations. Our goal is to analyze the expansion rate of these algorithms and integrate the algorithm with the best expansion rate within our storage management protocol. One of these algorithms is the DSC, given by Definition 7.

As we mentioned in Section 4.3.1, the original Gabber-Galil expander was designed for a discrete grid network. This graph defines 5 permutations between vertices that represent links (Definition 2) between sensor nodes. Our idea, for a geographical implementation of the expander graph is based on these links. Basically we used combinations of these permutations alternately and evaluated the performance for each combination. One combination that we have already described is DSC, which uses permutations σ_1 and σ_3 alternately, or:

$$\alpha_0 = (x, y) \rightarrow \alpha_1 = CW^1(\alpha_0) = (x, x + y) \rightarrow CW^2(\alpha_1) = (2x + y, x + y) \cdots$$

where $CW^i(x, y)$ is given by Definition 7. The two other combinations that we will describe contain the immediate neighbors of a sensor node. Notice that permutations $\sigma_2 = (x, x + y + 1)$ and $\sigma_4 = (x + y + 1, y)$ (for a discrete $M \times M$ grid, Section 4.3.1) refer to the closest vertices in the east and north directions (from Definition 2). We implemented these close neighbors simply by choosing the closest sensors on the north and east directions. In other words, these are sensors that shares a Voronoi edge with the Home Node. We define the sequence 04, as the following four links:

$$\alpha_0 = (x, y) \rightarrow \alpha_1 = CW^1(\alpha_0) \rightarrow \alpha_2 = North(\alpha_1) \rightarrow \alpha_3 = CW^2(\alpha_2) \rightarrow \alpha_4 = East(\alpha_3) \cdots$$

Notice that just like on DSC, we use these sequence recursively. Therefore, it repeats in a loop:

$$\alpha_4 \to \alpha_5 = CW^1(\alpha_4) \to \alpha_6 = North(\alpha_5) \to \alpha_7 = CW^2(\alpha_6) \to \alpha_8 = East(\alpha_7) \cdots$$

The difference between sequence 04 and 06 is that we also use close neighbors:

$$\alpha_0 = (x, y) \to \alpha_1 = CW^1(\alpha_0) \to \alpha_2 = North(\alpha_1) \to \alpha_3 = East(\alpha_2) \to$$

$$\to \alpha_4 = CW^2(\alpha_3) \to \alpha_5 = South(\alpha_4) \to \alpha_6 = West(\alpha_5) \to \cdots$$

In order to evaluate these sequences, we have developed a simulation on which sets of sensor nodes explore the WSN by either of these sequences. Sensors create links according to the definitions above and increment the number of sensors that are connected. We begin our simulation with a primary component $S_0 \subset S$ of size $|S_0|$ sensor nodes selected uniformly from S. On the beginning of the simulation ($k = 0$), each sensor $s \in S_0$ initiates its own chain. Then, for $1 \leq k \leq 200$, each sensor $s \in S_0$ expands its chain by adding one link, according to the sequence that we define: either 04, 06 or DSC. Therefore, on each index k we expand S_k by adding exactly $|S_0|$ links (the size of the primary group S_0). However, it is not guaranteed that every link would simply *discover* new sensors. As we can see in Figure 5.6, the greater k the more difficult it is to discover new sensors. In Figure 5.6 and 5.7 we show simulation results of different sets of S_0 that expand for $1 \leq k < 200$. The measure Cov reffers to the coverage of sensors (in percents from the WSN) that the group of sensors S_0 archives at every step k. Actually, that is the expansion rate, or the number of sensors that the "search-functions" discovered in every step k.

Figure 5.6 demonstrate the comparison between the three sequences that we described above (DSC, 04, 06). Each curve in Figure 5.6 represents a different simulation on which we tested different sequences. In this figure, we show the expansion rate results for a WSN of size 6400 sensor nodes, where the size of the primary group S_0 is 320 sensor nodes, or 5% of the WSN. We selected a primary group of $0.05 \cdot n$ that we choose randomly from the WSN. The idea is to simulate 5% sensors that discover the rest of the WSN. On Appendix A, Figures A.2 - A.5 we give more simulation results for various sizes of $|S_0|$.

The OPT curve, in Figure 5.6, refers to a theoretical upper bound for which sensors on S_0 discover exactly $|S_0|$ a new sensors at each step k. Therefore, in Figure 5.6, the group S_0 discover 320 (Which are 5%) new sensor nodes at every expansion $0 \leq k \leq 200$. However, after 20 expansions ($5 \times 20 = 100$) we already discover 100% of the sensor nodes (that's way the curves maintains the value 100 after $k = 20$ expansions).

As we mentioned, our DSC is deterministic by definition. Moreover, since our DSC is based on an expander graph, and since expander graphs are said to offer a good deterministic emulation of random behavior (Hoory et al., 2006), it is more than expected that we compare our sequences with the expansion rate of a random model such as random walk. A random walk is defined as a stochastic process that starts at one node of a graph, and at each step moves from the current node to an adjacent node that is chosen randomly and uniformly from the neighbors of the current node (Alon et al., 2008). We define a random walk over the complete graph as a walk (or sequence of vertices 5.3) where we choose each vertex randomly from the complete graph. we define two types of walks: rID, whose complete graph is the list of all sensor nodes (S) and rXY, whose complete graph is all the vertices in $[0,1)^2$.

Both these walks randomly select sensor nodes (or Home-Nodes) at each step k. rID is a walk that uniformly selects sensors from S. On each step k we choose $|S_0|$ sensors from S with a uniform probability. In other words, each sensor in S_0 uniformity selects one sensor from S. Alternatively, we can view the rID as a coupon collector problem, for each k we select $|S_0|$ random IDs, until we collect all IDs in S. That selection is optimal in term of graph coverage since all sensors have the same probability to be chosen. Our second walk, rXY uniformly selects points (or locations) from the unit square. Then we calculate their *Home-Nodes* (Corollary 1) in order to obtain the selected sensors. Notice that in this case, sensors do not have the same probability to be visited. Since the selection is uniform over the unit square, the probability to select some sensor s is proportional to its region $\mathrm{Vor}(s)$. The greater the Voronoi region the higher its probability to be selected. The problem of different Voronoi cell sizes was also discussed in (Dimakis et al., 2006; Naor & Wieder, 2007).

Both rXY and rID are known to be an efficient "search-functions" on P2P networks (Gkantsidis, Mihail, & Saberi, 2006). However, since sensors are not aware of the existence of all others, the implementation of rID in real WSN is impossible. In other words, rID require that all sensors should be familiar with each other. Thus, the best expansion rate that we may offer is that of rXY. Recall that since both rXY and rID walks are random, retrievals or reconstructions of these chains cannot be performed without maintaining path structure. However, as we demonstrate, they serve as good comparison to DSC.

In Figure 5.6 we compare between all the walks described above for a WSN of size 6400 sensor nodes, where every walk represents the mean of 200 iterations. Also, we confirmed these results by simulating different topologies (or uniform deployments of sensor nodes $X, Y = \mathcal{U}(0,1)$). The first and maybe the most noticeable observation in Figure 5.6 is that the greater the k the more difficult it is to find new sensors. All curves begin with a fast expansion rate and as k grows the expansion rate decreases. We can think about the decreasing rate as the distance from the OPT curve. An interesting observation is that both 04 and 05 sequences are far from a 100% coverage. They achieve

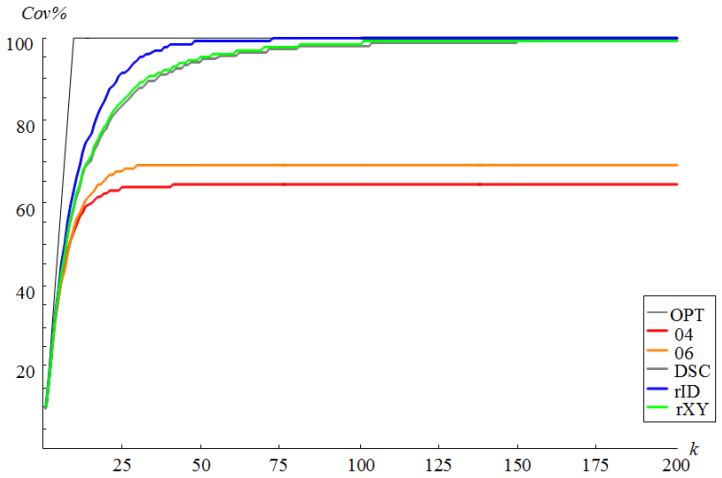


Fig. 5.6: Expanding sets simulation for different sequences. Simulations are conducted on a WSN of size 6400 sensor nodes, where the primary component S_0 is of size 320 (or 5%) sensor nodes.

lower expansion rates than the rate of DSC although their sequences make use of more links. That shows that the close neighbors that we added breaks the expander sequence. We can also observe that a random selection of sensor nodes rID achieves better results than selecting random geographical locations rXY. However the most important and remarkable observation is that the DSC and rXY curves demonstrate the same expansion rate and almost converge. Therefore, this figure confirms the claim that expander graphs offer a good deterministic emulation of random behavior (Hoory et al., 2006).

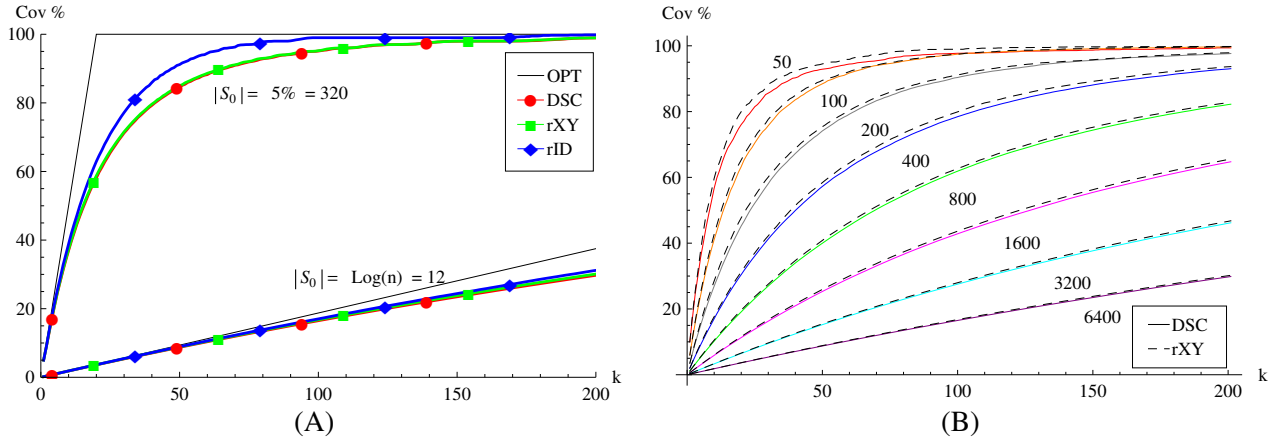


Fig. 5.7: Expanding sets results: (A) 6400 sensor network (two experiments) for $|S_0| = 320$ and 12 sensors (B) Varied network size.

Figure 5.7 presents the growth rate of $|S_j|$ for different primary group S_0 sizes and for different WSN sizes. We simulated various WSN that are uniformly deployed over the unit square for various

network sizes n. On the left side (A), we present two experiments WSN of size $n = 6400$ sensor nodes. In this figure, there are two groups of curves, the upper group referees to an experiment where the size of S_0 is 5% of the WSN size ($|S_0| = 5\%n = 320$). The lower group of curves refer to an experiment where the size of S_0 is $\sim Log_2(n)$ ($|S_0| = Log_2(n) \backsimeq 12$).

On Figure 5.7 we compare the random models only with our DSC. Notice that the OPT curve for $S_0 = 12$ sensors, in this figure, does not reach a 100% coverage. That can be explained as the maximal number of sensors that can be selected for $k = 200$ steps is $12 \cdot 200 = 2400 = 37.5\%$. That explains why all the curves for the $Log_2(n)$ experiments are below. An important observation is that as the WSN coverage is low, or when there are large amount of sensors that where not discovered, the efficiency of DSC is very similar to that of the random models rXY and rID. That observation can be seen on the beginning of the 5% curve, and even better, along the $Log_2(n)$ curve. From the 5% curve we can learn that up to $\sim 80\%$ of the WSN coverage, the expansion rate has an almost linear growth. The remaining 20% are characterized by a much slower rate. On Appendix A, Figures A.2-A.5 we can see that this characteristic repeats for different WSN sizes.

On Figure 5.7 (B), we fixed $|S_0| = Log_2(n)$ and plotted all the results from Appendix A on a single graph. In this graph we compare only between rXY and DSC for a variable network size n (50-6400 sensors). We can see that these two curves converge for every n value. Therefore, we may say that DSC scale with the network size.

Figure 5.7 clearly shows that the results of the DSC deterministic routing algorithm is close to those of rXY. Moreover, our experiment shows that the same DCS behavior repeats for varied $h = |S_0|$ and varied network sizes. Note that the great advantage of a deterministic walk is that it can be predicted. Therefore, sensors do not have to maintain or retrieve chain structures in order to retrieve all their events along the DSC. We title our DSC as a Deterministic "Random" Walk since it visits sensor nodes like a random walk over the geographic target area, while having a deterministic and explicit construction. Therefore, we claim that DSC may be considered to have an efficient **deterministic** expansion rate. The property of imitating a random behavior by a deterministic process (walk) is one of the main impacts of our work.

Chapter 6

Traffic Model

"Whenever a system becomes completely defined, some damn fool discovers something which either abolishes the system or expands it beyond recognition"

Edward A. Murphy (1918 - 1990)

Many research papers in the field of WSN assume that events are generated uniformly on the target area (Heinzelman et al., 2000; Ratnasamy et al., 2003; Silberstein & Yang, 2007). In other words, they assume that the probability for some sensor to sense events is equal in all regions at any given time. Therefore, energy, storage and communication resources (Heinzelman et al., 2000) are consumed equally by all the sensors while the WSN is deployed. However, in real WSN environment this may not be practical (Hung-Yu et al., 2005). One example, that we have already demonstrated (Section 3.5), is a WSN application for wildlife detection. In that example, sensors are deployed in field and monitor the habits of animals. As we showed, some regions (e.g., rivers) attract more animal than others. Therefore, sensors were deployed in these regions are subjected to more events than others. We call these regions, hot-spot regions and we call the sensors that are located in these regions, hot-spot sensors. Therefore, hot-spot sensors are expected to work more intensively than others. Their consumptions of energy, storage and communication are of higher levels.

In this section, we develop a traffic model for WSN that considers the heavy load from hot-spot regions. We analyze and study an event generation model that considers the traffic load from hot-spot regions. Later in Chapter 8, we will use this model for simulations and analysis.

6.1 Lifetime of a WSNs

Network lifetime is a critical key characteristic in the evaluation of every WSN (Dietrich & Dressler, 2009). Sensor nodes are placed out in the field and might be left unattended for months or years. On the design of WSN, it is very important to define the period of time that sensors may be left unattended. Then, the resource capacities of every sensor node should be designed to meet the needs of the WSN. Generally, the lifetime of a WSN is the period of time on which it can operate, while achieving its goals. Maybe the most important factor for the evaluation of WSN lifetime is the energy supply. Each node must be designed to manage its local supply of energy as efficient as possible in order to maximize the total network lifetime (Dietrich & Dressler, 2009). However, sometimes it is more efficient to adopt a global management, where sensors share their resources in order to achieve a network optimization. Accordingly, a great number of algorithms and methods were proposed to increase the lifetime of WSNs in terms of energy (e.g., (Hung-Yu et al., 2005; Wu et al., 2008; Silberstein & Yang, 2007; Heinzelman et al., 2000), etc.).

There are many ways to define the network lifetime, or the time that a WSN can operate. Basically, the definition is dependent on the WSN application and the way it is used. In this section, we will show the most common definitions for the lifetime of a WSN (Dietrich & Dressler, 2009):

Number of alive sensor nodes. This definition is the most common on the literature. According to this definition, the network lifetime last until the first sensor node fails. This definition is very critical for real time application. For example on cross border detection, if a cross border occurred on a region of the unique sensor who failed, the entire WSN fails to its mission. However, most cases allow *several sensors* to be malfunctioned due to multiple region coverage. The number of malfunctioned sensor should be defined carefully as we will show on the next definitions.

Sensor coverage. Suppose that only a finite set of target points on the target area are essential for monitoring goals, the corresponding coverage problem is called target coverage. There are two approaches for the target coverage problem. The first requires that only a percentage α of the region should be covered by at least one sensor (α-coverage). The second approach requires that regions of interest should be fully covered by at least k sensors (k-coverage).

Connectivity. This definition regards the ability of sensors to communicate between themselves within the WSN. That could be measured by the size of the connected component of sensors (percentage of the entire WSN). For example, on many-to-one WSN, we may define connectivity as the total number of packets that could be transmitted to the sink station. In that case, a sensor that cannot transmit to the sink station does not fulfilled the WSN goals.

In this work we investigate the ability of a WSN to store events. We assume many-to-many WSN configuration (see Section 3.3) where communication costs between sensor nodes are lower than the communication with the sink station (Hung-Yu et al., 2005; Heinzelman et al., 2000). Therefore,

sensors that are located on hot-spot regions and have their storage capacity depleted, can communicate with their neighbors (or initiate a multi-hop session to a distant sensor) in order to share their storage capacity. Therefore, we define the lifetime of our WSN to last as long as sensor nodes can store events within the WSN.

Notice that we have not yet discussed the retrieval of events. Since we defined the WSN lifetime to last as long as sensor nodes can store events, an important information that is still missing regards the method that those events can be restored. We consider following three cases. First, an end user can send a data request from a specific sensor's region. On that case, the responsible sensor initiates a `GET` message that travels along the DSC, collects all the events that were stored by the sensor and transmit them to the end user. Second, events can be retrieved by data mules (Luo et al., 2007). Basically, data mules are a sort of a mobile sink station that reaches the target field and communicate very close to the sensor nodes. They wirelessly collect data from encountered nodes and dump these data later to the base station. The third method is physically, collecting the sensors.

6.2 Model Description and Parameters

Let S be a set of sensors where n is the size of the WSN ($|S| = n$ is the number of sensors). We select a set of sensors $HS \subset S$ where $|HS| < n$ to be hot-spot sensors (see Section 8.8 for more information about methods to select the locations of HS sensors). By definition, these sensors generate more events than they can store. In other words, let M be the storage capacity of a sensor node (number of events that it can store), a hot-spot sensor $s \in HS$ is a sensor that sense, with a high probability, more than M.

Since every sensor has a storage capacity M, the overall WSN capacity is defined as $n \cdot M$ i.e., the maximum number of events that all the sensors can store together in total. We define a discrete time T_i that represents the generation of events. Therefore, given n sensors, each with M storage capacity units, T_i is defined as: $0 \leq T_i \leq n \cdot M$ (generation of all possible events that the WSN can store). At every time T_i, one event exactly is sensed by a unique sensor node and stored immediately. We set the probability for an event to be sensed by a HS sensor as p and to be sensed by a regular sensor (non HS) as $(1 - p)$.

Figure 6.1 presents the discrete time-line $0 \leq T_i \leq n \cdot M$. At every discrete time T_i, an event can occur on the region of a HS or regular sensor. Notice that within these two groups we select the sensor that actually sense and store the event, with a uniform probability. We use the parameter γ ($0 \leq \gamma \leq 1$) to represent the percentage of total storage that was consumed by the entire WSN on the discrete time $T_{\gamma nM}$. For example, $\gamma = .8$ refers to the discrete time T_i where $0.8 \cdot nM$ events occurred, or 80% of the WSN storage capacity should be occupied. We observe the WSN on different

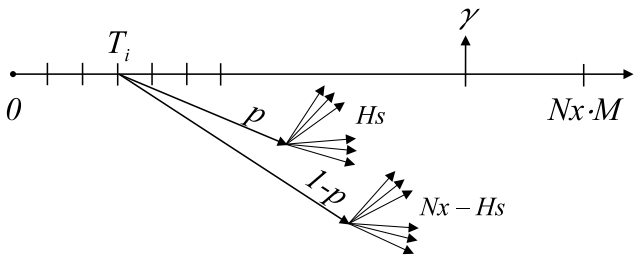


Fig. 6.1: Generation of events.

time clocks $T_{\gamma nM}$ in order to study its statistics. In Table 6.1 we present all the parameters that are used to describe our traffic model in this section. We also show the values that we will use in Chapter 8 for simulations.

Parameter	Meaning	Used Values
n	Network Size	$50 \leq n \leq 6400$
M	Memory Units	100
HS	HS sensors	$1, Log(n), 5\%, 10\%$
p	HS prob.	$0.6 \leq p \leq 0.85$
γ	Overall consum.	$0 \leq i \leq 1$

Tab. 6.1: Parameters for the traffic model.

We uniformity select $|HS|$ sensors from $|S|$ to be hot-spot sensors (non-uniform selection will be discussed in Section 8.8). However, we must guarantee that these sensors would generate more events than non-hot-spot sensors at each observation time $T_{\gamma nM}$. Equation 6.1 describes the condition on p that guarantees for HS sensors to generate more events than regular sensors. We compare between the average number of events that each group of sensors detect until some discrete time $T_{\gamma nM}$, when $\gamma \cdot nM$ events have already occurred. This equation shows that the relation between the size of the hot-spot group and the total size of the WSN should be lower than the parameter p:

$$\frac{\gamma \cdot n \cdot M \cdot p}{|HS|} > \frac{\gamma \cdot n \cdot M \cdot (1-p)}{n - |HS|}$$

$$p(n - |HS|) > |HS|(1-p) \tag{6.1}$$

$$p > \frac{|HS|}{n}$$

We chose to represent the occurrence of events on a discrete time from considerations of simplicity. However, notice that by applying any kind of distribution to the interval between two events, we change the characteristic of the traffic. For example, if we choose a negative exponential distribution between every two discrete time clocks on, we get a Poisson process for the arrival time of events (Papoulis & Pillai, 2002). However, in this work we aim to investigate the WSN lifetime in terms of storage, or the ability of the WSN to sense and store events. Therefore, we consider our time-line as the current amount of storage used by the WSN.

6.3 A Stochastic Problem

We intend to relate the WSN lifetime with the storage consumption. To do that, we need to calculate the discrete clock T_i where the first sensor gets its storage capacity depleted. Actually, in order to find T_i we need to calculate the expected time, on which the first sensor s generated exactly M events. In this section we will give a general solution for the case where every sensor has a different probability to sense an event. In order to make the problem more intuitive, we make an analogy to a game of cards with n players.

We consider a simple game of cards with n players. On each round i each player has its own probability to win. The probabilities for players to win a round are $p_1, p_2, p_3, \ldots, p_n$. Moreover, it is guaranteed that exactly one victory occurs on each round since $p_1 + p_2 + p_3 + \cdots + p_n = 1$. The first player who manages to accumulate M victories wins the game.

Observation: The player who wins the game should accumulate $M - 1$ victories until round $i - 1$. The order for these $M - 1$ victories is not important. Also, this player must win the last round. Let $p_1, p_2, p_3, \ldots, p_n$ represent the victory probability for every player respectively and let $k_1, k_2, k_3, \ldots, k_n$ represent the number of victories each player has accumulated. The probability for a player to win $(M - 1)$ rounds form $(i - 1)$ rounds in total is a Multinomial probability (Papoulis & Pillai, 2002). For this multinomial probability, we consider the following:

1. $p_1 + p_2 + p_3 + \ldots p_n = 1$ - A complete probability (valid for all multinomials).
2. $k_1 = (M - 1)$ - We fix the first player (for example) to be the winner. Therefore, this player should accumulate exactly $(M - 1)$ victories until round $(i - 1)$. Its coefficient k_1 should be set to $k_1 = (M - 1)$.
3. $k_1 + k_2 + k_3 + \ldots k_n = (i - 1)$ - The sum of all the coefficients $k_1, k_2, k_3, \ldots, k_n$ should be equal to $(i - 1)$. Remember that we calculate the multinomial distribution until round $(i - 1)$.
4. $k_1, k_2, k_3, \ldots, k_n \leq M - 1$ - This constrain is important to guarantee that no player wins until round $(i - 1)$. Note that we assume that the first player (with probability p_1) wins at the last round (round i).

We want to calculate the probability for player 1 to win the game on round i. The binomial that we described above gives the probability for this player to accumulate $M-1$ victories while no player wins. Therefore, we should multiply that probability with p_1 on the last round in order to obtain the probability for that player to win. In other words, we calculate the multinomial probability of that player to win in $M-1$ rounds until round $i-1$ and multiply the result by the probability for him to also win the last round (round i):

$p(Player_1$ *wins on round* $i)=$

$$\sum_{\substack{M-1+k_2+k_3+k_4+\cdots+k_n=i-1 \\ k_2,k_3,k_4,\ldots\leq M-1}} \binom{i-1}{(M-1)!k_2!k_3k_4!\cdots k_n!} {p_1}^{M*} \cdot {p_2}^{k_2} \cdot {p_3}^{k_3} \cdot {p_4}^{k_4} \cdots {p_n}^{k_n} \tag{6.2}$$

M^* **Explenation.** Actually, we should put p^{M-1} inside the multinomial, because the multinomial represents the probability of player 1 to accumulate $M-1$ victories. However, as we mentioned we should multiply the result by the probability of player 1 to win the last round. Therefore, we can put the constant p_1 inside the multinomial as in Equation 6.2: ${p_1}^{M*} = {p_1}^{M-1} \cdot p_1 = {p_1}^{M}$

To validate our calculations we give an example of four card players that have the following round-victory probabilities: $p_1 = 0.4$, $p_2 = 0.4$, $p_3 = 0.1$, $p_4 = 0.1$. Also, we fix $M = 40$ as the condition to win the game. First, we calculate the multinomial distribution for player number 1 (Equation 6.2) to win on round T_i, $M \leq i \leq nM - n + 1$. Explanation, before $i = M$ no player can accumulate M victories ($i < M$). Also, $nM - n$ is the last round on which it is possible for all players to accumulate $M-1$ victories (no one wins). Therefore, on round $nM - n + 1$ one player must win. In Figure 6.2 we compare simulation results with Equation 6.2.

If we think in terms of sensor nodes again, our question is: *what is the probability for sensor* s_1 *to accumulate* M *events on the discrete clock* i. Therefore, Equation 6.2 solves the probabilities for s_1 to get its storage capacity depleted on every round i. However, if we like to calculate the probability of sensor s_2, we can simply switch between the probabilities p_1 and p_2 of sensors s_1 and s_2 respectively.

6.4 Expected Number of Event Sensed by Regular Sensors

As we will show later, there is a significant importance to the number of events that are sensed by non-hot-spot sensors. Generally, we defined hot-spot sensors as sensors that are located on hot-spot regions, which generate more events than a single sensor can store. Accordingly, sensors that are located on non-hot-spot (or regular) regions are subjected to fewer events than their capacity. In other words, their storage is not utilized entirely. We would like to maximize the utilization of their storage

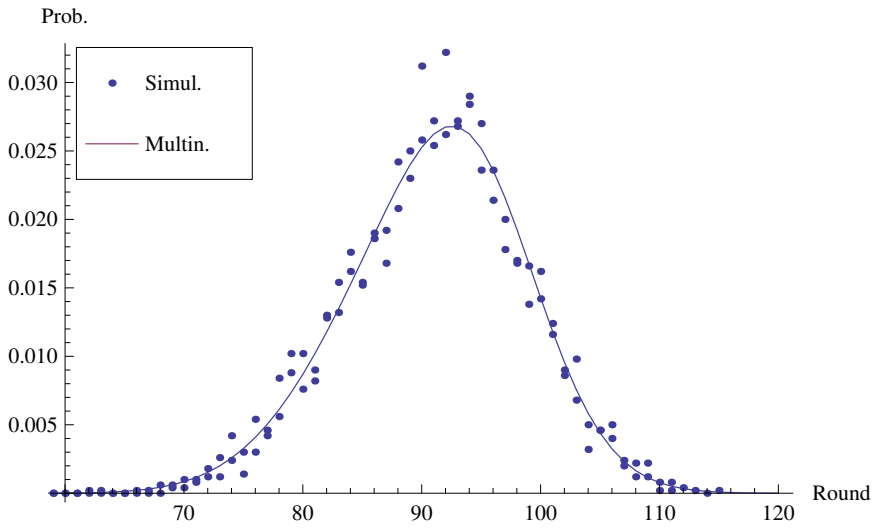


Fig. 6.2: Simulation results versus lifetime PDF (Equation 6.2) for a $\{0.4, 0.4, 0.1, 0.1\}$ configuration. The points represent probability from simulations and the curve is calculated analytically (Equation 6.2).

by allowing them to share their storage capacity. However, it is important to define the amount of storage that every sensor may share. For example, if a sensor shares its entire storage capacity, that sensor takes the chance of getting its entire storage capacity depleted before it generate a single event. Moreover, since sensors cannot know if they are located on hot-spot regions before they have their storage capacity depleted, it is extremely important to define how many storage units a sensor may share, before its is deployed on the target area.

In this section we will give a simple analysis that can help to determine the amount of storage that a sensor should share. Let T_i be a discrete time on which we observe the WSN, where $i = \gamma \cdot n \cdot M$ events were already generated. We may deduce that on the average case:

- $\gamma nM \cdot p$ - events were generated by hot-spot sensors.
- $\gamma nM \cdot (1 - p)$ - events that generated by non-hot-spot sensors.

In order to obtain the mean number of events sensed by regular and hot-spot sensors at the discrete time T_i, we can simply divide the values that we calculated above by the number of regular and hot-spot sensors respectively. However, as we will show in this section, it is important to analyze the distribution of events that are sensed by non-hot-spot sensors. That way, we can study the behavior of these sets of sensors and set our decision, based on the entire set.

We need to construct the probability function for the number of events sensed by a regular sensor node. Let p be the probability that a hot-spot sensor senses an event, let n denote the total number

of sensors and HS be the number of hot-spot sensors. Also, remember that once a regular event was sensed, it is distributed uniformly between all regular sensors. Therefore, we can calculate the probability p_R for a regular sensor to sense an event at any time T_i as:

$$p_R = \frac{1-p}{n-|HS|} \tag{6.3}$$

We may think about the process of generating events as a binomial distribution. A sensor may generate an event with probability p_B, or not generate an event with probability $(1-p_B)$. Therefore, the probability for some regular sensor s to sense exactly k events for some discrete time T_i can be calculated according to the binomial distribution $B(k;i,p_R)$. Let $i=\gamma\cdot nM$ be the number of events that occurred until discrete time i and let p_R according to Equation 6.3, the probability for a regular sensor to sense k events exactly is given by Equation 6.4:

$$B(k;\gamma\cdot nM,p_R) = \binom{\gamma\cdot nM}{k}(\frac{1-p}{n-|HS|})^k\cdot(1-\frac{1-p}{n-|HS|})^{\gamma\cdot nM-k} \tag{6.4}$$

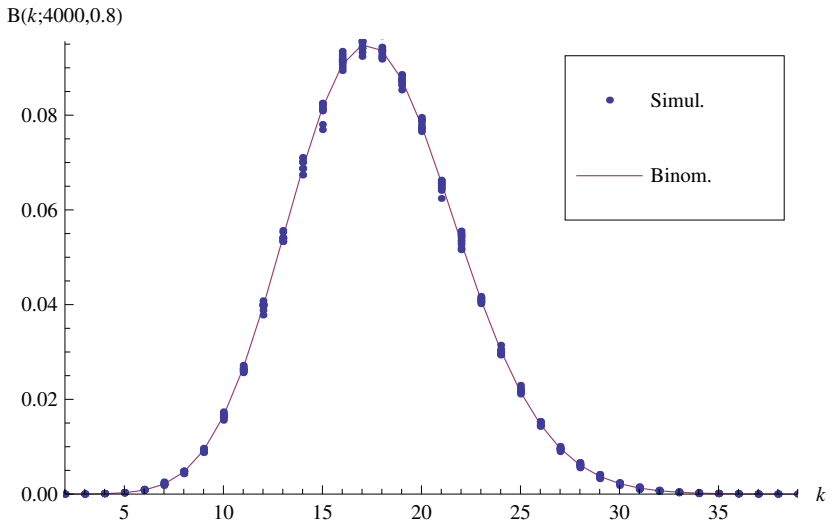


Fig. 6.3: Comparing event generation of regular sensors with the binomial distribution.

In order to demonstrate this equation we conducted a simulation on which $n=50$, $|HS|=5$, $p=0.8$, $M=100$ and $\gamma=0.8$. We generated $\gamma nM=4000$ events and distributed them between the sensors according to our traffic model. Then, based on the results, we calculated a histogram for the probability of regular sensors to generate k events. We repeated this simulation 1000 times. In Figure 6.3, we compare between the simulation results and Equation 6.4. It is very clear that the binomial

distribution represent our results. Moreover, this figure depicts that the mean number of events a regular sensor generates matches the mean that we demonstrated above $\frac{\gamma nM \cdot (1-p)}{n-|HS|} = 17.8$. However, we can also observe that many regular sensors generate more events than their mean values. Some regular sensors even generate more than 33 events. According to Figure 6.3, we should consider the binomial distribution (its mean, variance and standard division) in order to define how much storage capacity a sensor may share.

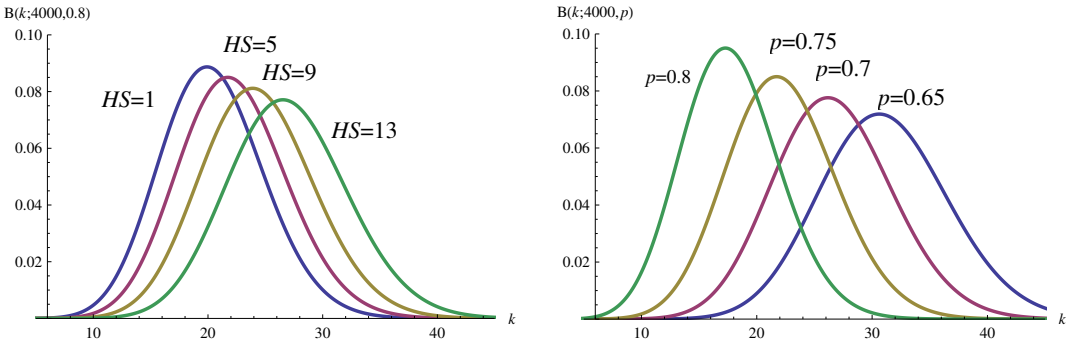


Fig. 6.4: Binomial distribution of events by regular sensors (Equation 6.4). Varying the number of HS sensors and the probability p.

In Figure 6.4 we vary the parameters HS and p from Equation 6.4 in order to study how they affect the distribution of events per regular sensor. As we can see, incrementing the HS probability p results in a shift of the mean to the left and reduction in the variance (the gaussian shape becomes more narrow). Reduction in the number of HS sensors, results also in shift of the mean to the left and reduction in the variance. In Equation 6.5 we give the mean and variance values of events generated by non-hot-spot sensors according to the binomial distribution:

$$E[B(k; \gamma \cdot nM, p_R)] = \gamma \cdot nM \cdot \frac{1-p}{n-|HS|}$$

$$VAR[B(k; \gamma \cdot nM, p_R)] = \gamma \cdot nM \cdot \frac{1-p}{n-|HS|}(1 - \frac{1-p}{n-|HS|}) \tag{6.5}$$

According to these equations we can explain the behavior of the graphs in Figure 6.4. As we increment the probability p, the expected value of events decreases (shifted left). Incrementing the number of hot-spot sensors reduce the denominator (expected value) and causes a right-shift. Also, from the Variance equation we can study how the Gaussian bell becomes narrow as we increment the value of p.

6.5 Number of Hot-Spot Events and Expected Lifetime

Equation 6.4 describes the binomial probability for a regular sensor to sense exactly k events from $\gamma \cdot nM$ (the total amount of events on time of observation γ). In the same way we can develop the binomial probability for a hot-spot sensor to sense k events on a γ observation time as:

$$B(k; \gamma \cdot nM, p_{HS}) = \binom{\gamma \cdot nM}{k} (\frac{p}{|HS|})^k \cdot (1 - \frac{p}{|HS|})^{\gamma \cdot nM - k} \tag{6.6}$$

In this section we will develop an expression for the expected time, on which a HS sensor senses exactly M data units. In order to calculate the probability of a hot-spot sensor to sense exactly M events, we need to change the parameter k in equation 6.6 to M:

$$B(k; \gamma \cdot nM, p_{HS}) = \binom{\gamma \cdot nM}{M} (\frac{p}{|HS|})^M \cdot (1 - \frac{p}{|HS|})^{\gamma \cdot nM - M} \tag{6.7}$$

Note that Equation 6.7 calculates the probability of a hot-spot sensor to sense M events on a discrete time $\gamma \cdot nM$. Also, notice that our variable in that equation is γ (and not M). Therefore, similar to Section 6.3 we can vary the discrete time and find a γ that maximizes the probability for a hot-spot sensor to generate M events. Actually, what we want to do is to shift the Gaussian bell (Figure 6.4) to a position where its mean value (which is the maximum probability) receives the value of M events.

$$E[B(k; \gamma \cdot nM, p_{HS})] = \gamma \cdot Mn \cdot \frac{p}{|HS|} \tag{6.8}$$

That is, by changing the coefficient of the binomial $\gamma \cdot nM$. Equation 6.8 describes the mean value for the binomial (and therefore for all values of k). Our goal is to compare the expected value with a value of M events:

$$\gamma \cdot Mn \cdot \frac{p}{|HS|} = M$$

$$\gamma \cdot n \cdot \frac{p}{|HS|} = 1 \tag{6.9}$$

$$\gamma = \frac{|HS|}{p \cdot n}$$

Therefore, the discrete time where the first hot-spot sensor depletes its storage (or sense exactly M events) is $\gamma = \frac{|HS|}{p \cdot n}$. This discrete time T_i is for $i = \gamma n M$ or $i = \frac{|HS| \cdot M}{p}$. Two important observations should be put in place here:

1. Since we deal only with two probabilities and since $p_R < p_H S$, it is more likely that a hot-spot sensor would be the first who depletes its storage capacity. Therefore it is enough to calculate only the expected time of an hot-spot to generate M events.

2. After the discrete time $i = \frac{|HS| \cdot M}{p}$ the entire system changes. That is due to the fact that the hot-spot sensor, who had its storage capacity depleted, contributes its probability to store an event to a sensor with available storage capacity. In other words, the probabilities $p_1, p_2, p_3, \ldots, p_n$ (Section 6.3) are changed (two probabilities merge together).

6.6 On the Lifetime of a WSN in the Literature

WSN's lifetime is usually mentioned in the literature in the scene of energy consumption. For example, Wu et al. (Wu et al., 2008) estimate the expected lifetime when coverage holes begins to form, due to energy exhaustion. Similarly, (Hung-Yu et al., 2005) define the WSN lifetime according to coverage holes and prolong the WSN lifetime by deploying more sensors over this regions. However, in our work the WSN lifetime is considered in the scene of storage. In other words, our protocol begin to construct DSC from the discrete time T_i where the first sensor had its storage capacity depleted (Section 6.5). Therefore, our goal is to prolong the WSN lifetime up to the discrete time where it is impossible to store farther events. Notice that although we define the WSN lifetime as due to the storage resource, other resources may affect the WSN lifetime as well.

As we mentioned in Section 6.1 there are many ways to define the WSN lifetime (number of alive nodes, sensor coverage, connectivity, etc.) (Dietrich & Dressler, 2009). Moreover, the WSN lifetime depends on all the available resources and not only on the resource that we try to optimize (storage in our case). Therefore, depending on the WSN configurations, lifetime might be affected by other resources. For example, while we simulate storage consumption in WSN, its lifetime might also be affected by the energy consumption. In other words, the WSN might create coverage holes or have connectivity problems even before the entire storage capacity was depleted.

Our work defines the WSN lifetime as function of storage consumption from 0 to nM events. In other words, we analyze the storage consumption and its implications, based on the limits of the storage resource. Energy constrains can be considered on top of our storage analysis. Thus, our analysis can be used to define the battery capacity that is needed in order to maintain the WSN effective up to a given lifetime (discrete time T_i). Alternatively, given a specific battery configurations,

we can use our analysis in order to calculate the expected WSN lifetime. Liao et al. (Liao & Wu, 2008) analyze their WSN protocol similarly. They define a storage management sachem for WSN and simulate the utilization of storage up to an energy limit in magnitude of $1000 \cdot 10^6 nJ$ (Liao & Wu, 2008). In other words, they fix the battery configuration and analyze their protocol performances up to that point. Our analysis is more general, in the scenes that it can be interpret for any given energy limit.

Chapter 7

DSC Protocol

"If I have seen further it is only by standing on the shoulders of giants"

Isaac Newton (1643 - 1727)

In this section we describe our distributed protocol for storage management in WSN. Our protocol optimizes the utilization of the network's overall storage capacity. Although we achieve an overall optimization, our protocol is designed to run *locally* on each individual sensor node. By locally, we mean that a sensor node should only be aware of its location and the length of its own DSC. Moreover, a sensor that is asked to store an event from another region should be able to calculate the next DSC link, if necessary.

Generally, our protocol can be thought as a simple abstraction that runs at each sensor node: *If possible store locally else store forward*. Our protocol implements the *store forward* abstraction both for events recorded by the sensor node and events that were directed to it. We begin with a high-level description of the two basic sessions, STORE and RETRIEVE, in order to demonstrate the distributed nature of our protocol. Later, in Section 7.4, we discuss low-level mechanisms, which drastically improve the performances of our protocol.

7.1 STORE

Whenever a sensor that already had its storage capacity depleted wishes to store an event, it addresses the last link of its own DSC chain (EOC - End Of Chain). Note that every sensor in its preliminary state begins with a zero length chain who's EOC is the sensor itself. If the sensor that is located at EOC have also depleted its storage capacity, we add a new link to the DSC that replaces the

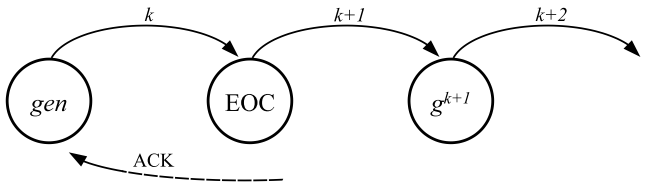


Fig. 7.1: PUT message searching for available storage capacity.

former EOC. The STORE session consists of two types of messages: PUT messages that seek for a sensor with available storage capacity and ACK messages that are sent back in the end of a successful STORE session.

Algorithm 1 ***PUT***$(g^k, k, gen, data)$
If *FreeStorage(data)* ***Then***
  StoreLocally(data)
  send ACK(gen,k)
Else *send* PUT$(CW^{k+1}(g^k), k+1, gen, data)$

The pseudo code in Algorithm 1 describes a PUT message that was received by sensor s. This message contains the geographical address (x, y coordinates) g^k that is located within the region of sensor s ($s = HomeNode(g^k)$). k is the link index of the DSC $\mathcal{C}_k(s)$ (Definition 7). It also contains the geographical address gen of the sensor that is responsible for the overflowed events $data$ (the sensor who initiated that chain is responsible for the gen address). Whenever a sensor that had its storage capacity depleted has to store data, it issues a PUT message to its EOC. The sensor that receives a PUT message can either store it locally or *store forward*:

- If s has already depleted its storage capacity, it will increment the DSC by sending the same PUT message to the next link. Note that although the DSC was initiated by gen, s can easily calculate g^{k+1} *locally*, without consulting gen.
- If s can store the $data$, it sends an acknowledgment to gen that contains the chain length k. Notice that k is updated recursively. Therefore, gen can update its chain size k and EOC address. When a new data is sensed, the gen sensor would address it directly to its new EOC.

Notice that the last line Algorithm 1 is recursive. Therefore, each time that the PUT message is reissued, the parameters g^k and k are recursively incremented. Figure 7.1 demonstrate a PUT message

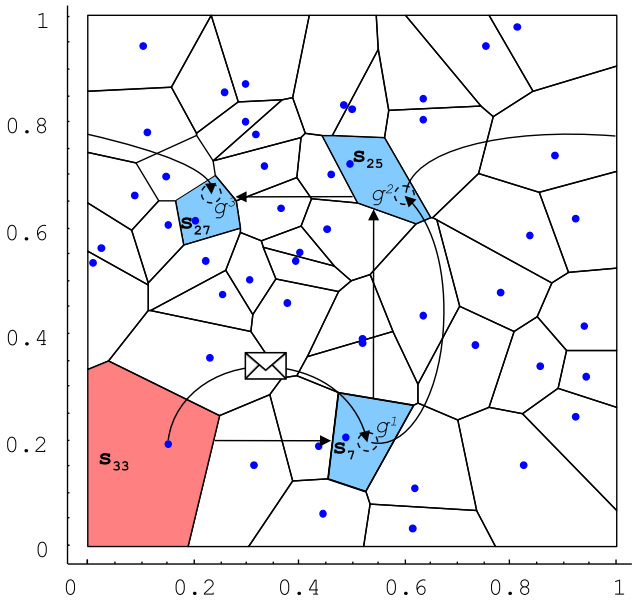


Fig. 7.2: The Voronoi diagram for a set of sensor nodes, DSC starting at s, $\mathcal{C}_3(s) = (s_33 = g^0, g^1, g^2, g^3)$ and the DCS $(s_{33}, s_7, s_{25}, s_{27})$.

that is sent from the EOC to other sensors and increments the DSC length. This figure shows how the parameter g^k is incremented until the data is stored and an ACK message is sent back.

7.2 RETRIEVE

As we mentioned in Chapter 3, sometimes on many-to-many WSN an end user wishes to retrieve some information from within the WSN. When a $user$ wants to RETRIEVE *all* the events from a specific sensor gen, we route GET messages along its DSC. Sensors that are included on that chain and contain data that was originally sensed by gen, retrieve that data back to the $user$.

Algorithm 2 ***GET***$(k, g^k, EOC, user)$
If *DataExist(gen)* ***Then***
  Send data to user
If $(g^k \neq EOC)$ ***Then***
  send ***GET***$(k+1, CW^{k+1}(g^k), EOC, user)$

Algorithm 2 describes a GET message, received by sensor s. This message contains the current k index of the chain, where g^k is a geographical address that is located within the region of sensor

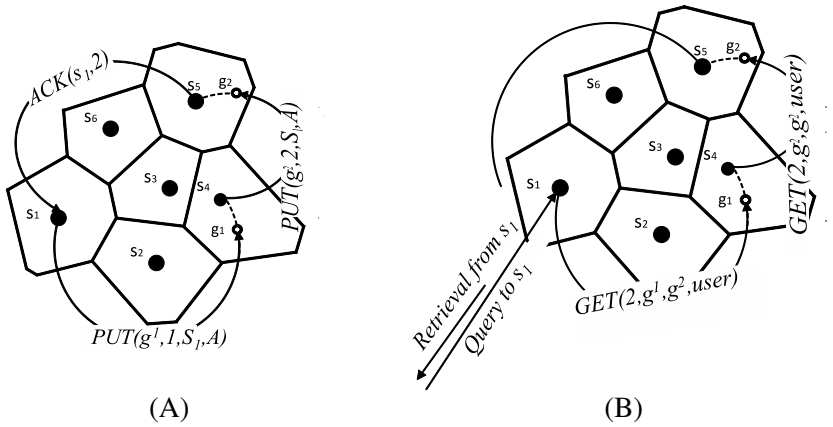
s ($s = HomeNode(g^k)$). It contains the EOC address, where the RETRIEVE session stops. It also contains the $user$'s address for retrieval. A sensor that previously stored data, originally sensed by gen, would retrieve that data to the $user$. In addition, if the current g^k geographical address is different from the EOC, another `GET` message is recursively sent to the next DSC link.

Note that our protocol does not require any initialization or change in hardware. Moreover, since DSC is deterministic, it is used both for tracking available sensors and for retrieving the distributed data, by reconstructing those chains. Fig. 7.2 illustrates a DSC of length 4, starting at s. Note that $\{g^1, g^2, g^3, g^4\}$ and $\{s_1, s_2, s_3, s_4\}$ suites the link $CW^i(s)$ and its discretization $Home(CW^i(s))$ respectively. Let $\mathcal{C}_3(s) = (s, g^1, g^2, g^3)$ where $g^1 = CW^0(s)$, and $g^2 = CW^1(g^1) = CW^1(CW^0(s))$, etc. DSC denotes the discrete set of sensors for this walk is (s, s^1, s^2, s^3) where $Home(g^i) = (s^i)$ (Corollary 1, GPSR protocol Section 5.1).

7.3 Examples for STORE and RETRIEVE Sessions

In Figure 7.3 we demonstrate the STORE and RETRIEVE sessions, using practical examples. In Figure 7.3 (A) we describe a STORE session, where data cannot be stored by the sensor who sensed it. Specifically, we describe an event A that was sensed by sensor s_1 that already knows g^1 (its EOC points on g^1). Therefore, s_1 sends its data to its EOC. Notice that sensor s_4 is Home-Node for the geographic location g_1. In this scenario, since s_4 has already got its storage capacity depleted, it continuous and sends the packet to the next link on the DSC (of s_1). s_4 can obtain the last geographical address and the number of link from the packet it received (g^2 and 2). Therefore, it calculates *localy* the point $g^2 = CW^2(g^1)$ (Definition 7). Since the Home-Node for $s_5 = HomeNode(g^2)$ has an available storage capacity, it stores the data for sensor s_1. Moreover, s_5 sends acknowledge to s_1 indicating that the current length of the chain is now 2 links. Therefore, s_1 would send its upcoming events directly to its $EOC = g^2$.

In Figure 7.3 (B) we describe a retrieval session. On this session, an end user accesses a specific region (the region of sensor s_1) in order to study all the events that took place there. Since not all events sensed by s_1 are stored on its local memory, it initiates a RETRIEVE session which passes through all the sensors with the data that was previously sensed by s_1. s_1 has its current chain of length (2) links and the last geographical address ($EOC = g^2$). These details are added to the get packet. Sensor s_4 who is Home-Node to the first link, understands that there is another link in that chain (the chain ends on g^2). Therefore, it continuous the retrieve session by re-sending the `GET` packet to the next link on the DSC. Finally, the packet reached the EOC (s_5) who responds directly to the end user. Since an end user is not involved in the retrieve session, these `GET` messages are transparent to him.



(A) (B)

Fig. 7.3: (A) STORE Session. (B) RETRIEVE Session.

7.4 Protocol Extensions

We began this chapter with a high level description of our protocol in order to allow the reader an intuitive understanding of our distributed storage model. In this section we discuss some additional low-level mechanisms that were not covered in the previous sections. These mechanisms can be used to reduce DSC costs and enhance our protocol performances:

Local Storage Consumption (LSC): A WSN contains sensors that produce more data than they can store locally. Since these sensors would have their storage capacity depleted at an early stage (see Section 6.1), they would have to construct DSC in order to store their overflowed events. As a result, these sensors will occupy the storage capacities of regular sensors. That may lead to a situation, in which a regular sensor would have its storage capacity depleted even before it has sensed any event. In order to prevent from regular sensors to also construct DSC, we assigned a fraction of the sensor's storage capacity only for a *Local Storage Consumption* (LSC). LSC is reserved only for local events and can't be used by any other sensors. As we will show in the next chapter, the size of that parameter plays a major role in our simulations.

Inactive Links: Each time that a DSC link addresses a sensor that already had its storage capacity depleted, our protocol waste important energy. Although we cannot prevent energy waste during the construction of DSCs (`PUT` messages), we can skip those links when a `GET` query is issued. To do this, a sensor that had its storage capacity depleted may keep a binary vector $b = \{0, 0, 1, 0, 1, ...\}$ of length k where the zeros represent inactive links. We use this extension to differentiate between the STORE and RETRIEVE sessions (Section 8). In other words, while the cost of a STORE session takes under considerations links that connect with unavailable sensors, the RETRIEVE session may consider only the link to sensors that really stored events (successful `PUT` messages).

Early notice: Let s_t be the EOC of some sensor s_i and suppose, that s_t has only one data unit left

in its storage capacity. When s_i sends an event to be stored in s_t, s_t would have its storage capacity depleted immediately after it stores the event from s_i. Therefore, s_t can send its ACK to s_i with a new chain length $k+1$. Thus, s_i will not address s_t anymore. Instead it will send its additional events to the next link of its DSC.

Familiarity with the region: Note that packets are sent to a geographical address that lies on a Home-Node regions. Therefore, a packet would have to encircle the Home-Node region in order to determine which sensor is the closest to the destination address (see Section 5.1). Also, according to our protocol a sensor that had its storage capacity depleted will continuous sending its overflowed events to the same sensor. We suggest a mechanism that reduces the transmit costs for encircling the Home Node region. Note that according to Corollary 2 this operation has a constant cost, since the packet travels six hops in average (there are six neighbors in average to each sensor node). According to our suggestion, the first packet that encircles a Home-Node region could inform it with the locations of its close neighbors (some close neighbors can be out of the Home-Node's communication range). A Home-Node, who received the first packet which was directed to its regions, can study all its close neighbor locations. Therefore, when it receives a new packet, the home node can calculate if the geographical address is within its region or not. That way we can save a lot of energy.

Chapter 8

Simulations

Deep thought: "... a computer of such infinite and subtle complexity that organic life itself shall form part of its operational matrix ..."

Douglas Adams (1952 - 2001)

In this chapter we describe a set of experiments that we have conducted in order to evaluate the efficiency of our protocol (Chapter 7). Earlier, in Section 5.5 we have shown, through specific simulations, the performances of our Deterministic "Random" Walk. We showed that we have developed a walk (Section 5.3) that can be used in order to "sample" the unit square, or to locate sensor nodes, with the same efficiency as a random selection of geographical locations $x, y = \mathcal{U}(0, 1)$ with a uniform distribution. Since the core of our protocol is the deterministic "random" walk, our analysis, presented in this chapter, is oriented towards the performances of that walk. We set out to evaluate two basic metrics, STORE and RETRIEVE. The metric STORE refers to the number of `PUT` messages that are sent by the protocol in order to store overflowed events from sensors that had their storage capacity depleted. Note that without the use of our protocol, the WSN would have a much shorter lifetime (Section 6.1). In other words, overflowed events would have been dropped and their data was lost. RETRIEVE is the number of hops that a `GET` message has to travel in order to collect all the events that were sensed by a specific sensor. For a more profound understanding of the difference between STORE and RETRIEVE costs, see Section 7.3. The simulations described in this section are based on the traffic model that we have developed in Chapter 6.

8.1 Motivation and Related Works

As we mentioned on the introduction, there are numerous works in the area of resource aggregation (Silberstein & Yang, 2007; Dimakis et al., 2006; Hung-Yu et al., 2005; Ratnasamy et al., 2003). However, very little is known about *pure* resource aggregation. By pure, we mean that the suggested scheme should allow a general aggregation model, rather than a solution for a specific case. The problem with specific cases is that they cannot consider all the possible scenarios. Therefore, there would simply be some scenarios, for which the resource aggregation model cannot support. In this work we suggest a novel storage aggregation model that aggregates the entire storage capacity on WSNs in a more *pure* way. In this section we will describe some related works that implement resource aggregations. We will demonstrate their differences on the design level from our work and give some motivation to our work.

Various works concern energy optimization. They consider the power consumption either as a local problem (within individual sensor - e.g. when it is active) or as a global optimization problem within the whole network. An example to the last is the work on energy holes (Wu et al., 2008). This work tries to solve the problem of high energy consumption on sensors that suffers high traffic load. They call those sensors that had their energy capacity depleted "energy holes" because they form coverage holes that cannot be tolerated on WSN applications. Their work is based on *many-to-one* WSN, where sensors communicates with a base station (sink). Therefore, sensors that are closer to that sink station observe a heavier load towards it and, therefore, are at risk of having their energy capacity depleted earlier than more distant sensors. However, thier research does not provide a solution to random energy holes that may be far from the sink station.

Ratnasamy et al. (Ratnasamy et al., 2003) suggest a data centric model that allows data aggregation on WSN. Generally, their work implements a distributed data structure within the WSN, which is used to aggregate both communications and data resources. Data with the same general name (e.g., elephant sighting) is stored on a specific sensor. They use a Geographic Hash Table (GHT) to hash events names into a geographical addresses. Similarly to our Continuous Discrete Approach (Section 4.4), they associate the continuous geographical address space with sensor locations. Therefore, queries for a specific information are sent directly to a hosting sensor without flooding the WSN. On their model, sensors are communicating within themselves in order to deliver events to their hashed locations. Unlike storage aggregation, data aggregation is based on the type of events that would be sensed. However, they do not pay attention to hot-spot environments and their implications on GHT.

Finally, Liao et al., (Liao & Wu, 2008) developed a storage management model for WSN which is oriented towards the GHT protocol that we described above. Since GHT direct events of the same type to a hash location within the region of a sensor node, there is a risk that hot-spot events would cause sensor nodes to get their storage capacity depleted. Their solution is that a sensor, which had

its storage capacity depleted, should set its location to $\{\infty, \infty\}$. That way, events are directed to the available sensors that are close to the hash location. The problem with their model is that although it provides a solution to the storage problem, creates coverage holes close to the hash locations. Note that the sensors that share their storage in that case are inside the hot-spot region or very close to it. An interesting contribution of their work is the multiple threshold mechanism that is different from our LSC (Local Storage Consumption) approach. According to their mechanism, a sensor that had its first storage threshold depleted should direct its events to its neighbors. When the majority of sensors had their storage threshold depleted, they decide together to use the next threshold level.

8.2 Configurations

In order to simulate our protocol, we have developed a WSN simulator under the Wolfram Mathematica 7 environment. For technical information about our simulator see Appendix B. We simulated WSNs of sizes $n = 50, 100, 200, 400, 800, 1600, 3200$ and 6400 sensor nodes. For each network size we simulated different WSN topologies, which are based on the uniform deployment of sensors $(x, y = \mathcal{U}(0, 1))$ over the unit square $[0, 1)^2$. This chapter describes a set of simulations, on which we aim to evaluate the efficiency of our protocol (Section 7) with respect to its "search-function" (Section 5.5) that locates available sensor nodes. In this section, we describe the general configuration that are used for all simulations in this chapter. We will describe the constants, variables and parameters that we used for configurations. Moreover, in each section we will provide a specific configuration table for specific simulations.

Constants. We set a storage capacity of $M = 100$ data units. The idea of setting $M = 100$ is to be able to represent storage usage between 0 and 100 percents. Note that as we increment parameter M, simulation running time is incremented as factor of n. Each of these units can be used to store a single event either by the proper sensor or be shared with the rest of the sensors on the WSN. To prevent the possibility of endless re-transmissions, due to `PUT` messages that cannot find sensors with available storage capacity, we fix a TTL (Time To Live) of 16 hops for `PUT` packets. In other words, if a STORE session has failed to store an event within 16 *consecutive* DSC links, that event is dropped. However, as we will show on our results, less than 0.05% of total messages are dropped. Each simulation that we conducted was repeated 100 times. Therefore, the results presented in this chapter represent the mean value of 100 simulations.

Variables. To evaluate our protocol, we wish to study the implications of different environment on the WSN. To do this, we vary both the environment and the protocol configurations. As we mentioned, we vary the size n of the WSN $50 \leq n \leq 6400$. Let T_i be a discrete time, where on every index i only a unique event may occur on a single sensor region (see Section 6.2). We set i to

vary $0 \leq i \leq nM$ according to the traffic model (Chapter 6). We define our simulation to last for time indexes $0 \leq i \leq \gamma \cdot nM$, where $\gamma < 1$ in order to reduce running time. We will show that for $\gamma = 0.8$ the STORE and RETRIEVE costs are still reasonable. However, the question of maximal γ is behind our scope since it involves much more factors (sensor size and cost of flash memory Vs. cost of power supply) as we show in Section 9.2. We set a *Local Storage Consumption* (LSC) constant (see Section 7.4) for all sensors that is initialized on the deployment of the WSN (it is constant during a simulation). LSC refers to a fraction of the storage that is restricted only for local usage. On our simulations we vary the LSC parameter in order to study its implications on the STORE and RETRIEVE costs.

For environment configurations, we vary both the number of HS sensors and the way they are deployed over the target area. As we showed in Section 6.2 Equation 6.1, the size of the WSN, the number of HS sensors and the probability p are related together. We vary the number of HS sensors $1 \leq |HS| < np$ (see Equation 6.1). On the following experiments we vary the probability for hot-spot events to occure $0.6 \leq p \leq 0.85$ and the number of sensors that are located on hot-spot regions $1 \leq n \leq 20\%$ of n. Also, we will test several configurations of hot-spot regions, where hot-spot sensors are uniformly distributed within the target area or concentrated on specific regions.

An important measure on our simulations is the *average number of links* obtained by the DSC protocol. During a simulation, every sensor who had its storage capacity depleted, constructs a DSC. The total number of links obtained by these DSC grows with T_i, since more and more sensors have their storage capacity depleted. We consider the cost of our protocol as number of links that it creates. However, since the load is divided over the WSN, our interest is in the average number of links, or number of links per sensor node. We note the measure $Links$ (y-axis in the graphs on the following sections) as the cost of our DSC protocol in links per sensor node.

8.3 STORE and RETRIEVE Analysis

In this first set of simulations, we would like to evaluate the two fundamental metrics of our protocol: STORE and RETRIEVE (Sections 7.1 and 7.2). As we mentioned in Chapter 6, we suspect that the Local Storage Consumption (LSC) parameter serves a key role in the evaluation of these two metrics. In other words, in this section we aim to investigate how does the number of storage units, that a sensor node can share, affects the performances of our protocol. To do so, we fixed the probability p, for an event to be sensed by a hot-spot sensor, to be $p = 0.8$. That value of p guarantees that at any discrete time of a simulation, 80% of all events are generated by hot-spot sensors. Therefore, these sensors rapidly get their storage capacity depleted (see Section 6.5), allowing our protocol to distribute their overflowed events.

Table 8.1 describe the configurations for the simulations described in this section. We chose to present the results for a relatively large sensor network with 1600 sensor nodes. Note that there is a relation between the storage capacity of a sensor node ($M = 100$) and its Local Storage Consumption $0 \leq LSC \leq M$. However, as we will demonstrate in this section, LSC values of $0 \leq LSC \leq 40$ are enough to understand the behavior of the WSN.

8.3.1 Simulation Description

In this section we demonstrate a specific simulation where all of the parameters that we describe above are constants. We demonstrate a simulation on which 80 hot-spot sensors are responsible for 80% of the events sensed by the WSN. Therefore, this group of hot-spot sensors gets their storage capacity depleted in the beginning of the simulation. These sensors use our protocol in order to discover available sensors that can store over loaded events. Therefore, throughout the simulation we construct DSC links that (basically) connect between hot-spot sensors and sensors that can share their storage. The x-axis in Figure 8.1 expresses the discrete time T_i in terms of percentage from the total storage capacity of the WSN $0 \leq i \leq 0.8 \cdot nM$. In other words, this figure demonstrates the generation of events up to 80% of the total WSN capacity. The y-axis measures the number of DSC links that were created in average. Notice that we distinguish between the STORE and RETRIEVE sessions (Section 7.4) according to the number of links that are used for each of these sessions.

Figure 8.1 shows the costs of storing and retrieving events over DSCs for each percentage from the total WSN capacity (or, for every discrete time clock i). For example, in order to utilize 70% of the WSN capacity we have to pay ~ 1.5 links per sensor node. Recall that these links are used for storing the overflowed events. Also, we pay ~ 0.7 links per sensor node in order to retrieve these events (on retrieval we access only sensors that actually contain data - Section 7.4). The reason for these costs is a high generation of events by hot-spot regions (5% of the sensors are responsible for 80% of the events).

Parameter	Values
n	1600
M	100
HS	$1, 9, 80, 160$ or $(1, Log(n), 5\%, 10\%)$
p	0.8
γ	$0 \leq \gamma \leq 0.8$
LSC	$0 - 40$
Environment	HS sensors are uniformly deployed

Tab. 8.1: Configuration for the STORE and RETRIEVE analysis.

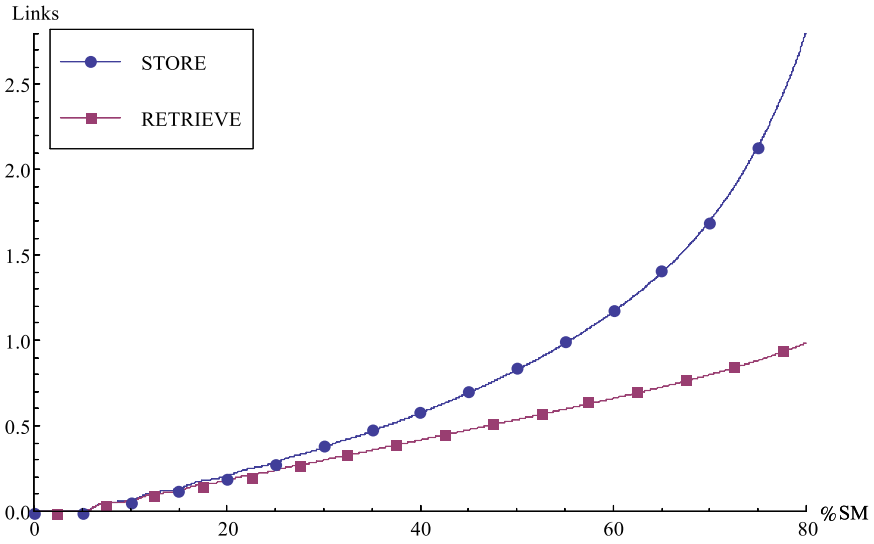


Fig. 8.1: Simulation of STORE and RETRIEVE for $|HS| = 80$ sensor nodes and $LSC = 23$. Average number of links with respect to the percentage from used System Memory (SM).

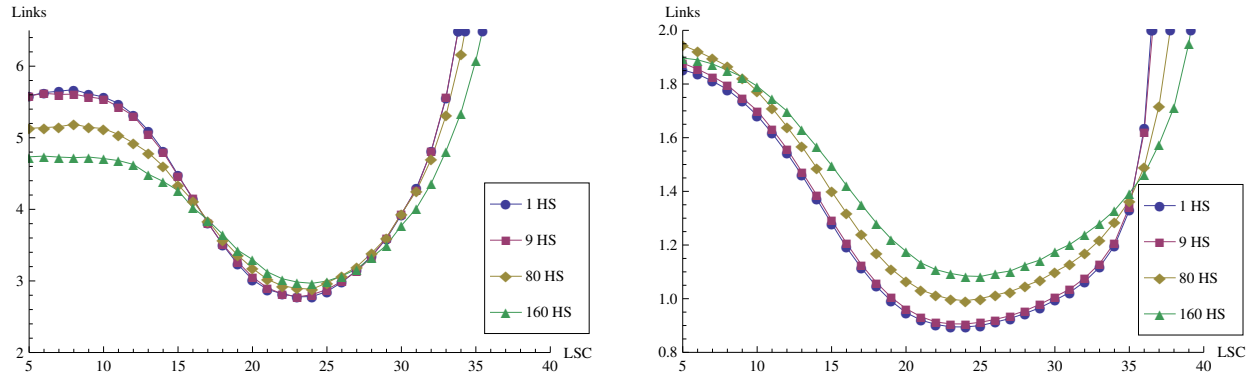
We may notice that up to 6.2% of the total WSN capacity (or discrete time $T_i : i =\backsim 9800$) there are no costs for storing and retrieving events. In other words, the first time that a sensor cannot store the events that it sensed occurs on $i =\backsim 9800$. Note that in Section 6.1 we defined this discrete time clock as the WSN lifetime. This discrete time is consistent with the γ value that we have calculated in Section 6.5 ($\gamma = HS/(pn)$).

For a low event level (up to 40% of the total storage), we can see that the difference between STORE and RETRIEVE is relatively small. That observation suites the results presented in Section 5.5 - at the beginning it is *"easy"* for DSC to locate available sensors for storage. However, as SM grows towards 80% the storage cost increments exponentially. Notice that even for 40% (where costs are very low) we have managed to increment WSN lifetime ~ 8 times more than it could be without the usage of our protocol.

8.3.2 STORE and RETRIEVE Vs. LSC Configuration

On the last section we demonstrate a single simulation where the LSC configuration was fixed to $LSC = 23$. In this section, we demonstrate simulation results for various values of LSC. Similarly to the last section, we demonstrate the STORE and RETRIEVE costs (equivalent to the two curves in Figure 8.1). However in this section we use different LSC values and four different sizes of hot-spot sets (including the 80 hot-spot configuration that we used in the last section). A quick look in Figure 8.2 reveals the reason for selecting $LSC = 23$ for the last section. We can see that this LSC value

produce the best results for the storage session.



(A) STORE (B) RETRIEVE

Fig. 8.2: Average number of links for the STORE and RETRIEVE sessions (see parameters in Table 8.1).

The x-axis in Figure 8.2 represents 35 different simulations that we conducted for variable LSC configurations. The y-axis measures the STORE and RETRIEVE costs in terms of DSC links (similarly to the y-axis from Figure 8.1). Note that retrieval costs are higher than those of storage, since `PUT` messages may also be routed to sensors that already had their storage capacity depleted. Also note that we observe the WSN on $\gamma = 0.8$, when the majority of the sensors in the network had their storage capacity depleted. Therefore, it is more difficult to track available sensors. Few observations are in place here:

1. Clearly, the STORE and RETRIEVE costs have an optimum point for each LSC configuration. Hence, if we can characterize that optimum point, we can optimize the chains lengths.

2. In Figure 8.2 we show an approximate costs of 3 links for storage and 1 link for retrieval. Notice that these costs are relatively low for the benefit of utilizing 80% from the entire storage capacity of the WSN. Or in other words, incrementing the WSN lifetime in an order of almost a whole magnitude.

3. An interesting observation is the RETRIEVE cost for a single hot-spot sensor (Figure (B)). Notice that RETRIEVE cost is only a little more than 0.8 links ($LSC \sim 23$). In other words, with a single sensor that generates 80% of the total capacity of the WSN we manage to store all events. Moreover, according to the retrieval costs, $\sim 80\%$ of the sensors in the WSN share their storage capacity with that single hot-spot sensor. We can think of this result as a long DSC chain that connects 80% of the sensor. Note that 80% of the sensors are the minimum number of sensors for the storage of the overflowed events.

4. As we increase the size of the hot-spot sensor group, the cost of RETRIEVE grows as well. The reason is that on several occasions, a regular sensor shares its storage capacity with more than one hot-spot sensor. Therefore, the storage and retrieval curves are shifted up for larger hot-spot groups.

5. Varying the LSC parameter demonstrates an important behavior. As we increment the LSC, the STORE and RETRIEVE costs decrease until an optimum point. If we pass the optimum point, these costs become growing exponentially. In Section 8.4 we will carefully study this behavior.

8.3.3 DSC Vs. Random Selection

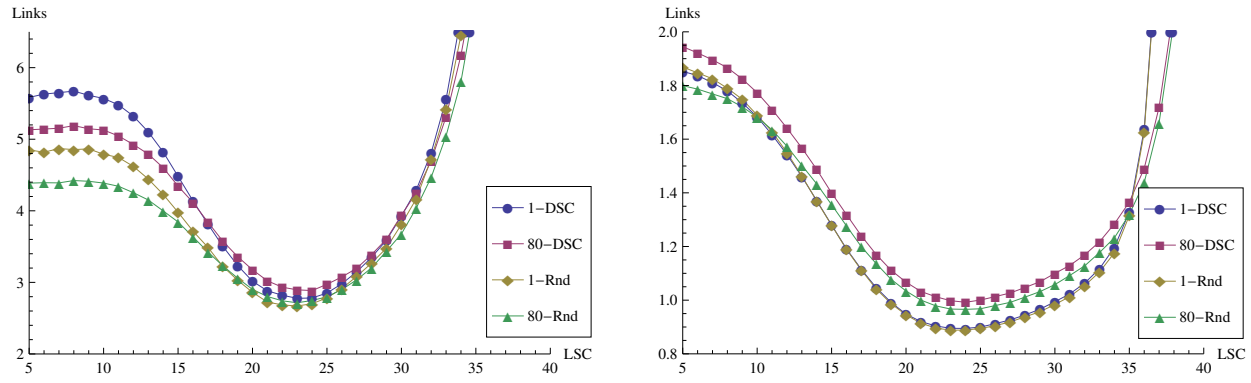
In Chapter 5 we described the Distributed Storage Chains (DSC) as a deterministic "random" walk. We named that walk after the expanding sets simulation (Section 5.5) where we compared the efficiency of our DSC with a random selection of geographical locations from the unit square. Notice that the principal innovation of our protocol (Chapter 7) is the usage of deterministic DSC. Similarly to the evaluation in Section 5.5, in this chapter we compare between the performance of our deterministic protocol with a random selection of geographical locations. In other words, we compare the way our protocol distribute overflowed events with the random selection of sensors by those who had their storage capacity depleted. We have repeated the simulation from Section 8.3.2 while changing the way DSC selects its links. Instead of calculating the DSC (Definition 7), we chose a random location (x, y) where $x, y = \mathcal{U}(0, 1)$ (uniformly chosen from the interval $[0, 1)$).

We wish to show that our protocol can be used in order to select sensors in the target field as efficiently as a random selection of geographical locations from the target area. However, we must remember that a random model has to consider the question of maintaining the structure of the chains. Therefore, a random model is a good reference, but it has additional costs.

Parameter	Values
n	1600
M	100
HS	$1, 9, 80, 160$ or $(1, Log(n), 5\%, 10\%)$
p	0.8
γ	$0 \leq \gamma \leq 0.8$
LSC	$0 - 40$
Environment	HS sensors are uniformly deployed
Storage Model	DSC Vs. Random selection

Tab. 8.2: Configuration for the DSC Vs. Random analysis.

In Figure 8.3 we demonstrate the differences between DSC and a random protocol (rXY from Sec. 5.5). It is very clear that the difference between the deterministic DSC and the random model is very small for every value of LSC. However, the difference becomes even smaller for the optimal LSC. In other words, if the LSC parameter can be chosen wisely, the cost of DSC and a random walk are the same. However, as we mentioned, there are additional costs for a random model. Therefore, although the results for the random selection of sensors seem a little better, they require from the chain structure to be maintained within the WSN. The great advantage of DSC is that the chain structure can be predicted based on sensors locations. As we showed in Section 5.5, the expansion rate of a rXY, or random selection of geographical locations, is the best we can get for the configuration of our WSN. Therefore, Figure 8.3 clearly demonstrate the efficiency of our protocol in locating available sensors and in maximizing the storage utilization of the WSN.



(A) STORE (B) RETRIEVE

Fig. 8.3: Comparison between DSC and random selection of sensors at 80% consumption of total storage capacity.

An important issue on the analysis of our protocol is the LSC parameter. For example, in Figure 8.3 (A) we demonstrate that if LSC parameter is not chosen wisely (e.g., $5 \leq LSC \leq 12$) the difference between our protocol and a random selection of sensors can be significant (~ 0.8 links per sensor node). Moreover, as we showed in Section 8.3 and also in this section, as we increment the LSC above its optimum value, the STORE and RETRIEVE costs increment exponentially. On the next section we will show how the variables of the traffic model (Chapter 6) are expressed in the calculation of LSC and how to optimize the STORE and RETRIEVE costs by choosing the appropriate LSC value.

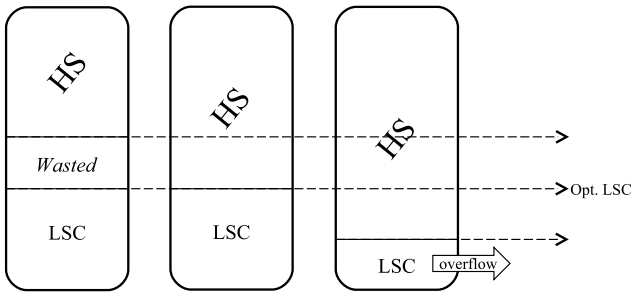


Fig. 8.4: Three possible cases for LSC values.

8.4 Optimal LSC analysis

As we mentioned in Section 6.4, there is a significant importance to the number of events that are generated by non-hot-spot regions. In order for the WSN to be efficient, a non-hot-spot sensor needs to know how many data units it may share. In other words, if a non-hot-spot sensor could have known how many events it would sense, it could have contributed the rest of its storage capacity to hot-spot sensors. However, since it is impossible to know the exact number of data units that every sensor would sense, we have to estimate a value to be used as Local Storage Consumption by all the sensors (regardless if they are hot-spots or not). In Section 8.3.2 we discussed the importance of LSC value by demonstrating the behavior of STORE and RETRIEVE costs for different LSC values. Moreover, we showed that for each configuration exist a LSC value that can minimize the link costs. Since LSC values must be set for all sensors before their deployment in field, there is a significant importance to the value that we chose. The reason is that we do not know (and cannot estimate) which sensors would eventually be deployed on hot-spot regions. Note that the design of WSN cannot assume the structure of the target area. If we could do that, we would have deployed on hot-spot regions sensors with larger storage capacities.

In Figure 8.4 we give an intuitive explanation to the behavior of the STORE and RETRIEVE curves (from Figures 8.2 and 8.3) around their optimum point. We may think about the rectangles in Figure 8.4 as if they represent the storage capacities of sensor nodes and the designation for each part of the storage. The figure in the middle demonstrates a situation, on which the LSC value is optimal. Therefore, this figure describes a non-hot-spot sensor that makes use of its entire local storage capacity and shares the rest with hot-spot sensors. On the right, we demonstrate a LSC that was chosen below its optimal value. On that case, non-hot-spot sensors share more storage capacity than they should have. Therefore, they are left with less storage capacity than they need. A non-hot-spot sensor that is included on a DSC of some hot-spot sensor would lose the entire storage capacity that it shares. In that case, a non-hot-spot sensor would also suffer from overflowed events. It would

have to use our protocol to store its additional events. Therefore, although hot-spot sensors could use less links, the STORE and RETRIEVE costs would be increased due to many non-hot-spot sensors that had their storage capacity depleted. Finally, on the left side of Figure 8.4 we demonstrate a non-hot-spot sensor that sets its LSC to a higher value than the number of events that it actually sense. In that case, a high LSC value generates difficulties for hot-spot sensors. The reason is, that non-hot-spot sensors share less storage capacity. Therefore, hot-spot sensors would have to generate more links (address more non-hot-spot sensors). Moreover, the storage capacity that is not utilized would get wasted (labeled as "Wasted in Figure 8.4), since hot-spot sensors cannot access it.

Parameter	Values
n	1600
M	100
HS	$1, 9, 80, 160$
p	$0.6 - 0.85$
γ	0.8
LSC	$0 - 70$
Environment	HS sensors are uniformly deployed

Tab. 8.3: Configuration for the varied p simulations.

In order to study the optimal values of the Local Storage Consumption (LSC), we repeat the simulation that we described in Section 8.3.2 with the configurations listed in Table 8.3. We use different p values in order to study their effect both on the curves and on the optimal LSC values. Figure 8.5 demonstrates three groups of curves that are similar to those describes in Sections 8.3.2 and 8.3.3. However, each of these groups describes simulation results for a specific p value, $p = 0.65, 0.75, 0.85$ (more results can be found in Appendix A Figure A.6). According to this figure, as we increment the values of p, the entire RETRIEVE curves are shifted to the left. Moreover, the retrieve costs in terms of links remain the same regardless the values of p (only for different LSC values).

Our explanation to the way that the curves are left-shifted (as p grows) is very simple. For example, in Figure 8.5 we observe the curve that represents $p = 0.65$ and $|HS| = 160$. As we can see, that curve has a minimum point at $LSC = 36$ data units. That minimum point refers to the the case in Figure 8.4 (middle), where non-hot-spot sensors consume 36 data units and share 64. If we increment the parameter p, non-hot-spot sensors will consume less than 36 data units. Therefore, maintaining the same $LSC = 36$, means that some of the Local Storage Consumption would get wasted (Figure 8.4 left). Moreover, incrementing the parameter p means that hot-spot sensors generate more events. Therefore, although they would need more shared storage capacity, the LSC parameter would not permit. In order to optimize our storage utilization, we will have to reduce the number of data units

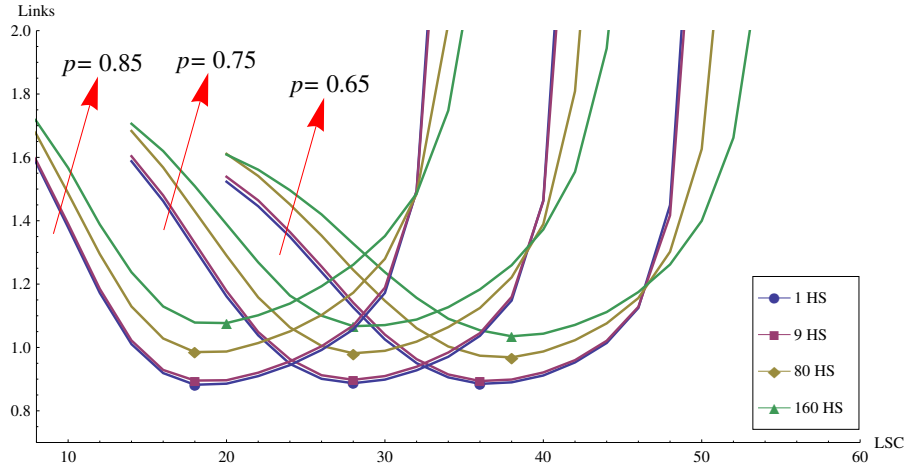
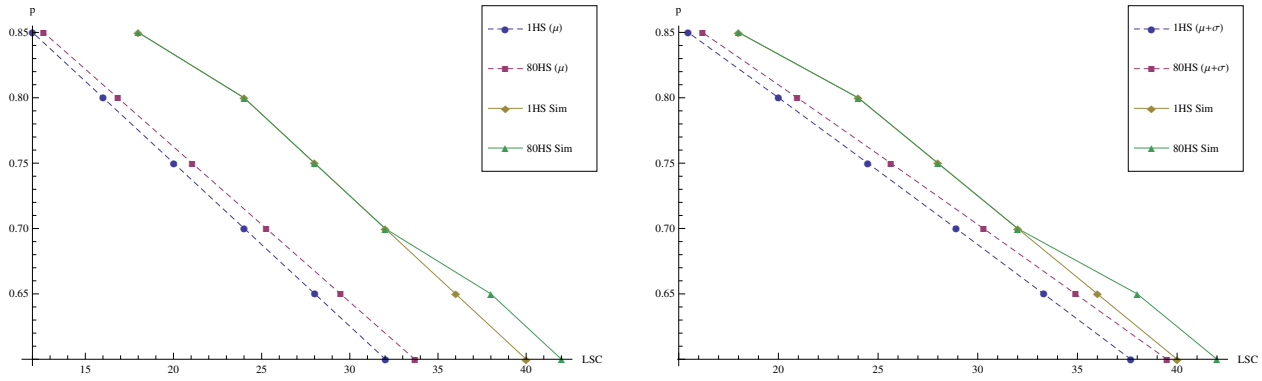


Fig. 8.5: RETRIEVE costs for different p values.

that can be used exclusively by non-hot-spot sensors and increment the amount of storage units that they share. In order to maintain optimization, the optimum points are shifted to the left (reducing the LSC value) while incrementing the probability p.



(A) Mean only (B) Mean + Variance

Fig. 8.6: Calculating optimal LSC with and without Variance.

In Section 6.4 we discussed the expected number of events that a non-hot-spot sensor generates during a simulation. We demonstrated (Figure 6.3) that non-hot-spot events have a binomial distribution. We discussed the importance of the mean and variance of this binomial and developed their formulas (Equation 6.5). In this section we would like to connect these theoretical formulas with the simulation results and the optimal values of the LSC. In Figure 8.5 we marked for every curve its optimal LSC value, which minimizes the RETRIEVE costs. We use these measures in order to construct a graph that relates between the hot-spot probability p and the LSC value that minimizes the RETRIEVE costs.

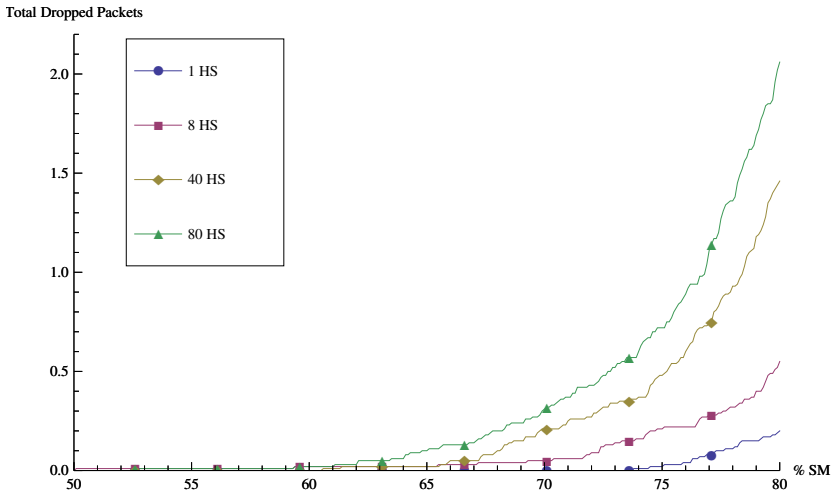
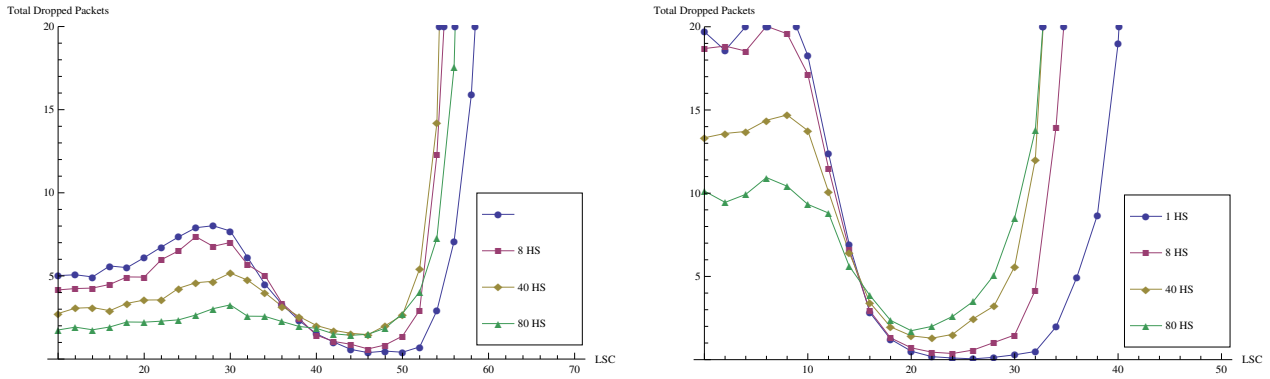


Fig. 8.7: Total number of dropped packets during a simulation ($n = 800$, $p = 0.8$ and $LSC = 27$).

In Figure 8.6 we present only the LSC values that minimizes the RETRIEVE costs. The continuous curves represents the minimum points for every p value from our simulations. The dashed curves represents theatrical LSC values that we have calculated. In Figure 8.6 (A) we use only the mean number of events that a non-hot-spot sensor sense. In other words, if we use these values of LSC we refer to the average case. However, we can see that the distance between the theoretical values that we calculated and the actual values, obtained by simulations, are very long. If we think about the binomial distribution and its Gaussian bell, there are a lot of sensors that produce more than the mean value. Actually, if we consider the standard division, at least 34.1% of the non-hot-spot sensors would sense more events than the expected value. In Figure 8.6 (B) we consider the theoretical LSC values as the sum of the mean and standard divisions (Equation 6.5). As we can see these values are much more adequate to the values that we received in our simulations. The results for hot-spot sets of sizes $|HS| = 9$ and 160 can be found in Appendix A (Figure A.7).

8.5 Number of Dropped Packets

In the beginning of this chapter, when we defined the configurations (Section 8.2), we defined a TTL for DSC packets that are sent as `PUT` messages. We wanted to avoid packets that are routed constantly in the WSN without finding a sensor with available storage capacity. We defined $TTL = 16$ which drops the packet after 16 *consecutive* `PUT` messages. Note that dropped packets mean that our protocol has failed to store overflowed events. In other words, a packet will be dropped when a DSC calculate 16 consecutive links to sensors who had their storage capacity depleted. Therefore,



(A) $p = 0.6$ (B) $p = 0.85$

Fig. 8.8: Number of dropped packets Vs. LSC (800 sensor nodes, $\gamma = 0.8$).

the most important evaluation is estimating how many packets where dropped by the protocol. In this section we demonstrate the behavior of our WSN as we vary the parameters n, p and LSC (see Table 8.3). We will show that the number of dropped packets is negligible.

Parameter	Values
n	$800, 1600$
M	100
HS	$1, 8, 9, 80, 160$
p	$0.6 - 0.85$
γ	0.8
LSC	$0 - 70$
Environment	HS sensors are uniformly deployed

Tab. 8.4: Configuration for the evaluation of dropped messages.

Figure 8.7 describes the number of dropped during a simulation for a WSN of size 800 sensor nodes with a LSC that is fixed to be 27 data units. Note that the LSC value that we chose is the optimal value (Section 8.4). Therefore, for this configuration, we expect to get the least number of dropped packets. The x-axis in Figure 8.7 refers to the discrete time on which the simulation is running (or to the percentage of the total System Memory that was consumed). As we can observe, no packets are dropped before $\backsim$ 60% of the WSN's storage was consumed. Note that for these configurations our protocol begins to distribute data units on 0.15% (for 1 HS) $\leq \gamma \leq 12.5\%$ (for 80 HS). Still, no packet is dropped before 60% of the WSN storage capacity was consumed. Also, we may observe that as we increment the size of the HS set, the number of dropped packets increases. This is easy to explain since more DSCs are searching for sensors with available storage capacity. Therefore, the probability of a packet to be dropped increases with the number of chains.

In Figure 8.8 we show the number of dropped packets in the end of a simulation ($\gamma = 0.8$). We vary the LSC around its optimal value and demonstrate the simulation results for two values of p (0.6 and 0.85). As we can see, the behavior of these curves is very similar to the behavior of the STORE and RETRIEVE costs. The minimum number of dropped packets is achieved at the optimal LSC value.

An important observation is that for low LSC values, the number of dropped packets is reasonable. Five dropped packets and twenty dropped packets are equal to 0.006% and 0.025% of the total storage capacity. Moreover, as we show in Appendix A (Figure A.9) the percentage and the behaviors of dropped packets for a 1600 sensor networks are with the same proportions. However, for the optimal LSC value, we get a constant number of dropped packets that is lower than 2 packets (ever for a $|HS| = 0.1n$).

We conclude that the number of dropped packets by the protocol is very low and negligible. Moreover, we can implement a specific solutions for packets that are routed more than 16 consecutive hops. An example for a specific solution can be greedy routing towards the closest sensor with available storage capacity. However, since the number of dropped packets is very low, we will not develop a solution in this work.

8.6 Load Balancing evaluation

As the total WSN storage consumption grows, the STORE session becomes much more expensive. Since more and more sensors have their storage capacity depleted, it becomes more difficult to find sensors with available storage capacity. Therefore, more DSC links are constructed in order to find available sensors. In this section, we would like to verify that the WSN maintain steady *load balancing*. Formally, we would like to show that the *load*, or the number of times regions are visited by DSC, is *balanced* over the unit square. To demonstrate the WSN load balancing, we present the load results of a 1600 sensor network. In Figure 8.9, each cell represent the Voronoi cell of a sensor. Its hue is set with respect to the number of times it was visited by any DSC chain. This figure demonstrate that there is a clear correlation between the size of a Voronoi cell and the number of times it was visited, the larger the cell the brighter it appears. That correlation reinforces the results that we received on the static construction (Sec.4.5 and fig. 4.5). Therefore, we may say that the links $CW^i(s)$ (Definition 7) are uniformly distributed over the unit surface. Moreover, the more smooth the deployment of sensors in field, the better the load balancing of the WSN. Note that the average number of calls to a sensor is 2.7.

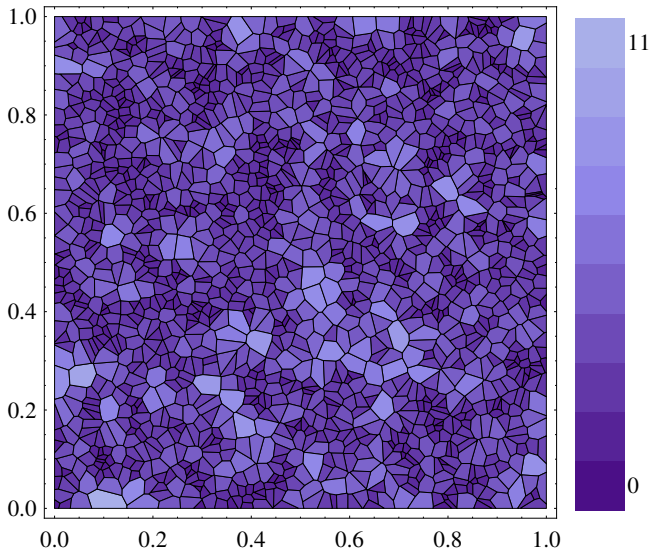


Fig. 8.9: Load Balancing Analysis for a 1600 WSN where $p = 0.8$ and $LSC = 23$. Number of visits (0-11) that each sensor region receives from DSCs. The darker the region color, the less it is visited.

8.7 Varying the Number of Hot-Spots

In the previous simulations we varied the HS component for specific hot-spot sets of sizes $1, Log(n), 5\%, 10\%$. Accordingly, we demonstrated how the WSN performances respond to the change in HS. However, since we only used four sizes of hot-spot sets, we had to imagine the gaps between all $|HS|$ values, in order to understand the behavior. In this section we show more values of HS set sizes in order to understand more deeply how does the number of hot-spot sensors affect the performances of our protocol. The configuration for these simulation is given in table 8.5.

Parameter	Values
n	800
M	100
HS	$1 - 400$
p	$0.6 - 0.85$
γ	$0.6 - 0.8$
LSC	$0 - 70$
Environment	HS sensors are uniformly deployed

Tab. 8.5: Configuration for the evaluation of dropped messages.

In Figure 8.10 we use $p = 0.8$ and vary the number of hot-spot sensors $0 \leq |HS| \leq 400$ Notice that our restriction on $|HS|$ according to Equation 6.1 is $|HS| < pn$. Also notice that 400 sensor

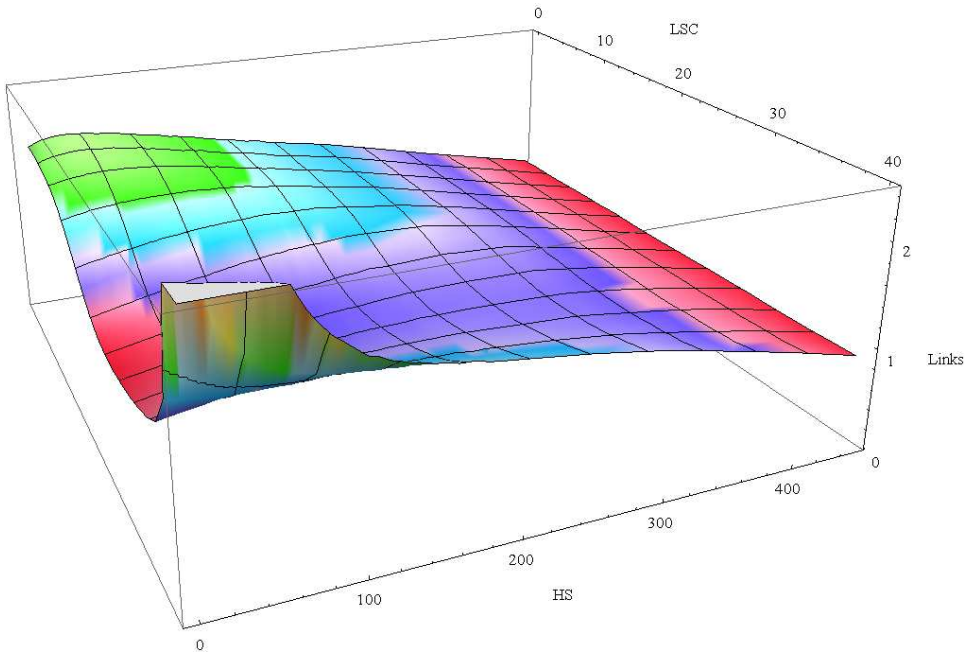


Fig. 8.10: Varying the number of hot-spots. $n = 800$ $p = 0.8$.

nodes sustain this relation. In Figure 8.10, the x-axis describes the number of hot-spot sensors, the y-axis describes the LSC value and the z-axis describes the average number of links for RETRIEVE. For low values of $|HS|$ we can observe the behavior that we saw in the previous chapters. The WSN has its optimum point for LSC, around the sum of mean and standard division ($\mu + \sigma$). As we increment the number of hot-spot, the LSC optimum point raises and, therefore, the number of links. However, beyond some $|HS|$ value, surface becomes more strait. This can be explained by the restriction that we defined on Equation 6.1. In other words, for these values, no exists significant difference between hot-spots and non-hot-spot sensors since they generate events with (almost) the same probability.

In Figure 8.11 we demonstrate three surfaces that represent different p configurations. It is clear that the characteristics of all surfaces are the same. However an interesting observation that we can see here is that as we reduce the p value, its cost for higher LSC value becomes less extreme as we increase the hot-spot set (the right "wings" of the surfaces become smaller as we reduce the parameter p. On Appendix A we demonstrate more angles of these three surfaces in Figure A.10.

8.8 Deployment of Hot-Spot sensors

In the previous sections we presented simulation results for uniform deployments of Host-Spot sensors (Figure 8.12-C) over the unit surface. In this section we would like to discuss the implication

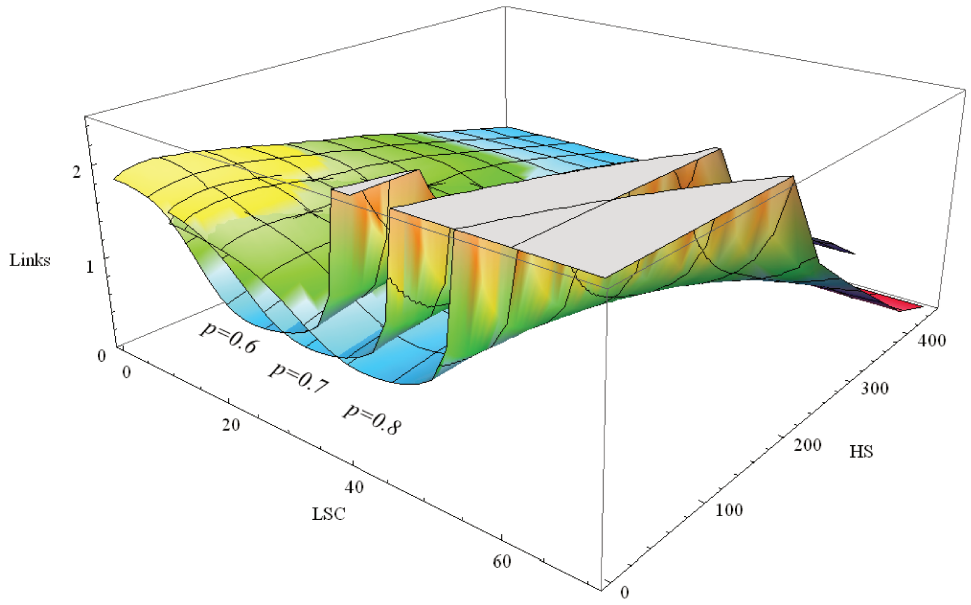


Fig. 8.11: Varying the number of hot-spots for 3 Different p values and $n = 800$.

of different configuration of hot-spot regions. Generally, a hot-spot region describes a phenomena that occur on the geographical field, independently of the deployment of sensors. In that case, a hot-spot region would be a combination of several sensor regions with a *geographical sequence*. Examples for geographical sequence are given in Figure 8.12-(A) and (B). We named these configurations *"valley"* and *"river"* respectively according to their shapes. These configurations can be useful on the evaluation of wildlife habitat monitoring (for example (Luo et al., 2007)). In other words, if we know that there should be a high activity of animals near the river or on a valley, we could have deployed there more sensors, or more storage reserves. But what if we cannot detect those specific locations? What if exist many valleys/rivers and we cannot characterize the hot-spot region before the deployment?

Figure 8.13 (and A.8 on Appendix A) describe a simulation that we conducted, over of hot-spot configurations (A) and (B) (Figure 8.12). First, we selected the hot-spot sensors with respect to these figures. That is, 33 sensors where selected for the valley and 60 for the river configurations. We simulated the protocol with a 80% probability for an event to occur on the valley (or river). Then, we compared the protocol results with a random selection of EOC (like on the previous simulations). As we can see, the results present the same characteristic that we discussed on previous simulations. DSC "selects" host sensors with the same efficiency as random selection does. Moreover, since the DSC are dependent on the location of the sensor who had its storage capacity depleted and since these locations are very close, we would expect that a random selection would be much better. However the results shows that the difference is very small. Moreover, we selected the location of the valley to

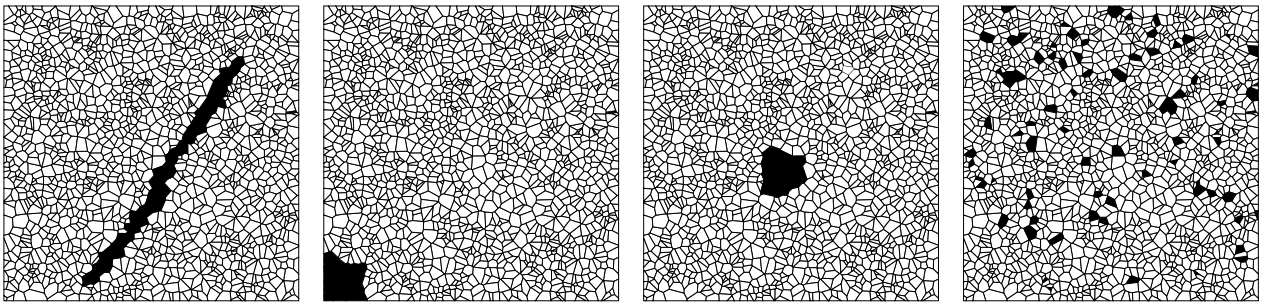


Fig. 8.12: Examples for deployment of hot-spot that cannot be predicted: (A) River of 60 hot-spot sensors. (B) Valley of 33 hot-spot sensors located in the bottom left. (C) Valley of 26 hot-spot sensors located in the center. (D) Random deployment of 80 sensors.

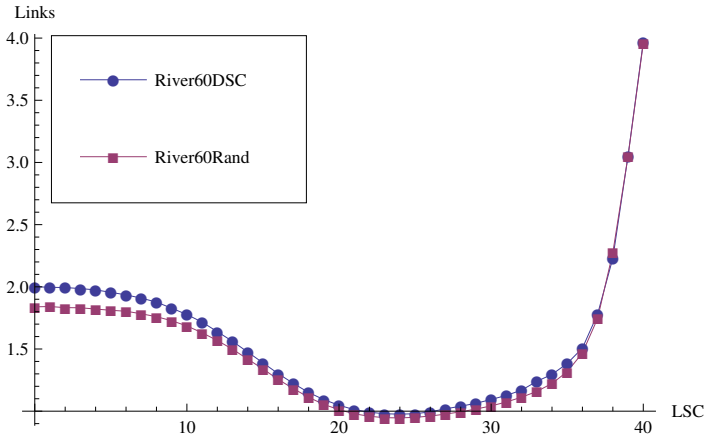


Fig. 8.13: Simulation results for the river deployments of hot-spot sensors (Figure 8.12 (A)).

be next to the axes origins. Since DSC select its links by summing the x and y coordinates, it should begin with a *slow progress* due to low values of x and y. However, based on the curves we can say that, eventually, a slow progress does not have a great effect on the total WSN costs. The results for the other two valley configurations can be found on Appendix A, Figure A.8.

8.9 Estimating the Energy Consumption

Throughout this chapter, we pointed out that the essence of our distributed storage model is in its ability to *deterministically* enable *"random"* interactions between sensor nodes. Our goal was to evaluate the performances of DSC, (overlay network) rather than the performances of the network layer (GPSR). However, one must bear in mind that each DSC link has its energy costs, due to transmission and re-transmissions between sensor nodes (see Section 5.1). Moreover, in Chapter 3 we

discussed different WSN architectures and their implication on energy costs. Specifically, in Section 3.4 we have developed equations to calculate the energy consumption in a WSN. We demonstrated the relation between direct and multi-hop transmissions. However, there is no clear conclusion, to which is the best transmission method at each hop. We demonstrated how the proportion between the transceiver and amplification energies (E_{elec} and ε_{amp} in Equation 3.6) can be used to determine which is the best transmission method. In this chapter, similarly to Liao et al. (Liao & Wu, 2008), we estimate the energy costs of our protocol, based on the direct transmission method. This type of calculation provides a good *upper bound* for the actual energy cost. As we will show later, we can use the results of direct transmission and modify the proportions between E_{elec} and ε_{amp} in order to control the upper bound of energy cost. We list the variables for our energy simulations in Table 8.6.

Parameter	Values
n	$800 - 3200$
M	100
HS	$1, Log(n), 5\%, 10\%$
p	$0.6 - 0.85$
γ	$0.6 - 0.8$
LSC	$0 - 70$
Environment	HS sensors are uniformly deployed
Energy constants	$E_{elec} = 50\ [nJ/bit]$ and $\varepsilon_{amp} = 100\ [pJ/bit/m^2]$
WSN density	trans. radius $r = 50m$; 1 Node/ 83 m^2

Tab. 8.6: Configuration for the evaluation of dropped messages.

8.9.1 Lengths of DSC Links

Logically, the most important measure for energy evaluation is the distances that packets have traveled due to our protocol. As we will show, the distance that packets travel contributes its square value to the total energy consumption. In this section, we will evaluate the distribution of distances that are derived by the definition of DSC (Definition 7). Remember that DSC routs `PUT` messages in strait lines (see an example on Figure 5.4). In other words, the packets are routed between points with similar x or y cordinates. Therefore, the largest distance that a packet may travel is 1. In order to study the distribution of distances d, we conducted the following simulation. We selected a random point $p = (x, y)$ where $x, y = \mathcal{U}(0, 1)$ are selected uniformity, on the same way that we have selected the sensor locations. Then, we initiate a DSC from that point p. The resulted DSC contained a sequence of links $DSC(p) = p \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \cdots \rightarrow l_k$. We measured the Euclidean distance between every consecutive links l_{i-1}, l_i. In Figure 8.14 we demonstrate the distance distribution for

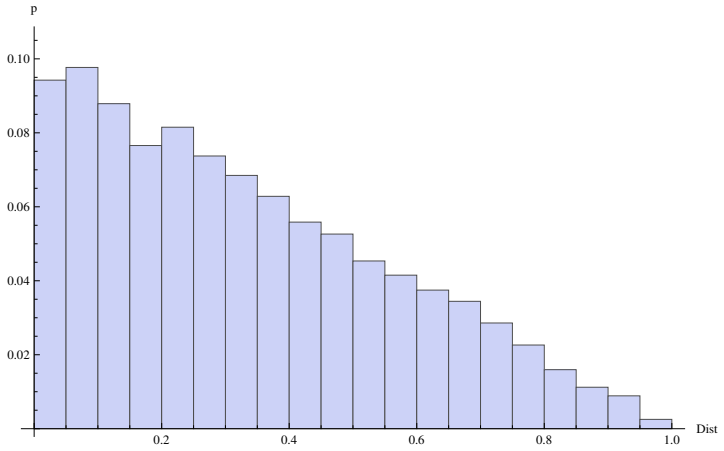


Fig. 8.14: Distance distribution for 100 DSCs, each of size $|DSC_i| = 100$ that begins on random locations.

100 random points p_i that generates DSCs of size $|DSC_i| = 100$ links each.

An interesting observation is that the exact distribution can be archives by combining two uniform variables. Let X, Y be two uniform variables $X, Y = \mathcal{U}(0, 1)$. The distributions of $Z = |X - Y|$ or $Z = |X + Y - 1|$ describes the distribution of our link distances perfectly. Note that in our case, the only random variables are the location $p = (x, y)$ of the point that initiates the DSC. Therefore, the rest of the chain is deterministic. Our explanation for these similar behaviors is due to the modulo 1 operator.

We can see that the probability for having long range links decreases with their length. In other words, short range transmissions are of the highest probability, while transmissions to the maximum distance 1 are of low probability. This observation is very important in the analysis of our protocol. It demonstrates that our protocol consumes most of its energy on short range transmission. However, even the length of short distance transmissions is dependent on the target field size, as we demonstrate on the next section.

8.9.2 Calculating the Energy Consumption

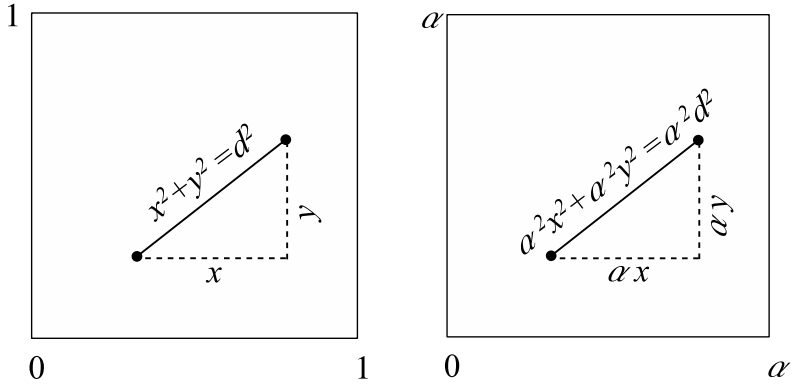
Equation 3.4 (Section 3.4.1) gives the energy consumption for a single packet of size k bits, which is transmitted to the distance $nr = d$. Notice that the notation n on that equation, represents number of hops (do not confuse with our network size n). Since we deal with direct transmission, we will use the notation d for the transmission distance. In this section we will explain how we measure the energy, consumed by our protocol for some discrete time T_t. To do so, we calculate the total energy that was consumed by the protocol during $0 \leq i \leq t$ for every value of t:

$$E_{direct}(t) = \sum_{i=0}^{t} k(2E_{elec}R_i + \varepsilon_{amp}{d_i}^2)$$

$$E_{direct}(t) = 2kE_{elec}\sum_{i=0}^{t} R_i + k\varepsilon_{amp}\sum_{i=0}^{t} {d_i}^2 \tag{8.1}$$

where the parameter R_i refers to the number of packets, produced by the protocol for storing an event on the discrete time i (number of `PUT` messages). Also, d_i refer to the sum of distances that same `PUT` message has passed until it was stored (or dropped). During a simulation we can sum these two vectors for every discrete time T_i. Notice that if have used a multi-hop fashion, R_i and d_i would have to be dependent also on the relation between E_{elec} and ε_{amp} (Equation 3.6). Therefore, our results would have been valid only for one configuration (of E_{elec} and ε_{amp}). Moreover, we can use Equation 8.1 and manipulate the proportion between E_{elec} and ε_{amp} (increase or decrease the direct transmission cost) according to Equation 3.6 and, therefore, provide a tighter upper bound to the energy cost.

As we mentioned, our WSN is deployed within the unit square. In order to scale our transmission results to a real target area we have to demonstrate how does the parameter d_i scales with target area. In Figure 8.15 we show, by simple geometry, how some distance d on the unit square, scales to the distance αd on a target area of size α^2.



(A) Unit Square (B) α^2 target area

Fig. 8.15: Scaling the distance that a packet travels.

We can use the description in Figure 8.15 to scale all the distances d_i. However, in order to compare between different configurations of WSN, scaling the size of the target area alone is not

enough. As we demonstrate in Section 3.3.1 (Table 3.1), there is a significant importance to the WSN density $\mu(r) = (n\pi r^2)/A$ measure. This measure considers both the size n of the WSN, the transmission radius r, and the size of the target area A. In order to compare energy consumption results between different WSN configurations, one has to consider the measure $\mu(r)$.

In this section we use the density configuration from the work of Liao et al., (Liao & Wu, 2008) in order to give our energy results some real values. We begin by explaining how we calculate the parameter α (Figure 8.15). In order to obtain the same WSN density we have calculate the proportion between our density $\mu(r)$ and the density that we compare with $\mu(r_c)$. Since we chose the same transmission radius $r = 50\ m$, we get:

$$\alpha^2 = \frac{\mu(r)}{\mu(r_c)} = \frac{n\pi r^2}{1} \cdot \frac{A_c}{n_c \pi r^2} = n \cdot \frac{500^2}{3000} \simeq 83n$$

In other words, since we use the same transmission range, we got $\alpha^2 = 83n$ which is the WSN density (1 Node / 83 m^2). Multiplying the WSN size by the node density α would yield an equivalent WSN to the one described in (Liao & Wu, 2008). Also, we use the same energy constants $E_{elec} = 50$ $[nJ/bit]$ and $\varepsilon_{amp} = 100$ $[pJ/bit/m^2]$. These parameters are slightly better than the current state-of-the-art Bluetooth specifications (700 Kbps radio that operate at 2.7 $[V]$ and 30 $[mA]$, or 115 $[nJ/bit]$ (Heinzelman et al., 2000)). Finally, we assume packet size of 8 bytes (similar to the store packets in (Liao & Wu, 2008)). Applying these constrains into Equation 8.1, we get:

$$E_{direct}(t) = 2kE_{elec} \sum_{i=0}^{t} R_i + k\varepsilon_{amp} \sum_{i=0}^{t} {d_i}^2 \alpha^2$$

$$E_{direct}(t) = 2kE_{elec} \sum_{i=0}^{t} R_i + k\varepsilon_{amp} \frac{A_c}{n_c} \cdot n \cdot \sum_{i=0}^{t} {d_i}^2 \tag{8.2}$$

where the relation $\frac{A_c}{n_c} = 83$ and the parameter n is the network size that we simulate.

8.9.3 Total Energy Consumption

In this section we calculate the energy consumption for two network sizes ($n = 800, 1600$). We will preserve the same network density (see Section 3.3.2). In other words, we maintain the same number of sensor nodes within the transmission range, as described in Equation 3.1. We use the same node density (Liao & Wu, 2008) (1 Node / $9 \times 9\ m^2$).

Figure 8.16 shows simulation results in terms of total energy consumption for 1600 sensors that are depleted over $365 \times 365\ m^2$ target area (node density μ = 1 Node / $9 \times 9\ m^2$). The x-axis describes the discrete time T_i in percentage from the total storage consumption of the WSN (SM - System Memory). The- y-axis describes the total energy consumption in Joules. In other words, the total energy consumption in order to utilize x percents of the total storage capacity. Note that this figure shows the total energy consumption for $0 \leq LSC \leq 40$ (same LSC range that we used on the previous sections). Therefore the total energy consumption vary between the optimal and worst LSC values. The curve in this figure represents the mean total energy consumption (mean among all LSC values) and the markers represent energy range (between different LSC values).

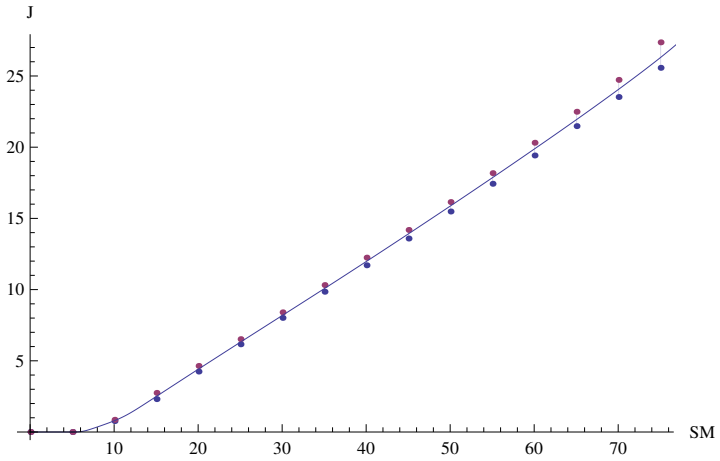


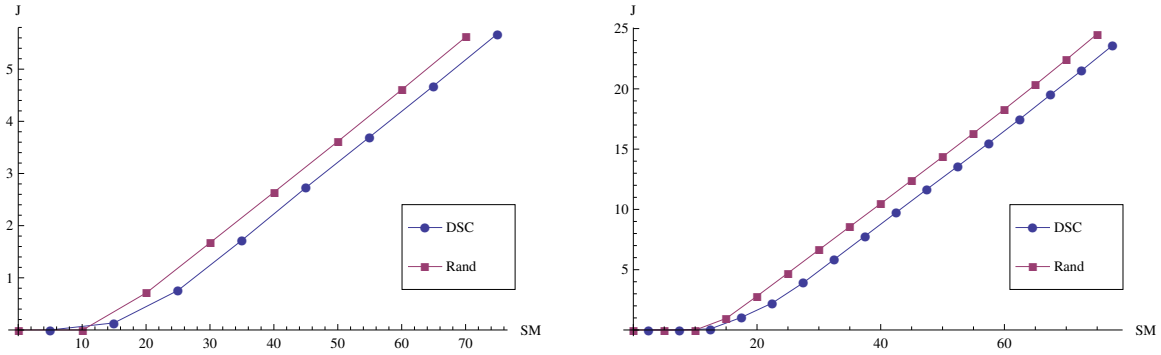
Fig. 8.16: Energy Analysis for a 1600 WSN where $|HS| = 160$ sensors. The WSN is deployed over a $365 \times 365\ m^2$ target area. $p = 0.8$ and $0 \leq LSC \leq 40$.

Similarly to Figure 8.1, we can notice that on the first 5% of the simulation there is no energy consumption. Accordingly, that can be explained, since in the beginning of the simulation all sensor nodes have available storage capacity. Therefore, no transmission is needed and all of the events are stored locally. However, as the simulation continues, we may notice that energy consumption increase (almost) linearly. We note that for the utilization of 80% from the total storage capacity the total energy consumption value is $\sim 27\ [Joule]$. This value meets the average values of $16.9 \cdot 10^{-}3$ [Joule] per sensor node. We consider this energy cost as reasonable cost according to (Kahn et al., 1999) that suggests a daily energy consumption of $1\ [Joule/day]$ per sensor node. Moreover, note that this coast extends the WSN lifetime from 8% to 80% storage consumption. In other words, we prolong the WSN lifetime by a hole magnitude with an energy cost of $16.9 \cdot 10^{-}3$ [Joule] per sensor.

The most interesting observation in Figure 8.16 is that the energy consumption is hardly affected when we change the LSC values. Notice that we have conducted simulation for LSC values of $0 \leq LSC \leq 40$. The range markers in this figure represent the minimum and maximum energy costs. We

can see that the amplitude of these curves (as we modify LSC) are small in the end of the simulations and invisible on the beginning. We explain this behavior as follows. In Section 8.3.2, we demonstrated the $STORE$ costs while varying the LSC parameter. The results in that section demonstrated that the differences between the optimal LSC and the worst case LSC were $3 \leq links \leq 6$ per sensor node. This means $1600 \cdot 3 = 4800$ links. However if we consider Equation 8.2 for 80% of the WSN storage capacity, the total number of overflowed events are $0.8 \cdot pnM - |HS| \cdot M$, or $94,400$ events for the 1600 sensor network. As we can see, the proportions between these numbers are very small. Therefore, the majority of the consumed energy is invested in the actual transmission of events to be stored in available sensors.

In Figure 8.17 we show the total energy consumption for 800 and 1600 WSN and compare its performances between our DSC protocol and a random walk rXY as described in Section 5.5. We maintain density factor of μ = 1 Node / 9×9 m^2 (deployments over 258×258 and 365×365 m^2 target areas). Also, both hot-spot sets are both of $0.1 \cdot n$.

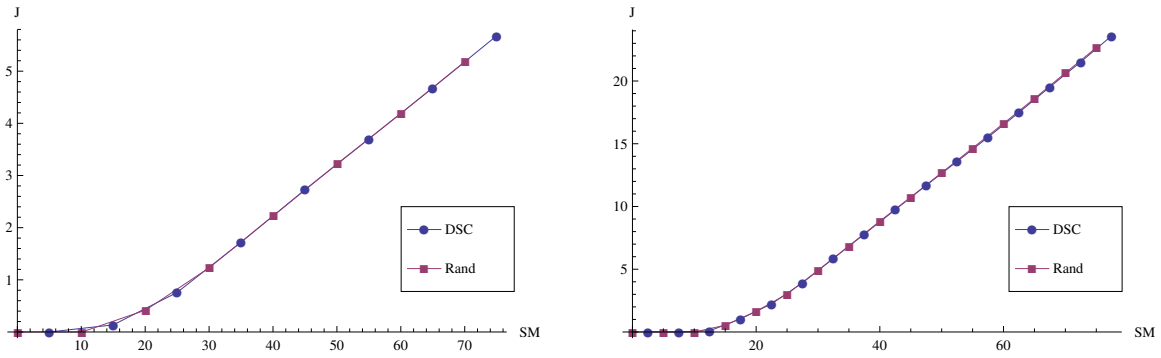


(A) $n = 1800$ (B) $n = 1600$

Fig. 8.17: Energy Analysis for DSC Vs. Random walk rXY, where $|HS| = 80$ and 160 sensors respectively. The WSN is deployed over a 258×258 and 365×365 m^2 target areas respectively and $p = 0.8$.

Note that the results in Figure 8.17 refers to the optimal LSC parameter from Section 8.3.2 ($LSC = 23$) that results in minimum energy consumption. It is very clear that energy consumption of DSC is lower than rXY. We explain this observation by the fact that rXY has longer transmission ranges. Note that while DSC transmit packets only horizontally and vertically, rXY can transmit diagonally. Therefore, rXY transmission range, within the unit surface, is of $0 \leq range \leq \sqrt{2}$ while DSC transmission range is $0 \leq range \leq 1$. However, the energy cost of both methods is on the same magnitude in terms of average energy consumption per sensor node. In order to make a more fare compression between DSC and a random walk, we simulated a random walk that progresses only horizontally and vertically. In other words, it selects random coordinates on the x and y axis alternately (similarily to the DSC, Definition 7). Accordingly, the results are compared with DSC

results in Figure 8.18.



(A) $n = 1800$ (B) $n = 1600$

Fig. 8.18: Energy Analysis for DSC Vs. Random walk, where $|HS| = 80$ and 160 sensors respectively. WSN deployed over a 258×258 and 365×365 m^2 target areas respectively and $p = 0.8$.

Figure 8.18 clearly demonstrate that the energy costs for a random algorithm are equal to those of DSC. Moreover, if we compare between the results from the two WSN sizes we can observe that while these two curves have the same behavior, their scales (y-axis) are different. Thus, even if we consider the average energy consumption, the 800 sensor network performances (with maximal energy consumption of $8.75 \cdot 10^{-3}$ Joule) and the 1600 sensor network (with maximal energy consumption of $16.9 \cdot 10^{-3}$ Joule) are of different magnitudes. Therefore, the size of the target area has a great impact on our protocol's transmission costs. This can be explained both intuitively and mathematically. Intuitively, as the target area grows, long distance routs become more and more expensive, since the distances between sensor nodes become greater. Mathematically, we can notice that the parameter d, in Equation 8.2, is calculated in the energy cost, using its square value.

8.9.4 Number of Events

Although the difference between our model and the model suggested by Liao et al., (Liao & Wu, 2008) are immense (see Section 8.1), a compartment between these results can be very helpful. However, it is important to put in mind that these two methods are different, in order to understand the relation between these results. We begin this section by dryly describing the results of these two methods. Then, we will compare between them and explain their differences.

On their paper (Liao & Wu, 2008), they simulate the distribution of $3,000$ data packets and received a maximum total energy consumption of 600 and 800 $[10^6 nJ]$ for two of their methods. In Figure 8.19 we present the energy costs for 800, 1600 and 3200 WSN sizes that we have normalized to the same network density as in (Liao & Wu, 2008). We have scaled the y-axis to match the $[10^6 nJ]$ measure as described by (Liao & Wu, 2008). For explanations on the $[10^6 nJ]$ measure, please see

Section 6.6. Notice that on the x-axis we measure events (instead of the percent from the total storage). As we can see, as the network size grows, it takes more time for it to consume energy. This can be explained, because the WSN lifetime depends on the WSN size n. The greater the n, it takes more time for hot-spot sensors to get their storage capacity depleted.

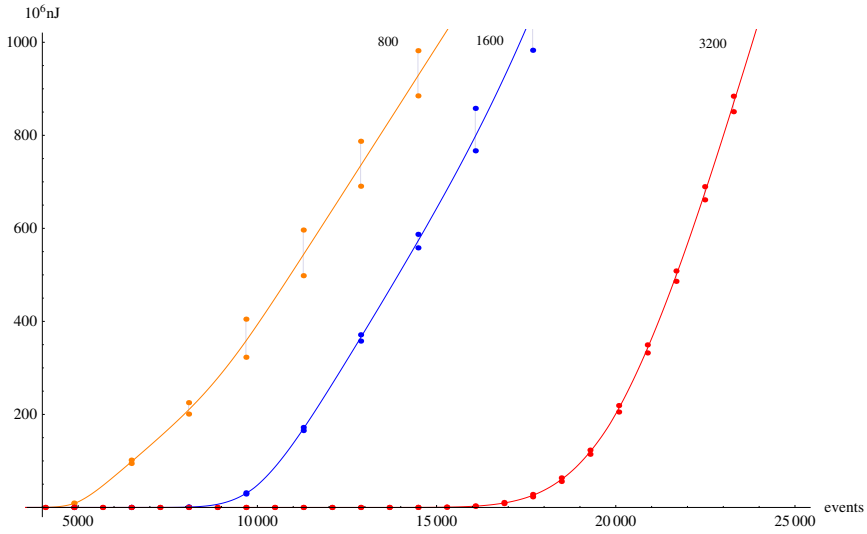


Fig. 8.19: Total energy consumption up to 1000 [10^6 nJ].

We can notice that as the network size grows, these curves become steeper. Since we chose to compare with the same network density, as the network size grow, so as the target area and, therefore, the energy costs grow as well. Notice that if we presented these simulations on for an average energy cost, the slops would become more equal. In order to compare between the energy costs, we fix the energy cost 600 $[10^6 nJ]$ and observe the number of overflowed events that were routed by the protocol. The simulation conducted by Liao et al., have managed to route 2500 events while 100 were dropped, we have managed to store 5400 ($n = 800$), 3800 ($n = 1600$) and 1700 ($n = 3200$) events while less than 2 packets where dropped.

Important observations should be put in place for this compartment: In this section we used the parameters of (Liao & Wu, 2008) more for getting real data rather than comparing. Our storage schemas is very different:

1. The main difference between these two models is that we distribute the load uniformly over the WSN target area. On the other hand Liao et al., directs overflowed events to close neighbors. Therefore, while they economize energy due to the short distances, they create coverage holes within the WSN. Thus, if we have measured the load balancing of their simulation as we measured in Section 8.6 we could have seen dark rings that encircle each hot-spot sensor.

2. Accordingly there is a difference in the energy costs between our configuration where $n = 3200$ sensors and their configuration with 3000 senors. The differences could be explained, because our packets travel to a longer distances (uniform distribution Vs. coverage holes). However, they do not specify how many overflowed data packets where actually stored. They specify 4% of dropped packets while we present $\backsim 0$ (less than two packets). They stop the simulation after 0.003% of the total WSN capacity.

3. An important aspect of our work was analyzing the traffic model that we used for simulations. However, Liao et al., did not define how data is generated on their model. Moreover, it seem that the only sensor that generate events are the HS sensors. Therefore, that model do not consider the interference from non to spot sensors, which is an important parameter that affected our results a lot.

4. Liao et al., considers two types of packets for storing and retrieval of data. We considered only one packet.

5. They only considered one case, where $|HS| = 4$. They do not show the WSN behavior for different HS or fordifferent n values.

Chapter 9

Conclusions

"Wenn ihr wollt, ist es kein Märchen" (German)
"Im TirtZu, Ein Zo Agada" (Hebrew)
"If you will it, it is no dream. (English)"
"Se quiseres, não será um sonho (Português)"

Benjamin Theodor Ze'ev Herzl (1860 - 1904)

In this work we have presented a distributed protocol for storage aggregation on WSN. We began with an implementation of a static expander graph that is based on the work of Gabber and Galil (Gabber & Galil, 1979). We described the theory underlying that motivated us to use this kind of graph. We continued and developed a Distributed Storage Chain (DSC) mechanism, which is actually a dynamic implementation of sub graphs from the complete expander. We showed through experimental design that those sub graphs are also expanders. We established a storage aggregation protocol that is based on these dynamic expanders. Finally, we formulated a traffic model for hot-spot environment and evaluated the performances of our protocol via simulations that consider that kind environment.

We evaluated the performances of our protocol by simulating sensor networks of up to 6400 sensor nodes. We demonstrate that DSC can be used as an efficient *deterministic "search-function"* that can cover the entire WSN with the same efficiency as random selection of geographical locations. Sensor networks that implement our protocol may operate under unfairness conditions between sensors (or between sensor regions). Moreover, since our protocol is an application layer protocol, no hardware modifications are needed. In this work we simulated various configurations of hot-spot regions. We varied the probability of hot-spot regions to generate events, their number and the way they are distributed over the target field. Also, we varied the WSN size, the number of events that sensor may store locally (LSC) and the percentage of total WSN storage that is consumed. Our sim-

ulations showed that DSC may tolerate up to a single sensor that may generate alone 80% of the total network's events. In this work, we described three main contributions and innovations:

First. We developed a *distributed geographic expander overlay network* for WSNs that are uniformly distributed over a target area. We measured the network performances through extensive simulations and found that its behavior is similar to that of an expander graph. In other words, we have managed to construct a WSN whose diameter scales with its size, while the number of links per sensor node remains constant (as we increment the network size). Moreover, we have managed to take the discrete expander graph and implement it over a continuous geographical region. Thus, we have established a *geographic expander network topology* for WSN, which can be constructed distributedly. Our next contribution is based on that topology.

Second. We developed a *Deterministic Geographic "Random" Walk* that uniformly *"samples"* the target area. This walk gains its uniqueness for its ability to generate *"random"* partnerships between sensor nodes. However, those partnerships are deterministic, by definition (Corollary 1) and can be calculated by a specific formula (Definition 7). We have shown that our Deterministic Geographic "Random" Walk is *smooth*, in the scenes that when it is repeated, it divides the unit surface uniformly. Our main innovation in this work is that our deterministic Geographic "Random" Walk allows sensors to select partners uniformly from the network as if they select random locations of a *Home Nodes*. We use this walk in order to select sensor groups that can share their resources. Moreover, its effectiveness derived from its uniform "sampling" characteristic. Hence, if conducted from different sensors locations it would not cause starvation on specific regions.

Third. We developed a distributed WSN protocol that can optimize the storage utilization of WSNs in order to prolong its lifetime. This protocol defines a *Deterministic Geographic "Random" Walk* (Section 5.5) and enable sensors to share their storage capacities. For WSN applications, our walk is of great impact, because it can both be implemented locally and no information needs to be kept in order to reconstruct it. Thus, no routing tables or instructions are maintained for the retrieval of data by the protocol (only local data). Even if a sensor dies, its storage chain can be reconstructed, based only on its location (or its region). We have demonstrated how to optimize our protocol, based on the traffic model that we have developed. We showed how its performances change with the number of hot-spot events that are generated on the target field. If we could characterize the target field according to our traffic model, it is possible to optimize the costs of our protocol. However, as we showed, even when nothing is known on the target area, our protocol still drastically improve the WSN lifetime.

9.1 More Applications

We began this work by saying that we use the storage problem as an example of resource aggregation. Actually, as we demonstrated throughout this work, the essence of our model is in its ability to *deterministically* enable *"random"* interactions between sensor nodes. In this section we describe more applications that can make use of this interesting property by adjusting our protocol platform.

1. On Many-to-One WSN architecture there is tremendous importance to the way data is transmitted to the sink station. Generally, we assume that the sink station is far from all sensor nodes, in the sense that transmission to the sink station consume more energy than transmission between sensor nodes. When sensors transmit in a multi-hop fashion, there is a risk that sensors that are closer to the sink station would carry the load of the whole WSN (Hung-Yu et al., 2005). Therefore, those sensors would rapidly get their energy capacity depleted and die. Heinzelman et al., (Heinzelman et al., 2000) suggest a clustering method on which the WSN is divided to clusters. Every cluster selects a cluster-head which transmit all the data towards the sink station. That way, they are able to process all the data on the cluster and, therefore, send less data to the sink station. Also, since cluster-heads are re-selected every defined time period, they achieve a more balanced energy consumption.

   We believe that our deterministic "random" walk can be used as an efficient method both for selecting the clusters and for selecting the cluster-heads. Since the success of their algorithm depends in the distribution of the cluster-heads over the target area, and since our method offers a uniform distribution, we believe that our method can contribute a lot to that clustering technique.

2. Sensor nodes may be requested to backup their data. Creating a backup on a nearby sensor might be an energy efficient solution, but sometimes backup is required due to environment concerns such as fire, flood, strong winds, etc. Therefore, we might need to make a backup of the data far from the sensor node. Moreover, we would like to avoid specific regions, to which most of the WSN data is sent. In other words, we would need a more uniform solution, such as our distributed storage model offer.

   Aly et al. (Aly, Kong, & Soljanin, 2008) describes the scenario where k events are very critical for the WSN. These events where discovered by k sensors, and are spreaded over the WSN in order to increment the probability of them to be found. Their basic idea is that every set of $(1+\epsilon)k$ sensors would contain these k events with a high probability. To do so, (Aly et al., 2008) uses a random walk to that disseminates these events over the WSN. Also, the number of events k and the WSN size n plays an important role, whether they are given to all the sensors

or being estimated by them. Our DSC protocol can be used for the same objective, without the knowledge of these parameters while using our deterministic "random" walk (Section 5.5).

3. Some non-real-time WSN uses data mule devices (Luo et al., 2007) in order to collect data from encountered locations. On that case, sensors should direct their data to regions where it might be encountered by a data mule. Since data mules may visit (or sample) the target area in a uniform manner, we can exploiting the deterministic *"random"* nature of the DSC. In other words, a sensor node can raise the probability for gathering its data by data mules by sending its information over a DSC.

4. Naor et al., (Naor & Wieder, 2007) suggested distributing process jobs between multiple processors. They used a distributed hash table that creates links between different processes. When some processor gets its CPU congested, processes can transfer themselves from one processor to another, based on those links. Their hash table was designed for 1-D interval where processes where "Home-Node" to sub-intervals. However, if we think about their problem in 2-D, our DSC can offer a 2-D distributed hash table that matches their needs.

9.2 Future Work

Throughout this work, and especially in the Simulations Chapter (Chapter 8) we demonstrated interesting observations both for our protocol and for our deterministic "random" walk. Our observations are based on various simulations, on which we varied most of the WSN variables in order to study our problem on different levels. In this section we will demonstrate some ideas that we have for future work:

1. In this work we deal with Homogenous WSN where all sensors have the same resource capacities. As we already mentioned, if we knew which regions are hot-spot regions in the scene that they generate more events than others we could deploy there sensors with higher storage capacities. However, we can think about this problem a little bit differently. Suppose that we have deployed a heterogeneous WSN, where some sensors have more resources than others. Suppose that sensors with large capacities are deployed uniformly and suppose that the target area has hot-spot regions. Therefore, our problem is how to transmit the overflowed events from hot-spot regions to sensors with large storage capacity. Since these sensors are deployed uniformly over the target area, we can easily use DSC to distribute data uniformly over the target field. Therefore, these chains would discover the large capacity sensors with high probability.

   We suggest exploring the implication of a heterogenous WSN on the performances of our protocol. Also we suggest exploring the proportion between high capacity and regular sensors in

order to find the combination that optimizes the WSN performances in terms of lifetime.

2. In this work we repeated and demonstrated by different types of simulations that our DSC can be used efficiently for creating interactions between sensors. In other words, DSC can be used as efficiently as random walk over the target area. Moreover, we showed that these chains uniformly "select" geographical locations (or Home-Nodes) in $[0,1)^2$. In Theory 1 we referred to the work of Gabber and Galil (Gabber & Galil, 1979) that justifies the use of their transform for our static network (see Section 4.5). However we could not analytically prove that DSC (which select the two transformations alternatively) actually selects points in $[0,1)^2$ on a uniform manner. In Appendix C we formulate the problem that we have tried to proof.

3. Liao et al., (Liao & Wu, 2008) suggest a multiple threshold mechanism for storage management in WSN. They define T_L storage threshold levels for all the sensors in the WSN. According to their model, the storage capacities of all sensor nodes are divided into T_L sub-capacities. Then, the WSN synchronizes the threshold level that is used currently. A sensor that had current storage threshold level depleted, would "direct" its events to one of its close neighbors. It would direct it to the closest neighbor that has available storage capacity (within the current trashily). In other words, according to their model, although that the sensor still has available storage capacity, it would utilize it only when the WSN gets the decision to move to the next threshold.

   We believe that the multi-threshold mechanism may be useful in our protocol. That is, our DSC would not enforce sensor nodes to completely get their storage capacity depleted. We believe that there could be much more future work in the area of multi-threshold. For example, let LSC be set to $LSC = 20\%$, suppose that a sensor had its current threshold capacity depleted, although it only utilized its $LSC = 20\%$ (from the current threshold capacity). We can say that this sensor is not a hot-spot sensor. Moreover, this sensor can determine locally that it is not a hot-spot sensor. Therefore, based on that information, it may use its next threshold level instead of constructing more DSC links. A sensor that consumed its entire threshold and was not directed by any other sensor, can deduce that it is a hot-spot sensor. We also believe that the number of thresholds and their sizes (e.g., they may divide the local storage capacity to T_L non-equal sub-capacities) are interesting design questions.

4. A fundamental issue that was repeated throughout our work is that sensor nodes should know their locations. Furthermore, we have discussed this issue and mentioned that location can be obtained either by a GPS or by some kind of localization technique (Section 4.4 and (Khan et al., 2009)). In our work, location is important for three main reasons:

(a) Events are associated with the locations where they were observed.

(b) GPSR (Karp & Kung, 2000) uses sensor locations to advance packets to their destination.

(c) DSC uses sensor locations in order to calculate links (Definition 7)

Since energy consumption by GPS is very high, it is recommended to reduce the use of GPS in WSN applications. An alternative is to use the triangulations between sensor nodes (Delaunay triangulation (Berg, 2000)) in order to estimate their locations. Khan et al. (Khan et al., 2009) suggest a localization technique for the 2-D unit surface that requires only 3 *anchor* sensors. These anchor sensors are either fixed sensors or high energy sensors with GPS that know their exact location. Moreover, these devices should *anchor* the WSN in a way that all sensors lie in the convex hull of the 3 anchors.

Depending on the WSN application, if the location of the observed events is not important we can use virtual coordinates. In other words, DSC can be implemented regardless of GPS. It is possible to use a distributed protocol to select anchors within the WSN. These anchors will define the unit square (which is essential for the modulo 1 operator). Based on the triangulations, each sensor could define its distance from the anchor sensors. Then, these distances can be converted to sensor's virtual coordinates.

Chapter 10

Conclusões

"No que diz respeito ao empenho, ao compromisso, ao esforço, à dedicação, não existe meio termo. Ou você faz uma coisa bem feita ou não faz"

Ayrton Senna (1960 - 1994)

Neste trabalho apresentou-se um protocolo distribuído para agregação de armazenamento em Redes de Sensores Sem Fio (RSSF). Começa-se pela implementação de um grafo estático tipo *expander* que é baseado no trabalho de Gabber-Galil (Gabber & Galil, 1979). Descreve-se a teoria basilar que motiva o uso deste tipo de grafo. Assim, desenvolveu-se uma cadeia de armazenamento distribuído (*Distributed Storage Chain - DSC*). Efetivamente, estas cadeias de armazenamento são uma implementação dinâmica de sub-grafos do grafo tipo *expander*. Mostrou-se nesse trabalho através de um projeto experimental, que estes sub-grafos são também do tipo expender. Finalmente, estabeleceu-se um protocolo de armazenamento agregado baseado nestes *expander*s dinâmicos.

Avaliou-se o desempenho deste protocolo através de RSSF de até 6400 nós de sensores. Demonstrou-se que o protocolo DSC pode ser usado como *uma função de busca determinística*, que consegue cobrir a rede inteira com eficiência de um passeio aleatório. Redes de sensores que implementarem este protocolo podem operar em condições de desigualdade dentre os sensores (ou dentre as regiões). Além disso, desde que este protocolo possa ser implementado na camada de aplicação, nenhuma alteração no hardware será necessária. Neste trabalho simula-se um grupo relativamente pequeno de sensores que geram 80% de atividades. Estas simulações mostram que a DSC pode tolerar até um único sensor que gera por si mesmo até 80% das atividades da RSSF. Também mostramos simulações de cenários específicos onde a sequência das regiões de sensores causam desigualdades dentro do campo alvo.

As três maiores contribuições e inovações desse trabalho são: **Primeiro.** Desenvolveu-se um

expander geográfico distribuído - rede overlay para RSSF que é distribuído uniformemente acima de um campo alvo. Mediu-se os desempenhos da rede através de várias simulações e descobriu-se que seu comportamento é similar a um grafo tipo *expander*. Em outras palavras, quando se aumenta o tamanho da rede, seu diâmetro não muda sua escala e o grau dos nós (numero dos enlaces por nó) mantêm-se constante. Portanto, estabiliza-se uma topologia RSSF geográfica que pode ser construída distributivamente. A próxima contribuição é baseada nesta topologia.

Segundo. Desenvolveu-se um Passeio Determinístico "Aleatório" Geográfico que consegue "mostrar" o campo alvo na forma unitária. Esse passeio ganhou sua singularidade por sua habilidade de gerar parcerias aleatórias dentre os sensores. Também, estas parcerias podem ser previstas e reconstruídas. Porém, estas parcerias são definitivamente determinísticas e dependem somente das locações dos sensores no campo alvo. Este novo passeio determinístico "aleatório" geográfico habilita os sensores a selecionarem uniformemente parceiros na rede como se tivessem selecionados locações aleatórias dentre as regiões dos *Home Nodes*. Usa-se este passeio para selecionar grupos de sensores que possam compartilhar suas reservas de armazenamento. Além disso, sua eficiência deriva de sua característica "uniforme" de amostragem. Por isso, quando conduzido de diferentes localizações dos sensores, não geraria "starvation" em regiões especificas.

Terceira. Desenvolveu-se um protocolo que otimiza a utilização da capacidade do armazenamento do RSSF. Este protocolo usa o *passeio determinístico "aleatório" geográfico* para possibilitar que os sensores possam compartilhar suas capacidades de armazenamento. Este passeio apresenta um grande impacto, pois nenhuma informação precisa ser mantida para os passeios poderem ser reconstruídos. Em outras palavras, nenhuma tabela de roteamento e nenhum tipo de instrução são necessários para retirar dados hospedados em algum sensor. Como resultado, mesmo que um sensor morra, a sua cadeia de armazenamento pode ser reconstruída a partir de apenas sua localização. Além disso, mais aplicações podem ser desenvolvidas usando esta plataforma de protocolo. Um exemplo é RSSF que usa *"data mule devices"* (Luo et al., 2007) para coletar dados das regiões encontradas. Um sensor pode aumentar a probabilidade para que seus dados sejam recuperados, através da geração de múltiplas copias de segurança de seus dados numa cadeia DSC. Assim, explora-se a característica determinística *"aleatória"* do DSC, e a sua implementação distribuída.

10.1 Mais Aplicações

Começa-se este trabalho dizendo-se que o problema do armazenamento é um exemplo da agregação de recursos. Na verdade, como foi demonstrado através deste trabalho, a essência do modelo aqui apresentado é a sua aptidão de *deterministicamente* habilitar interações *"aleatórias"* entre nós de sensores. Nesta seção, descreve-se mais aplicações que podem se valer desta propriedade tão

interessante, através do ajuste desta plataforma de protocolo apresentada

1. Na arquitetura de *Many-to-One* RSSF há uma enorme importância à forma com que os dados são transmitidos para a estação Sink. Geralmente, assume-se que a estação Sink fica longe de todos os nós de sensores, no sentido de que a transmissão para a estação Sink consome mais energia do que a transmissão entre os nós de sensores. Quando sensores transmitem em uma maneira *multi-hop* há o risco de que os sensores próximos à estação Sink acabem por carregar a carga de toda a RSSF (Hung-Yu et al., 2005). Desta forma, os sensores rapidamente esgotariam sua energia e morreriam. Heinzelman et al., (Heinzelman et al., 2000) sugere um método de *clustering* no qual a RSSF é dividida em *clusters*. Cada cluster seleciona seu *cluster-head* que transmite todos os dados para a estação Sink. Deste modo, os sensores tornam-se capazes de processar todos os dados no próprio cluster e assim enviam menos dados para a estação Sink. Além disso, como os *cluster-heads* são re-selecionados a cada período definido, eles atingem um consumo de energia mais balanceado.

   Acredita-se que o passeio determinístico "aleatório" sugerido, possa ser usado como um método muito eficiente tanto para seleção de *clusters* como para a seleção de *cluster-heads*. Uma vez que o sucesso de seus algoritmos dependem da distribuição de *cluster-heads* sobre um campo-alvo, e considerando que este método oferece uma distribuição uniforme, acredita-se que este trabalho sugerido aqui realmente possa contribuir para esta técnica de *clustering*.

2. Os nós de sensores podem precisar fazer copia de segurança de seus dados. Criar uma copia de segurança num sensor próximo pode ser uma solução eficiente para consumo de energia, mas muitas vezes a copia de segurança é requerida devido às questões do ambiente, como fogo, vendavais, etc. Assim, pode-se desejar fazer a copia de segurança de um sensor distante. Além disso, pode-se evitar algumas regiões específicas, para as quais a maioria dos dados da RSSF são enviados. Em outras palavras, necessita-se de uma solução uniforme, exatamente como o que este protocolo distribuído para agregação de armazenamento em RSSF, aqui apresentado, oferece.

   Aly et al. (Aly et al., 2008) descreve o cenário onde k eventos são muito críticos para a RSSF. Estes eventos foram indetificados por k sensores, e estão espalhados ao longo das RSSF para incrementar a probabilidade de que eles sejam encontrados. O princípio ideia é que cada conjunto de $(1+\epsilon)k$ sensores possua esses k eventos com alta probabilidade. Para isso, (Aly et al., 2008) utiliza um passeio aleatório que espalha estes eventos. Além disso, o número k de eventos e o tamanho n da RSSF são parâmetros importantes neste artigo (Aly et al., 2008), que são dados a todos os sensores ou estimados por eles. O protocolo DSC pode ser usado sem o conhecimento destes parâmetros e pode usar o Passeio "Aleatório" determinístico proposto

(Section 5.5).

3. Algumas RSSF de tempo não-real usam *mule devices* (Luo et al., 2007) para coletarem dados das regiões encontradas. Nesse caso, sensores deveriam enviar seus dados para regiões onde os *data-mules* pudessem ser encontrados. Uma vez que o *data-mule* possa visitar(ou amostrar) o campo-alvo de maneira uniforme, pode-se explorar a característica do passeio determinístico *"aleatório"* da DSC. Ou seja, um nó de sensor pode levantar sua probabilidade de reunir dados através de *data mules*, enviando sua informação sobre a DSC.

4. Naor et al., (Naor & Wieder, 2007) sugeriu a distribuição de tarefas de processamento entre múltiplos processadores. Eles usaram uma *hash table* distribuída que cria ligações entre diferentes processos. Quando um processador fica com a sua CPU congestionada, tarefas de processamento podem ser transferidas de um processador para outro, baseado nestas ligações. Suas *hash tables* foram desenhadas para intervalo 1-D, onde os processos eram os "Home-Nodes" dos subintervalos. Entretanto, se pensarmos neste problema em 2-D, nossa DSC pode oferecer uma *hash-table* distribuída 2-D que atende estas necessidades.

10.2 Trabalho futuro

Através deste trabalho, e especialmente no Capítulo de Simulações (Capítulo 8) demonstrou-se observações muito interessantes tanto sobre este novo protocolo como para o novo passeio determinístico "aleatório". Estas observações baseiam-se em diversas simulações, nas quais foram variadas a grande maioria dos parâmetros da RSSF, para poder estudar o problema em diferentes níveis. Nesta seção, apresentam-se as idéias que podem ser desenvolvidas em trabalhos futuros.

1. Neste trabalho, lida-se com as RSSF homogêneas, onde todos os sensores possuem a mesma capacidade de recursos. Como anteriormente mencionado, caso se saiba quais regiões são os *hot-spots* do cenário, que geram muito mais atividades que as demais, poderia se implantar ali sensores com mais capacidade de armazenamento. Entretanto, pode se pensar sobre esta questão de forma um pouco diferente. Suponha que se tenha implantado uma RSSF heterogênea (onde alguns sensores tenham mais recursos que outros). Suponha que aqueles sensores com maior capacidade de armazenamento sejam implantados uniformemente e que o campo-alvo possua regiões *hot-spots*. Assim, o problema será como transmitir as atividades de sobrecarga de regiões *hot-spots* aos sensores com maior capacidade de armazenamento. Desde que estes sensores tenham sido implantados uniformemente sobre um campo-alvo, pode-se facilmente usar a DSC para distribuir os dados uniformemente sobre este campo-alvo. Desta

forma, as cadeias teriam uma probabilidade muito alta de descobrir os sensores com maiores capacidades de armazenamento.

Sugere-se aqui que se explore a implicação de RSSF heterogêneas e os desempenhos do protocolo sugerido aqui. Sugere-se também, que se explore a proporção entre os números dos sensores com alta capacidade e os números sensores regulares, para que se encontre a combinação ideal que otimize as desempenhos das RSSF em termos de tempo de vida.

2. Neste trabalho, repetiu-se e demonstrou-se , através de diferentes tipos de simulações, que nossa DSC pode ser usada eficientemente para criar interações entre sensores. Em outras palavras, a DSC pode ser usada com eficácia como um passeio aleatório sobre um campo-alvo. Além disso, mostrou-se que estas cadeias selecionam uniformemente as localizações geográficas (ou *Home-Nodes*) em $[0, 1)^2$.

   No Teorema 1 faz-se referência ao trabalho de Gabber e Galil (Gabber & Galil, 1979) que justifica o uso da sua transformada para a rede estática que foi sugerida na Seção 4.5. Entretanto, não se pode dar uma prova matemática de que a DSC, a qual seleciona das duas transformadas alternativamente, realmente selecione pontos em $[0, 1)^2$ de modo uniforme. No Apêndice C formulou-se esse problema, para poder prová-lo.

3. Liao et al., (Liao & Wu, 2008) sugeriu um mecanismo de limiar múltiplo (*multiple threshold mechanism*) para armazenamento em RSSF. Eles definiram T_L níveis de limiares de armazenamento para todos os sensores da RSSF. De acordo com este modelo, a capacidade de armazenamento de todos os nós de sensores são divididos em T_L sub-capacidades. Assim, a RSSF sincroniza os níveis dos limiares que são usados naquele momento. Um sensor que tenha tido seu limiar de armazenamento regular esgotado, "direcionaria" seus eventos para seu vizinho mais próximo que tenha capacidade disponível de armazenamento (dentro do limiar atual). Em outras palavras, de acordo com este modelo, apesar dos sensores ainda terem espaço de armazenamentos disponíveis, eles o utilizariam apenas se a RSSF decidisse alcançar o próximo limiar.

   Acredita-se que os mecanismos de limiares múltiplos podem ser úteis no protocolo sugerido aqui. Ou seja, a RSSF não forçaria os sensores a esgotaram suas capacidades totalmente. Acredita-se que podem haver muitos trabalhos futuros sobre este tema de multi-limiares. Por exemplo, seja LSC a capacidade local quando $LSC = 20\%$ e supondo-se que um sensor tenha tido seu limiar atual esgotado apesar de ter utilizado somente seu $LSC = 20\%$ (da então atual capacidade limiar). Pode-se afirmar que este sensor não era um sensor *hot-spot*. Além disso, este sensor pode determinar localmente que ele não é um *hot-spot*. Desta forma e baseado nesta informação, o sensor pode usar o limiar seguinte em vez de construir novos enlaces de

DSC. Similarmente, pode-se deduzir que um sensor que tenha consumido completamente o seu limiar e que não tenha direcionado sua sobrecarga para outro sensor, seja um sensor *hot-spot*. Acredita-se também que o número de limiares e seus tamanhos (por exemplo, eles podem dividir sua capacidade de armazenamento local para sub-capacidades que não são iguais) são questões interessantes de modelagem.

4. Uma questão, repetida ao longo deste trabalho, é que os nós de sensores devem conhecer suas próprias localizações. também foi discutido este assunto e mencionado que a localização pode ser obtida por um GPS ou por algums técnicas de localização (Seção 4.4 e (Khan et al., 2009)). Neste trabalho, as três razões principais pela quais a localização é importante são:

   (a) Os eventos estão associados os locais que onde foram observados;

   (b) O protocolo GPSR (Karp & Kung, 2000) utiliza os locais dos sensores para enviar os pacotes aos seus destinos;

   (c) DSC usa as localizações dos sensores para calcular seus enlaces (Definição 7).

   Como o consumo de energia do GPS é muito alto, recomenda-se reduzir o seu uso em aplicações de RSSF. Um solução é usar triangulações entre os nós de sensores (por exemplo triangulação de Delaunay (Berg, 2000)) para estimar suas localizações. Khan et al. (Khan et al., 2009) sugere uma técnica de localização para o quadrado unitário 2-D que requer apenas 3 sensores de âncora (*anchor sensors*). Estes sensores podem ser estações fixas, ou sensores de alta energia com dispositivo de GPS que sabem a sua localização exata. Além disso, estes sensores devem *âncorar* a RSSF de uma forma que todos os outros sensores estejam na envoltória convexa (*convex hull*) destes três.

   Dependendo da aplicação da RSSF, se o local dos eventos observados não for importante, pode-se usar coordenadas virtuais. Em outras palavras, o protocolo DSC pode ser implementado de forma totalmente independente do GPS. Deste modo, sensores âncoras podem ser definidos por um protocolo distribuído. Estas âncoras vão definir o quadrado unitário (o que é essencial para o operador de módulo 1), e, com base nas triangulações, cada sensor pode definir a distância entre ele e os sensores âncora. Finalmente, estas distâncias podem ser convertidas para coordenadas virtuais dos sensores.

References

Ajay Jangra, R., Swati, & Priyanka. (2010, December). Wireless sensor network (wsn): Architectural design issues and challenges. *International Journal on Computer Science and Engineering(IJCSE)*, *2*, 3089-3094. Available from http://www.enggjournals.com/ijcse

Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002, aug). A survey on sensor networks. *Communications Magazine, IEEE*, *40*(8), 102 - 114.

Alon, N., Avin, C., Koucky, M., Kozma, G., Lotker, Z., & Tuttle, M. R. (2008). Many random walks are faster than one. In *Proceedings of the twentieth annual symposium on parallelism in algorithms and architectures* (pp. 119–128). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1378533.1378557

Aly, S., Kong, Z., & Soljanin, E. (2008, april). Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In *Information processing in sensor networks, 2008. ipsn '08. international conference on* (p. 171 -182).

Andreas, V. H., H, V., Köpke, A., Karl, H., Wolisz, A., & Berlin, T. U. (n.d.). *A common wireless sensor network architecture?*

Berg, M. (2000). *Computational geometry: algorithms and applications*. Springer. Available from http://books.google.com/books?id=C8zaAWuOIOcC

Bermeo, F. (2010). *Technology blog @ONLINE*. Available from http://bermeofabian.blogspot.com/2010/07/smart-dust.html

Camtepe, S., Yener, B., & Yung, M. (2006, june). Expander graph based key distribution mechanisms in wireless sensor networks. In *Communications, 2006. icc '06. ieee international conference on* (Vol. 5, p. 2262 -2267).

Clark, B. N., Colbourn, C. J., & Johnson, D. S. (1991, January). Unit disk graphs. *Discrete Math.*, *86*, 165–177. Available from http://portal.acm.org/citation.cfm?id=108059.108074

Culler, D., Estrin, D., & Srivastava, M. (2004, aug.). Guest editors' introduction: Overview of sensor networks. *Computer*, *37*(8), 41 - 49.

Diestel, R. (2006). *Graph theory*. Springer. Available from http://books.google.com/books?id=aR2TMYQr2CMC

Dietrich, I., & Dressler, F. (2009, February). On the lifetime of wireless sensor networks. *ACM Trans. Sen. Netw.*, *5*, 5:1–5:39. Available from http://doi.acm.org/10.1145/1464420.1464425

Dimakis, A. G., Sarwate, A. D., & Wainwright, M. J. (2006). Geographic gossip: efficient aggregation for sensor networks. In *Proceedings of the 5th international conference on information processing in sensor networks* (pp. 69–76). New York, NY, USA: ACM. Available from

http://doi.acm.org/10.1145/1127777.1127791

Estrin, D., Govindan, R., & Heidemann, J. (1999). Next century challenges: Scalable coordination in sensor networks. In (pp. 263–270).

Gabber, O., & Galil, Z. (1979). Explicit constructions of linear size superconcentrators. In *Proceedings of the 20th annual symposium on foundations of computer science* (pp. 364–370). Washington, DC, USA: IEEE Computer Society. Available from http://portal.acm.org/citation.cfm?id=1398508.1382635

Gkantsidis, C., Mihail, M., & Saberi, A. (2006, March). Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.*, *63*, 241–263. Available from http://portal.acm.org/citation.cfm?id=1141193.1141199

Greenberg, A., & Goodman, J. (1993, jan). Sharp approximate models of deflection routing in mesh networks. In (Vol. 41, p. 210 -223).

Heinzelman, W. R., Ch, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In (pp. 3005–3014).

Hoory, S., Linial, N., Wigderson, A., & Overview, A. (2006). Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S*, *43*, 439–561.

Hung-Yu, S., Gwo-Jong, Y., & Jang-Ping, S. (2005). Energy hole healing protocol for surveillance sensor networks. In *In wasn workshop on wireless, ad hoc, and sensor networks.* in WASN Workshop on Wireless, Ad Hoc, and Sensor Networks.

Jardak, C., Riihijärvi, J., & Mähönen, P. (2010). Extremely large-scale sensing applications for planetary wsns. In *Proceedings of the 2nd acm international workshop on hot topics in planet-scale measurement* (pp. 3:1–3:6). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1834616.1834621

Kahn, J. M., Katz, R. H., & Pister, K. S. J. (1999). Next century challenges: mobile networking for "smart dust". In *Proceedings of the 5th annual acm/ieee international conference on mobile computing and networking* (pp. 271–278). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/313451.313558

Karp, B., & Kung, H. T. (2000). Gpsr: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on mobile computing and networking* (pp. 243–254). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/345910.345953

Khan, U., Kar, S., & Moura, J. (2009, may). Distributed sensor localization in random environments using minimal number of anchor nodes. *Signal Processing, IEEE Transactions on*, *57*(5), 2000 -2016.

Lewis, F. L. (2005). Wireless sensor networks. In *Smart environments* (pp. 11–46). John Wiley &

Sons, Inc. Available from http://dx.doi.org/10.1002/047168659X.ch2

Liao, W.-H., & Wu, W.-C. (2008, June). Effective hotspot storage management schemes in wireless sensor networks. *Comput. Commun.*, *31*, 2131–2141. Available from http://portal.acm.org/citation.cfm?id=1379906.1380029

Luo, L., Huang, C., Abdelzaher, T., & Stankovic, J. (2007, may). Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Infocom 2007. 26th ieee international conference on computer communications. ieee* (p. 1802 -1810).

Michael Winkler, K. H., Klaus-Dieter Tuchs, & Barclay, G. (2008). Theoretical and practical aspects of military wireless sensor networks. *Journal of Telecommunications and Information Technology (JTIT)*, *2*, 37-75. Available from http://www.nit.eu/publications/journal-jtit

Naor, M., & Wieder, U. (2007, August). Novel architectures for p2p applications: The continuous-discrete approach. *ACM Trans. Algorithms*, *3*. Available from http://doi.acm.org/10.1145/1273340.1273350

Papoulis, A., & Pillai, S. (2002). *Probability, random variables, and stochastic processes*. McGraw-Hill. Available from http://books.google.com/books?id=YYwQAQAAIAAJ

Pister, K. S. (1999, June). *Smart dust project @ONLINE*. Available from http://robotics.eecs.berkeley.edu/~pister

Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., et al. (2003, August). Data-centric storage in sensornets with ght, a geographic hash table. *Mob. Netw. Appl.*, *8*, 427–442. Available from http://dx.doi.org/10.1023/A:1024591915518

Sheng, B., Li, Q., & Mao, W. (2006). Data storage placement in sensor networks. In *Proceedings of the 7th acm international symposium on mobile ad hoc networking and computing* (pp. 344–355). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/1132905.1132943

Silberstein, A., & Yang, J. (2007, april). Many-to-many aggregation for sensor networks. In *Data engineering, 2007. icde 2007. ieee 23rd international conference on* (p. 986 -995).

Wu, X., Chen, G., & Das, S. K. (2008, May). Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Trans. Parallel Distrib. Syst.*, *19*, 710–720. Available from http://portal.acm.org/citation.cfm?id=1399097.1399351

Appendix A

Extension for Our Results

Throughout this work we demonstrate various performance results from different simulations that we have conducted. Since we have varied multiple parameters ($n, p, \gamma, HS, |S_0|$, etc.) there are obviously a lot of simulation results. In this Appendix, we demonstrate some simulation results that could not fit inside the text. We begin with the results for the comparison between sensors region sizes and the number of static links that "selects" them (Section 4.5). We demonstrate this relation for various network sizes, and show that the same behavior repeats for different n (WSN size) values.

Figures A.2 - A.5 demonstrate the deterministic "random " walk that we have presented in Section 5.5 for WSN of sizes $50 - 6400$ sensors with a varied size of primary group S_0. These results are the main impacts of our work since they demonstrate the behavior of our DSC algorithm are as efficient as a random selection of geographical locations in $[0, 1)^2$, or as a random walk over the target field.

The rest of the figures in this section refer to the simulations consucted in Chapter 8. Figure A.6 demonstrate that the characteristics of the LSC curves maintain for different sizes of HS sets and different p values. Moreover, we demonstrate how the optimal LSC value changes almost together for all the curves as we modify the p parameter. On Figure A.7 we show only the optimal points form Figure A.6 and compare them with the mean and variance values. This figure extend the simulation results from Figure 8.6 for different HS sizes. Figure 8.12 extend the simulation results from Section 8.8, for the two valley deployments (Figure 8.12). Figure A.9 demonstrate the drooped packets analysis for more p values (Section 8.5). Figures A.11 and A.10 demonstrate the simulation results for varied sizes of HS sets from Section 8.7.

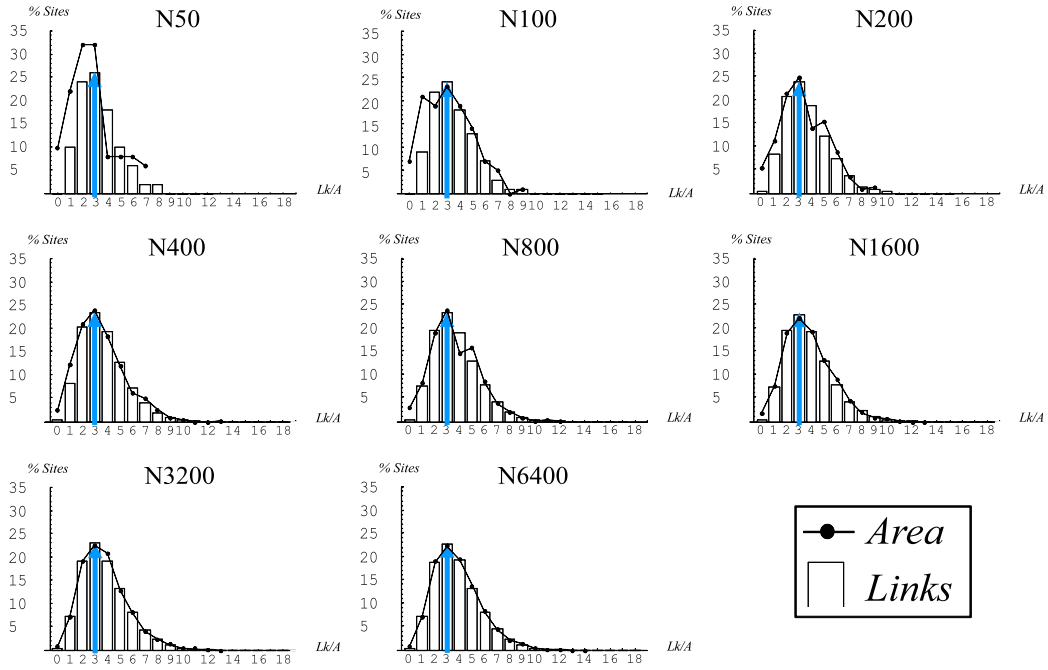


Fig. A.1: Area Vs. Number of Links for various network sizes. Note that for all n (WSN size) the number of links is distributed over the mean of 3 links.

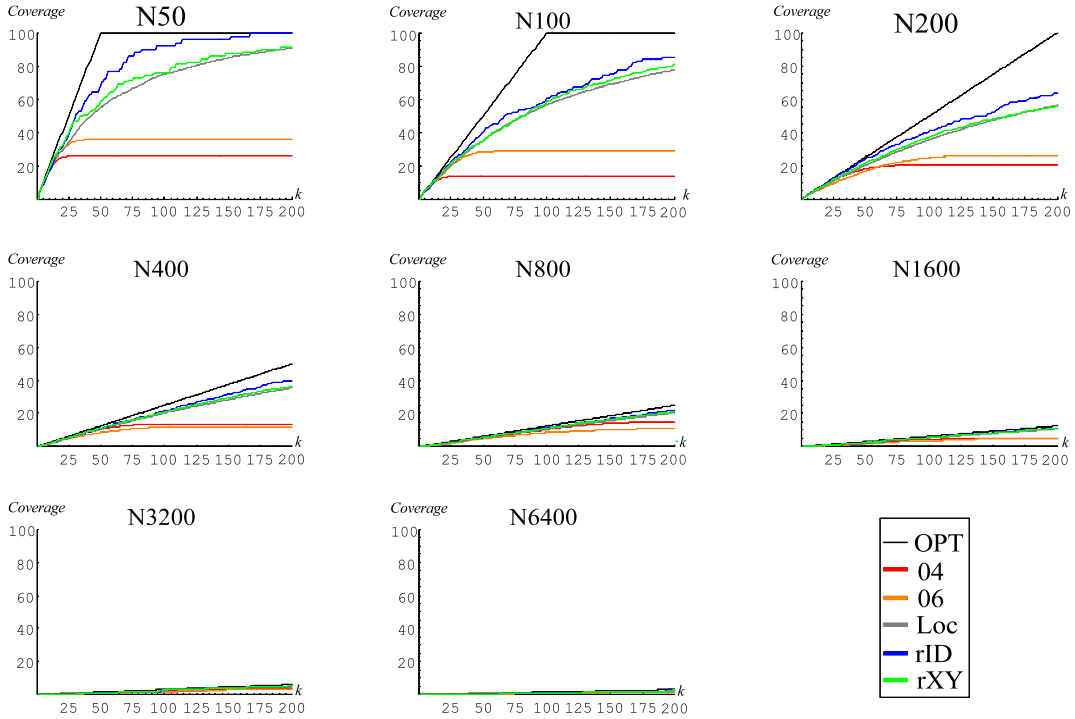


Fig. A.2: Sets simulation for 1 sensor primary group.

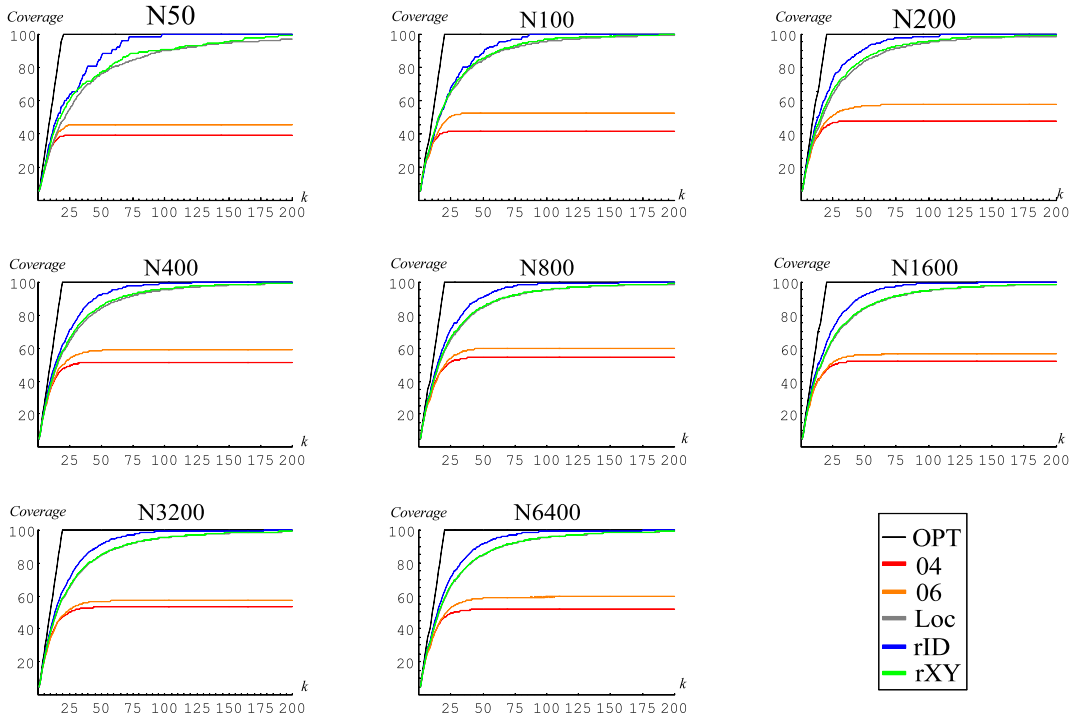


Fig. A.3: Sets simulation for Log(n) primary group.

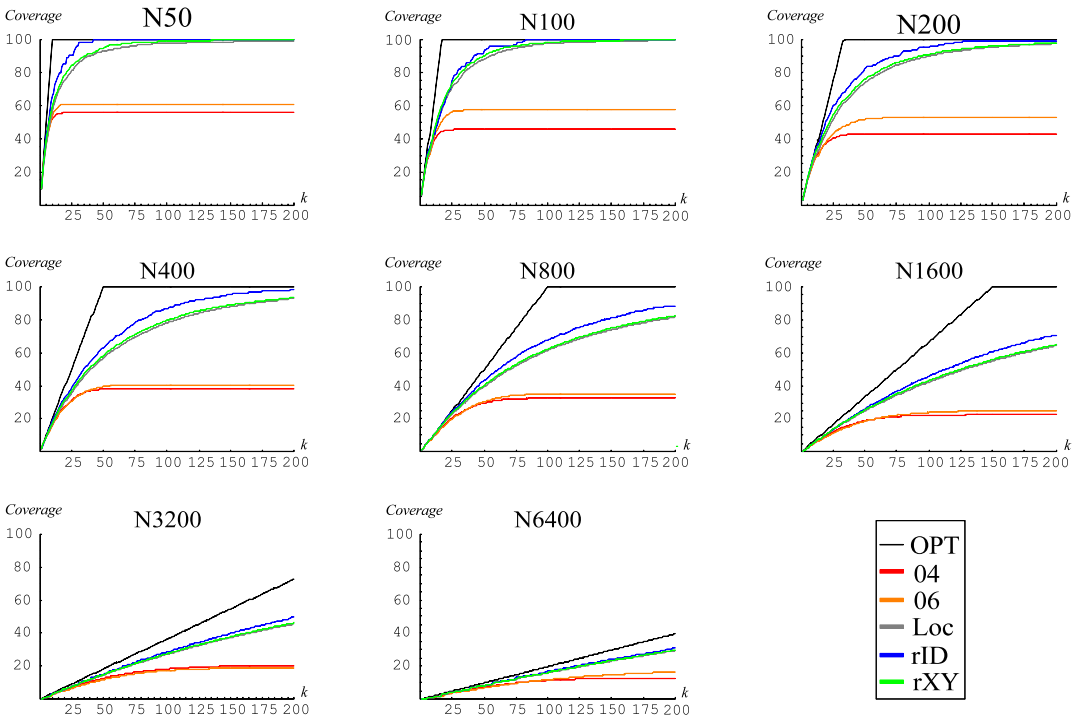


Fig. A.4: Sets simulation for 5% sensors primary group.

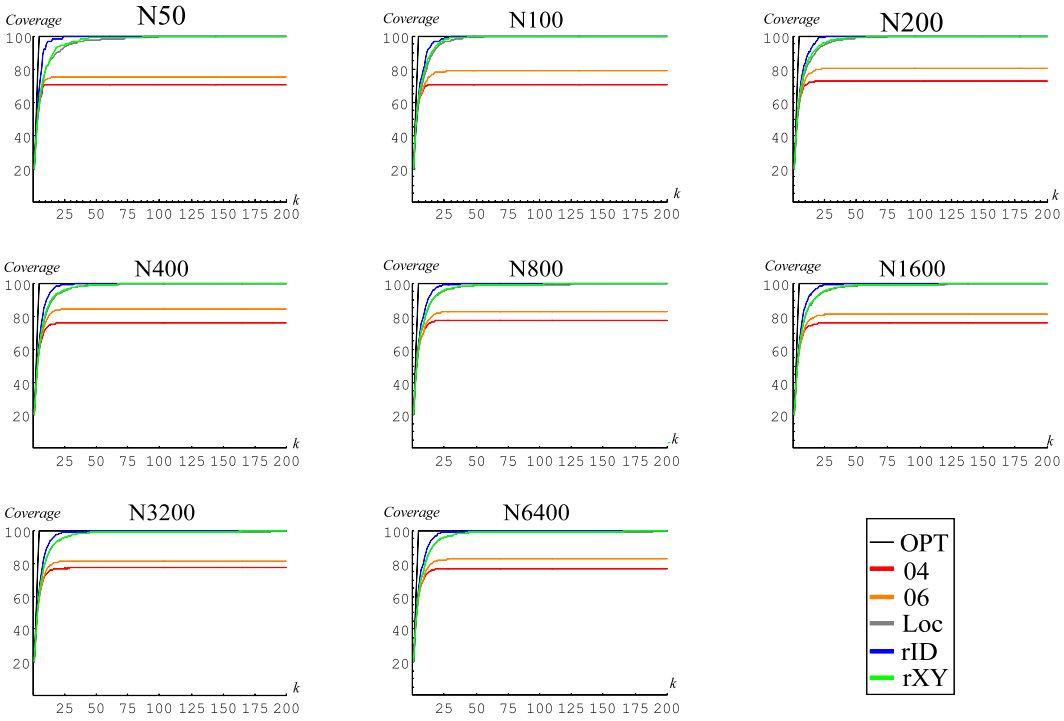


Fig. A.5: Sets simulation for 20% sensors primary group.

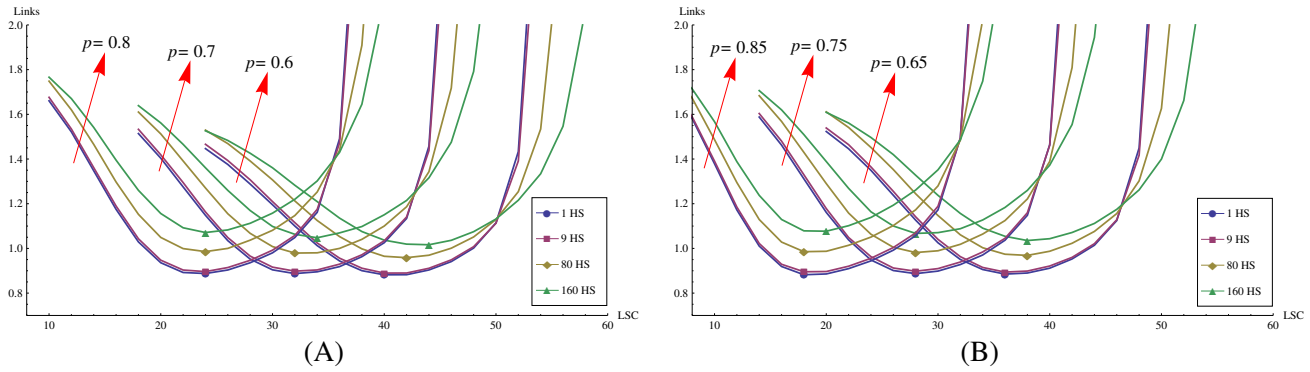
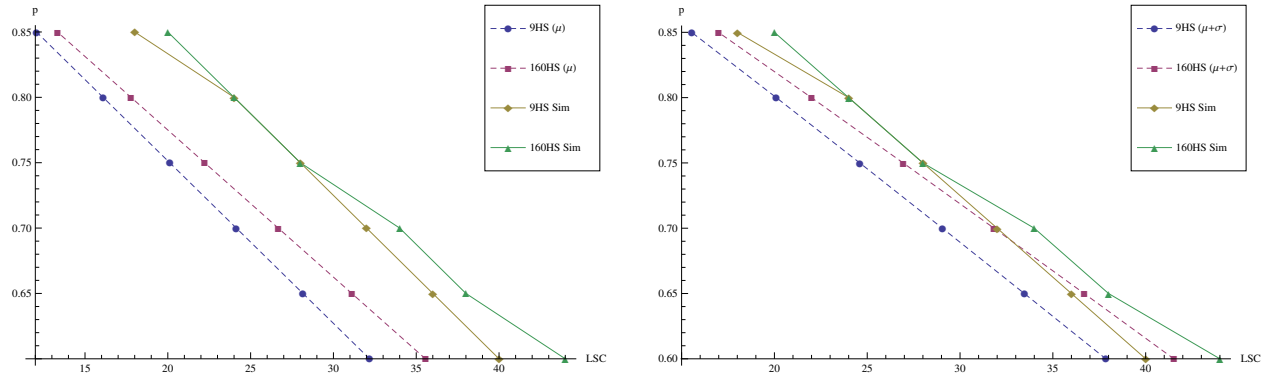


Fig. A.6: RETRIEVE costs for different p values. $n = 1600$ sensor nodes.



(A) Mean only (B) Mean + Variance

Fig. A.7: Calculating optimal LSC with and without Variance.

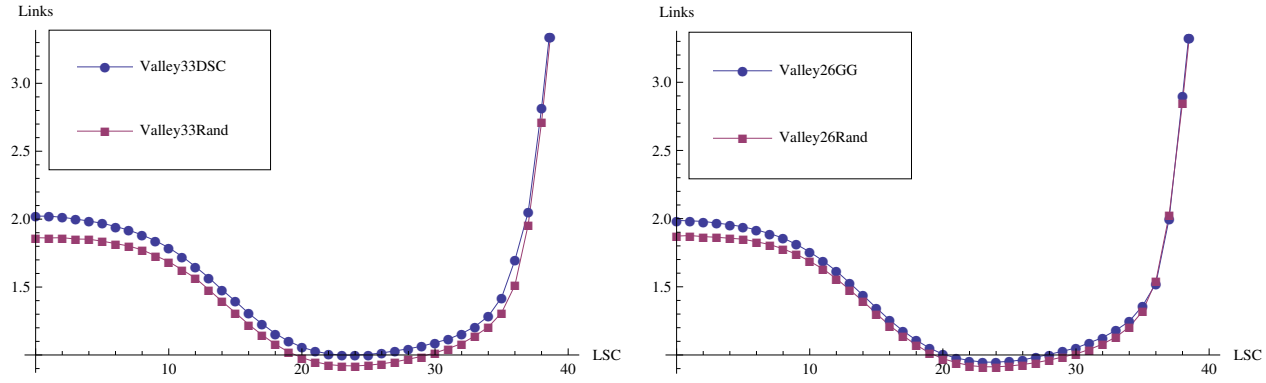
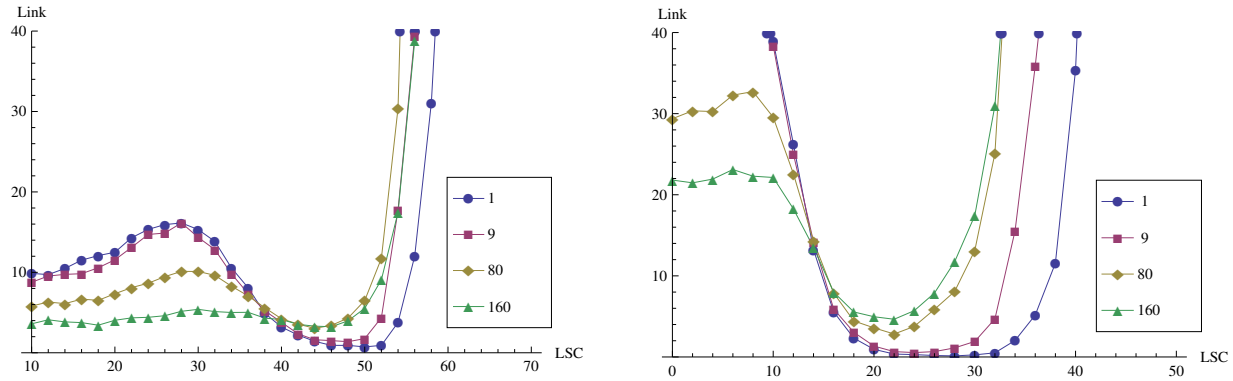


Fig. A.8: Simulation results for the valleys deployments of hot-spot sensors (Figure 8.12).



(A) $p = 0.6$ (B) $p = 0.85$

Fig. A.9: Number of dropped packets Vs. LSC (800 sensor nodes, $\gamma = 0.8$).

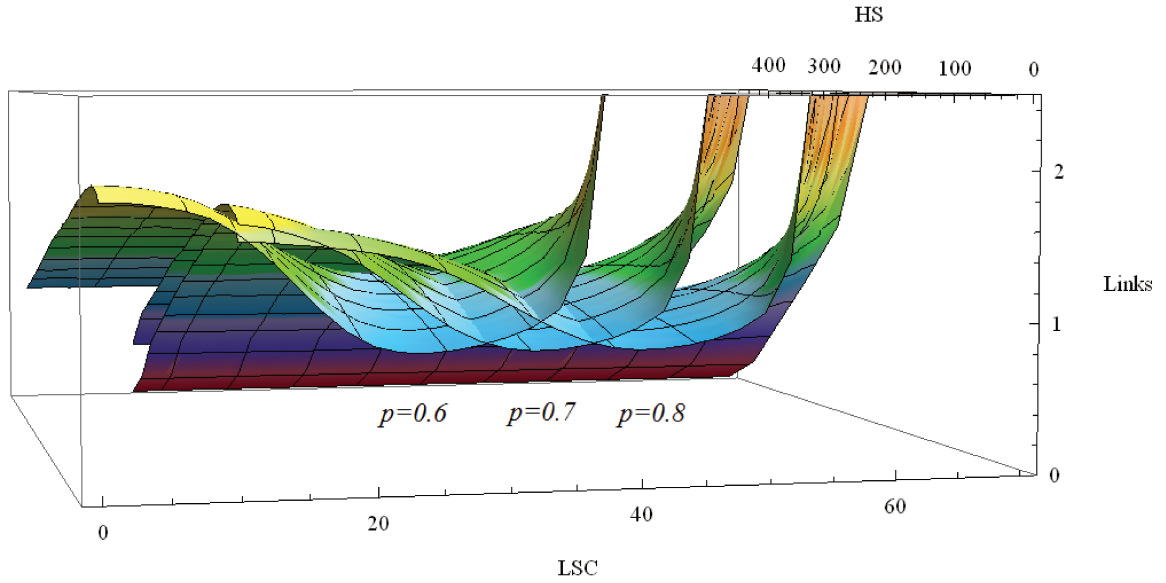


Fig. A.10: Varying the number of hot-spots for 3 Different p values (angle 1).

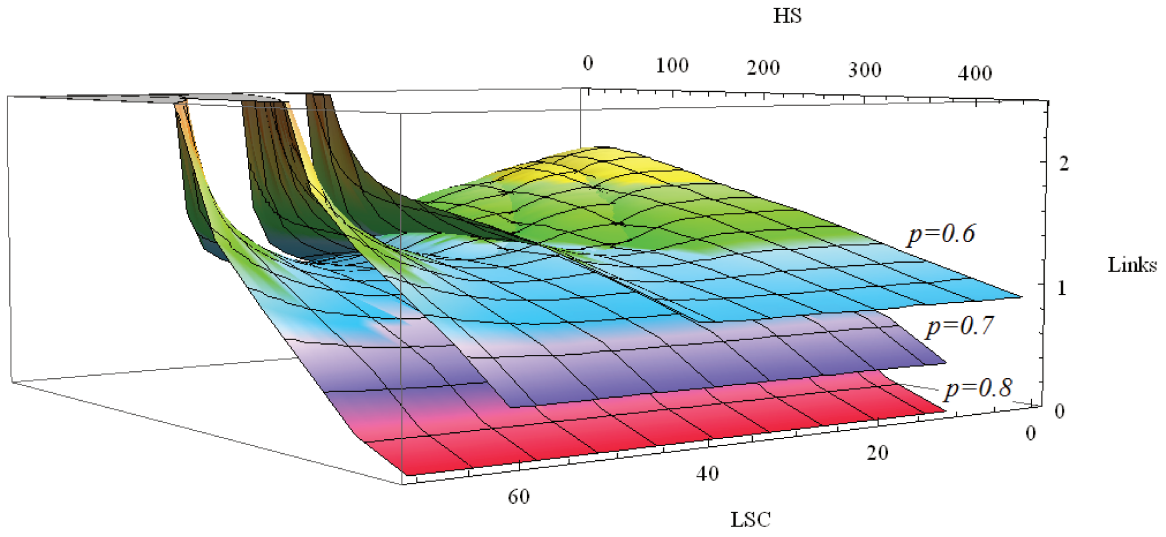


Fig. A.11: Varying the number of hot-spots for 3 Different p values (angle 2).

Appendix B

WSN Simulator

Although there are many published researches in the field of WSN, there is still a lot of work ahead. Winkler et al. (Michael Winkler & Barclay, 2008), describes the 3GSN (see Section 3.2.3) as immature systems that are still at the proof-of-concept stage. Moreover, Handziski et al. (Andreas et al., n.d.), titles the WSN research as heavily fragmented. "It seems that independent developments take place on both sides of the Atlantic without using common standards". Thus, there is not yet a common *"lingua franca"* as in some other fields of networking research. Mostly, the absent in common standards is mostly reflected in the field of simulations. This observation is very important, since almost all works in the field of WSN are based on simulations.

Different research groups use different simulation environments for performance evaluation. The most popular simulation environments in the literature include (Andreas et al., n.d.): NS-2, Opnet, GlomoSim, Qualnet and Om-NET++, etc. However, none of these simulation tools satisfy all WSN demands. For example, the wireless channel model is too simplistic and not easily changed (NS-2), not all relevant protocols are easily available (Omnet) and some are commercial (Opnet, Qualnet). Moreover, as we described on Chapter 6, most WSN research assume that events are uniformly generated in target area. Respectively, the most important problem is that there is no simulation tool that implements a convincing model for sensor excitations. Therefore, since the core of our work is based on non-uniform traffic model, we had to implement an environment that can supports such kind of traffic. In other words, our simulator implements a sensor excitation model that is based on non-uniform event generation on the target area.

In terms of running time, the bottleneck for our simulations is in the point location algorithm. In other words, when a sensor wishes to "store forward" its events, it direct them to a geographical address that is obtained by the DSC. In order to determine the specific sensor that receives those events, our simulator should retrieve the Home-Node for that specific geographical address. Since almost 80% of the events are "stored forward" this problem is of a large scale.

A naive algorithm that solves the point location problem can be comparing the distances of the geographical address with all the sensors in S. That algorithm would have the cost of $O(n)$, where n is the number of sensors in the WSN. If $p \cdot Mn$ events are to be stored forward, the simulator running time has a $O(n^2)$ complexity. That factor makes computation time very expensive. Therefore, our major challenge was in implementing a heavy computational geometry data structure that supports the point location search (see the slabs data structure in (Berg, 2000), Chapter 6) in order to reduce the computational complexity. The data structure that we used, reduced the simulator complexity from $O(n^2)$ to $O(nLog(n))$.

B.1 Development

We have implemented our simulator over two different platforms. Our first code was written in C++ and contained a full implementation of the slabs data structure ((Berg, 2000), Chapter 6). The slabs structure reduces the cost for every point location search to $O(Log(n))$. Therefore we achieved a $O(nLog(n))$ running time. Notice that the open source Computational Geometry Algorithms Library (CGAL, http://www.cgal.org) contains the Doubly-Connected Edge List (DCEL) data structure that implements the very same slabs structure. However, at the time we did not know about the existence of that library. Our C++ code was very large and complex, it contained more than 5000 lines of code. Therefore, it was difficult for modifications and debuging. However our code was independent and implemented the whole simulator without relying on existing codes.

Wolfram mathematica has a Kd-Tree (Berg, 2000) data stricture that also implements the point location problem for $(Log(n))$ complexity. With Wolfram Mathematica we have managed to reduce the lines of code from 5000 to 700 while implementing more options. Moreover, the advantage of implementing the simulation code in Wolfram Mathematica is that it allows programming, plotting and script writing with the same kernel. Therefore, the develop time was reduced drastically. In the next section we will give a general description of the simulator, we will demonstrate how to run simulations and how to modify the code.

B.2 DSC Simulator V1.3

The latest version of our Mathematica simulator is V1.3. The folder DSC V1x3 is our main simulations folder. It contains two Mathematica (*.nb) notebook files, a data file (*.m) with specific simulation configurations and the NBD folder. The NBD folder contains six library files that implement all the functions that are used by our simulator. The main notebook file, “simulation.nb”, begins by loading all these libraries. It loads the parameters from the “config.m” file and executes the

simulation.

Apart of the simulator folder, we maintain different WSN topologies of different sizes on the DataBase folder. We use that folder for simulations I O. Our simulator reads sensors locations from the desired topology folder and writes the results back to that folder.

B.2.1 Configurations

The specific configurations for a simulation are defined in the "config.m" file. We can design a new simulation, simply by modifying the parameters in this file. We use the "config.nb" file in order to change these parameters manually and write them into "config.m". In Table B.1 we describe all the parameters that can be changed for simulations.

Parameter	Explenation
Nx	Network Size. The simulator will access this network size in the DataBase and execute the same simulation for every network topology in that folder
$HsList$	Number of hot-spots to select uniformly. Notice that this parameter is a vector. The simulator will perform new simulation for each entery of that vector.
p	hot-spot probability
$LocalStorageVector$	Number of LSC units Notice that this parameter is a vector. The simulator will perform new simulation for each entery of that vector.
$sampleRpeatMax$	Number of iterations that each simulation should be repeated
γ	Maximum storage consumption (of the entire WSN)
$SimulationName$	Result files will begin with this prefix.
$SimulationPurpose$	The header of the results will contain this string. This string will explain what we tried to simulate in that specific simulation.

Tab. B.1: Simulation parameters, stored in the "config.m" file.

In Section B.2.2 we will demonstrate how to design a set of simulations with different parameters that can be executed in parallel.

B.2.2 Parallel Simulations

The Wolfram Mathematica 7 can accelerate processes by using multiple cores. However, its ComputationalGeometry library (which implements the K-d tree) does not makes use of the multi-threading operations. In order to reduce simulation time we process several simulations in parallel.

In other words, we divide our main simulation into several "sub-simulations". We run each "sub-simulations" on different core.

For example, if we would like to conduct a simulation for some WSN topology and examine different LSC values where $LSC = 0{-}41$, we could divide that simulation to several "sub-simulations". If we wish to utilize six different cores in our machine, we will divide the main simulation to six different simulations, where the LSC parameter for these simulations varies: $[0, 7)$, $[7, 14)$, $[14, 21)$, $[21, 28)$, $[28, 35)$, $[35$ On the Scripts folder, Scripts $2-3$ makes it easier to design a simulation and divide it to several cores automatically. Actually, these scripts create a simulation folder for each core we wish to utilize and write the specific parameters to "config.m" (instead of writing them manually). In other words, we create 6 different simulation folders, each with a different LSC vector. Unfortunately, in order to run these folders simultaneously, one has to open a Mathematica session (manually) for each folder and execute its "Simulation.nb" file. We did not managed create a script (in Mathematica or any other language) that can execute these "Simulation.nb" files automatically. Scripts $4-5$ makes it easier to collect all the results into the same I O folder.

Appendix C

DSC - Analytical Formulation

In Chapter 5 we discussed the possibility of alternatively using the Gabber-Galil transform to generate a DSC sequence (Defenition 7) that can be used as a "search-function". In section 5.5 we simulated this sequence and demonstrated that it has the same characteristics as a random walk, or as a random selection of Home-Nodes. In Chapter 8 we have demonstrated the motivation for applying this sequence within our protocol and evaluated its performances via simulations. However, throughout this work we could not analytically prove that our sequence simply select points in $[0,1)^2$ uniformly. In this appendix we give our preliminary work for showing that our sequence is indeed a deterministic "random" walk.

C.1 Objective

Our objective in this section is to define the characteristics of a deterministic "random" walk as a part of a sketch for the analytical analysis of DSC.

C.2 Definitions

C.2.1 General

Definition C.1. Unit Square. We define the unit square as the surface $I = [0,1) \times [0,1)$. Therefore, we work above all points (x,y) in I. In other words, all points (x,y) that satisfy: $0 \leq x,y < 1$

Definition C.2. Planar Voronoi diagram. Let S be a set $S = s_1, s_2, s_3, \ldots, s_n$ of $n \geq 2$ distinct points ($s_i = (x,y)$) that represent sensor locations on the unit square. We say that the planar Voronoi diagram for these sensors is a partition of I into n non-overlapping regions. Each point in

I is associated with its closest member from S. That is, if a point p is located in the region of some sensor $s_i \in S$, it has to satisfy $Min(|s_i - p|)$ among all points $s_i \in S$. We ignore the measure zero set of points that falls on the border between two Voronoi cells.

Corollary C.1. Home-Node. We say that a sensor s_i is a Home-Node for some point $p \in [0,1)^2$ iff that point is included inside the Voronoi region of sensor s_i $p \in \mathrm{Vor}(s_i)$. In other words, among all generators s_j, the Euclidean minimum $Min(|s_j - p|)$ is achieved only for sensor s_i.

Definition C.3. Link. A link (p_i, p_j) is an attribute that represent communication and which associate point p_i with point p_j. Our links are directed links i.e., by noting (p_i, p_j) we mean that the link's direction is from p_i to p_j. Or in other words, communication flows from p_i to p_j.

Definition C.4. Chain. A chain C of size $k+1$ is an ordered set of points $C = (p_0, p_1, p_2, \ldots, p_k)$, where exists a link between every two consecutive points. By definition, $p_0 \in S$ must be a sensor. We call that sensor p_0 a generator, since it literally generates the chain. The discretization of C is an ordered set of sensors which is based on the Home-Nodes of each point p_i.

C.2.2 Specific Definitions

Definition C.5. Distributed Storage Chain (DSC). DSC is a chain $\mathcal{C}_k(s)$ (Definition C4) that is used by some sensor s for storage allocation. Generally, $\mathcal{C}_k(s)$ is an ordered set of points that visits sensors in S for storage allocation purpose. $\mathcal{C}_k(s)$ is a deterministic chain of length $k+1$, which is initiated by some sensor $s = p_0 = (x_s, y_s)$. We define the sequence of vertices $\mathcal{C}_k(s) = (p_0, p_1, p_2, \ldots, p_k)$ according to the following recursive rule:

$$s = p_0 \text{ and for } 1 \leq i \leq k,\ p_i = CW^i(p_{i-1})$$

where the i^{th} link (g^{i-1}, g^i) is given by::

$$CW^i(x,y) = \begin{cases} East : \{x+y, y\} \pmod 1 & \textit{for i even} \\ North : \{x, x+y\} \pmod 1 & \textit{for i odd} \end{cases}$$

Notice that the east and west links are the same links as represented by the Gabber-Galil expander graph (Gabber & Galil, 1979).

Observation. Unit tours. In Definition C.1. we specified that we work above all points (x, y) in I. Actually, according to the $(\mathrm{mod}\ 1)$ operator, we work above the unit torus. Note that links that exceed the unit square's margins, spins from the other side.

Definition C.6. Uniform sampling. We say that a chain forms a uniform sampling of I if it obeys both the characteristics of spatial and temporal uniform distributions:

- Spatial uniform distribution. For every chain large enough ($n \to \infty$) and any area A bounded by any type of polygon, $A \subseteq [0,1)^2$, the number of point from the chain inside A is proportional to A. In other words, if there are x points within A, we expect to find $(1-A) \cdot x$ points out of that polygon.

- Temporal uniform distribution. For every interval k and a partial chain $\mathcal{C}' = (p_i, p_{i+1}, p_{i+2}, \cdots, p_{i+k})$ within a given chain, we obtain a spatial uniform distribution.

Definition C.7. Smoothness. Smooth. We apply the one dimensional measure smooth (Naor & Wieder, 2007) for two dimensions. The smoothness of S is denoted by $\rho(S)$ and is defined to be:

$$\forall i,j \quad max|\frac{Vor(s_i)}{Vor(s_j)}|$$

where $Vor(s)$ represents the total area of the region of sensor s. If it is guaranteed that the smoothness of S is bounded by some constant, independent of n, we say that S is smooth.

Definition C.8. Accurate Points. Accurate points are points with coordinates which are long enough (many digits after the decimal point). In order to define the term "long enough" we use one of the following two definitions:

1. $p = (x,y): \quad x,y \in$ *Set of Irrational numbers*

2. $p = (x,y): \quad x,y = \frac{m}{n}, \quad n \to \infty$

In other words accurate points use coordinates with a large number of digits after the decimal point. The problem with coordinates that contain a small number of digits after the decimal point is that they may cause repetitions. An example for a repetition: $(0.5,0.5) \to (0.5,0) \to (0.5,0) \to (0.5,0.5)$

C.3 Formulation

1. We begin with a set S of n sensors that are distributed uniformly over the two dimensional unit torus $[0,1) \times [0,1)$. We assume that sensors coordinates (x,y) are represented by accurate points (Definition C.8). Moreover, we assume that the distribution of region sizes is smooth.

2. We divide the unit torus into n regions according to the Voronoi diagram of sensors locations. Notice that every sensor $s \in S$ is Home-Node for all the points $p \in Vor(s)$ inside its Voronoi region.

3. We construct a DSC for some sensor s walk according to Definition C.5.by alternatively using the two Gabber-Galil transformations.

4. We would like to show that the sequence $DSC(s) = (p_0, p_1, p_2, \cdots, p_n)$ for some sensor s and some large length n is a uniform sampling of the unit torus. Notice that here we talk about the Voronoi diagram and the smoothness of a chain. The smoothness of DSC(s) is, therefore, denoted by $\rho(DSC(s))$ and is defined to be:

$$\forall i, j \quad max|\frac{Vor(p_i)}{Vor(p_j)}|$$