Self-Organization of Wireless Sensor Networks for Data-Centric Routing Using Bloom Filters

by

Peter Hebden

Submitted in total fulfilment of
the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Software Engineering
The University of Melbourne
Australia

June 26, 2009

The University of Melbourne

Australia

Abstract

Self-Organization of Wireless Sensor Networks for Data-Centric Routing Using Bloom Filters

by Peter Hebden

Wireless sensor networks are primarily composed of many resource constrained battery powered devices known as *motes*. In order to meet useful quality of service guarantees, more energy efficient protocols for self-organization and communication are needed. First, we present Distributed Asynchronous Clustering in the context of homogeneous networks and show that it generates well separated cluster heads and energy efficient clusters. Second, we present Distributed Asynchronous Clustering in the context of heterogeneous networks and use its local feedback mechanism, which we call *pre-emptive recruiting*, to generate networks with small-world properties. Third, given such a self-organized network, we present Bloom Gradient Routing, which uses Bloom filters to suppress wasteful radio transmissions. Experimental data, generated by our network simulator, indicates that when the wireless sensor network is organized as a small-world and augmented with Bloom filters, not only do short paths exist but the network is able to use them to service queries and, therefore, both latency and the energy consumed by radio communication are reduced, and network lifetime is increased to a far greater extent than would be possible using comparable protocols described in the literature.

This is to certify that

(i) the thesis comprises only my original work,

(ii) due acknowledgement has been made in the text to all other material used,

(iii) the thesis is less than 100,000 words in length, exclusive of table, maps, bibliographies, appendices and footnotes.

Signature_____

Date_____

Acknowledgments

I would like to thank the Australian Government, The University of Melbourne, and National ICT Australia for their financial support in the form of IPRS, MIRS, and NICTA Scholarships. In addition, I would also like to thank my Supervisors, Dr. Adrian Pearce and Dr. Lars Kulik, and the members of my PhD commitee, Associate Professor Chris Leckie and Associate Professor Raj Buyya.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) rely on large numbers of tiny sensor nodes to monitor physical environments. These devices, commonly known as *motes*, are substantially smaller than handheld devices such as mobile phones, or personal digital assistants (PDAs). In this thesis we use the terms *mote*, *sensor*, and *sensor node* interchangeably to refer to a generic "sensor node" that comes equipped with a processor, memory, radio, a power source, and one or more sensors selected from a large universe of sensor types. In most cases they are deployed close to their target, must operate autonomously over extended periods of time, and may provide periodic measurements of specific phenomena such as temperature, humidity, light, salinity, carbon monoxide gas, chlorine levels in a water system, machine vibration, and so on. They may also provide event notification to applications that monitor their environment for specific conditions such as intrusion, machine failure, or the presence harmful chemical or biological agents [2]. Sensor networks are a new paradigm for providing a wide variety of environmental monitoring services based on motes equipped with an application specific set of sensor types.

While numerous and easily deployed, individual low cost motes have very limited capabilities, and they must operate autonomously and without maintenance for as long as possible. Most are equipped with only a few sensor types [38]. They are very constrained in terms of processing, memory, radio bandwidth and, most critically, energy [30]. For most applications radio communication is by far the most energy intensive task. Not surprisingly, most research has focused on this aspect of sensor networks [113]. Indeed, *network lifetime* is the most important performance metric. We define network lifetime as the time during which an application specific percent of motes are operational and connected, and the network meets quality of service (QoS) require-

ments. Unfortunately, homogeneous networks built from motes do not scale – resulting in a network characterized by high latency, dropped packets, and node failure [32].

The fundamental problem addressed by this thesis is as follows: battery powered motes are the weakest link in sensor networks; and where many thousands of motes are deployed over a large geographic area, the network will fail to meet application QoS requirements unless energy efficient protocols for self-organization and communication are implemented. **What mechanisms should sensor networks use to generate a scalable and efficient topology, and what forms of in-network processing should they use to reduce consumption of scarce resources during query dissemination and, thereby, extend network lifetime?**

1.1 Large Scale Wireless Sensor Networks

This thesis will focus on very large-scale, widely-deployed, hierarchical networks where many thousands of resource constrained motes communicate, via a relatively small number of more powerful devices (*gateways* and *hubs*), with one or more sinks. The motes are assumed to be consistent with the vision of sensor networks expressed in [30]: very minimal, low cost, stationary devices designed to do little more than take measurements and transmit over short distances. Gateways and hubs provide services such as in-network processing, routing, and long range radio links. Each sink is an entry point for application specific queries and an exit point for sensor data.

Figure 1.1 shows a hypothetical wireless sensor network. The sinks, located at each corner, are stationary. Application nodes (laptops, PDAs, etc.) may be mobile but are assumed to remain stationary while querying the network. Network applications retrieve information from the sensor network via sinks. Users may be interested in specific information and invoke queries much as they would with a traditional SQL database. Application queries enter the network via sinks (network access points), request real-time information, rather than information about a stored collection of data, and must be routed efficiently by relay nodes to *a priori* unknown motes with data of interest. Matching data from many sources must flow back along multi-hop paths to

the application via a sink.

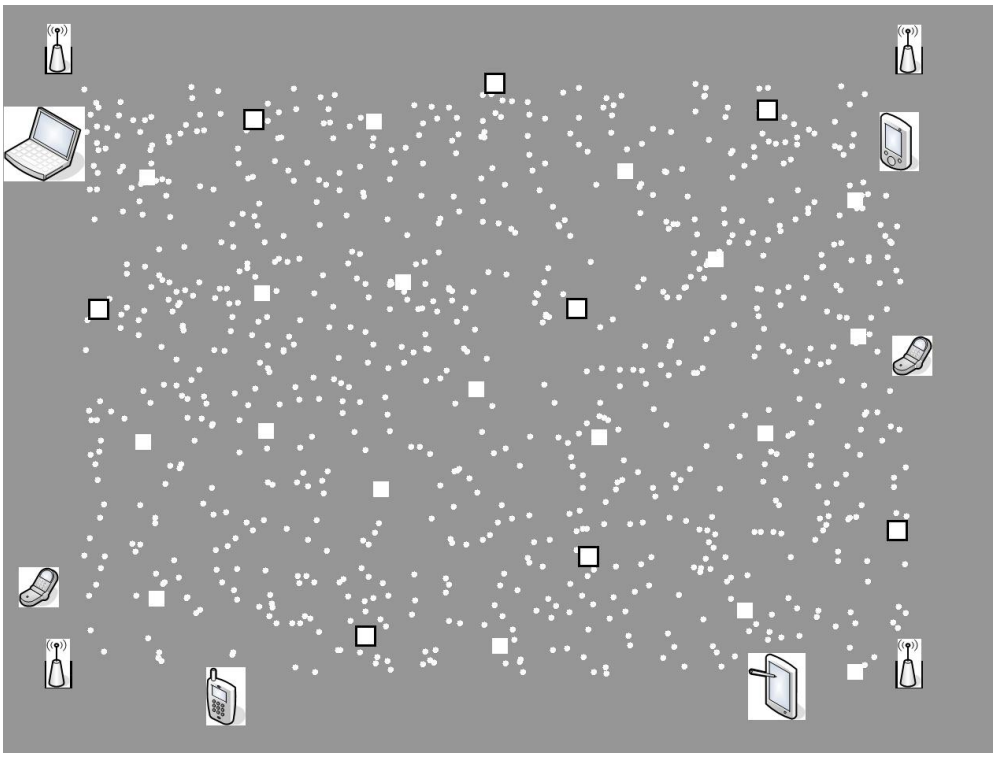


Figure 1.1: **WSN:** Sinks are represented by antennas at each corner. Mobile devices access the network via the sinks. Motes are small white dots, while relay nodes are white squares.

1.2 Design Challenges

First, each query should be routed efficiently from each sink to motes with data of interest. This is challenging because the relevant motes are not known *a priori*, and simply flooding the entire network with every query would be too costly. Second, matching data should be efficiently routed back to each sink. This is challenging because sensor data from too many motes will cause congestion and, where there are many sinks, routing tables will grow too large to fit in memory.

Suppose a monitoring application sends queries to a WSN via a sink. This application requests data from the WSN periodically to monitor a physical environment. For example, "get all temperature measurements T greater than 25 degrees centigrade".

First, this query should be delivered only to a subset of sensor nodes. If there are a million sensors and only a few in one region have data of interest, flooding the entire network with this query would be a waste of resources. But, flooding is not necessary if each relay node has been given hints as to where such data is likely to be found. Second, after a query has been delivered to a mote, sensor data must be guided "up" to the sink. Third, distributed information for intelligent forwarding must be maintained while queries, sensor data, and the network topology change over time.

In order to provide a scalable solution, we must consider the case where there are many applications sending many different and complex queries via many sinks into the WSN. In this case the network may need to deliver data from many widely distributed motes to many sinks. This data can be reduced by a number of *in-network processing* techniques such as hierarchical clustering, data aggregation, duplicate suppression, filtering, and data-centric routing. These techniques may be combined to process data as close to the data source as possible to reduce the total quantity of data transmitted.

Also, each node must maintain much more knowledge about the network in order to intelligently forward packets, and that knowledge must be updated at a rate that matches the current conditions. Since sensor nodes are very memory constrained, the amount of routing information they can store is limited. To reduce communication costs, which is the fundamental challenge addressed by this thesis, routing information should be stored in and maintained by a scalable, space, time and energy efficient technique.

1.3 An Outline of Our Solution

First, the network must self-organize such that it provides a framework for efficient communication. We use a fully distributed protocol for self-organization that utilizes local feedback to generate hierarchical clusters for data aggregation. However, since the scale of homogeneous networks is limited, we also apply our clustering protocol to deploy less resource constrained devices.

Second, the network must maintain distributed information to guide query dissem-

ination. We have adopted the data-centric communication paradigm, where message content drives the system, and use scalable techniques based on Bloom filters to reduce communication costs. The combination of a fully distributed protocols for self-organization, using a local feedback mechanism, and data-centric communication, using Bloom filters [11], provides a scalable and energy efficient solution.

A Bloom filter is a space efficient, probabilistic data structure that represents a set of elements. The same set of hash functions is used store a set of elements in the filter and to test elements against the filter for set membership. While false positives are possible, their rate may be reduced by allocating more memory [11]. In this thesis, Bloom filters play a central role in reducing communication costs.

1.3.1 Distributed Clustering

To facilitate in-network processing and message delivery, we have developed a protocol for self-organization, Distributed Asynchronous Clustering (DAC) [49], that generates energy efficient topologies. While hierarchical clustering has been presented in other work such as Low Energy Adaptive Clustering Hierarchy (LEACH) [55, 53], the DAC protocol consistently generates cluster heads that are more evenly distributed.

In protocols that generate random clusters, a small percent of random nodes volunteer to be cluster heads without regard to local events. In contrast, DAC clusters are formed asynchronously such that no two proximate nodes become cluster heads. As a result, a rational hierarchy of cluster heads are identified to maintain and use distributed data structures based on Bloom filters to make routing, storage, and aggregation decisions.

1.3.2 Bloom Gradient Routing

The current trend in hardware development for sensor networks indicates that the memory capacity of motes has been increasing steadily, but severe energy constraints remain [88]. In this thesis, we use techniques based on Bloom filters [11] to suppress unproductive radio transmissions. These techniques exploit the trade-offs, between com-

putation and radio costs and between memory and radio costs, to reduce latency and energy consumption. As a result, the network responds faster to queries and network lifetime is extended.

Our Bloom Gradient Routing (BGR) protocol implements data-centric routing with Bloom filters, which provide a compact, probabilistic representation of a set of elements [11]. In data-centric routing, sensor attributes are used to name data, and packet delivery is based on named data rather than node addresses. BGR uses Bloom filters to represent named data and sensor meta-data and, thereby, disseminates queries at a lower energy cost than techniques used by Directed Diffusion [63], and other deterministic algorithms.

In addition, BGR uses a compound data structure which we call a Queued Bloom Filter (QBF). A QBF is a set of Bloom filters in a queue such that the oldest filter is removed from the front of the queue and discarded, and new Bloom filters are added to the back of the queue. Information about queries and sensor generated data are stored in separate QBFs, one for each neighbor. They are used by the network to discard stale data and use recent data to make intelligent routing decisions, thereby enabling highly adaptable and energy efficient packet delivery.

A large body of work has addressed the problems related to extending network lifetime. However, most do not use Bloom filters at all, some use flat topologies of homogeneous nodes, some make unrealistic assumptions, such as unlimited radio range [55, 53], and some present solutions that use techniques, such as flooding [63], that are not scalable.

1.4 Organization of the Thesis

This thesis begins with three background chapters. An overview of wireless sensor networks in Chapter 2, a survey of the Bloom filter data structure in Chapter 3, and a survey of Bloom filter applications in Chapter 4. The main contribution chapters of this thesis are divided into two parts. In the first part, Chapters 5 and 6, we focus on distributed protocols for the self-organization of wireless sensor networks. In the sec-

ond part, Chapter 7, we focus on efficient query dissemination and our novel technique for suppressing unproductive queries. In detail, the chapters in the thesis are organized as follows.

Chapter 2 – Wireless Sensor Networks provides a concise overview of of sensor networks: their applications, resource constrained hardware, wireless communication, routing, query processing, and highlights several design issues. This chapter provides the motivation for developing new application specific protocols for self-organization and query dissemination.

Chapter 3 – Bloom Filter Data Structures provides a survey of this space efficient data structure, which is easy to implement in hardware and requires few assumptions. We present details regarding its construction, operation, and show that their properties make them useful in resource constrained domains – especially in sensor networks where multiple resource constraints limit or invalidate existing protocols.

Chapter 4 – Applications of Bloom Filters provides a brief survey of applications, especially in the context of networks. Although the original motivation was lack of memory space, their utility has been rediscovered many times in many domains, but an apparent under appreciation in the wireless sensor network domain. The severe resource constraints (not only memory) imposed by WSN hardware motivates the application of Bloom filters presented in this thesis.

Chapter 5 – Distributed Asynchronous Clustering presents our fully distributed hierarchical clustering protocol for scalable homogeneous wireless sensor networks.

Chapter 6 – Clustering for Small-World Properties shows that distributed clustering protocols, such as presented in Chapter 5, should use the pre-emptive recruiting mechanism to generate heterogeneous networks with small-world properties, i.e., highly clustered networks characterized by short path lengths between nodes.

Chapter 7 – Bloom Gradient Routing in a Small-World presents our protocol for data-centric routing in a small-world network of *heterogeneous* nodes.

Chapter 9 – Conclusions and Future Work concludes this thesis by summarizing its main contributions and highlights a number of possible directions for future research.

Appendices A – E provide a convenient reference to additional background information on Micro-Electro-Mechanical Systems Technology, Radio Propagation Models, Signal to Noise Ratio, Calculating the Optimal Number of Hash Functions, and The MD5 Algorithm and Hashing.

1.5 Contributions of the Thesis

Chapter 5 presents a fully distributed hierarchical clustering protocol for *homogeneous* sensor networks where limited beacon range is used as the primary parameter for self-organization. In this protocol, nodes volunteer asynchronously for cluster head duty and use a radio beacon to pre-emptively recruit members. This mechanisim, which we call *pre-emptive recruiting*, generates an energy efficient topology characterized by well separated cluster heads, and enables performance that is superior to comparable protocols. To further extend network scalability, lifetime and capability, well separated cluster heads could be easily located and replaced by more powerful devices. This extension is presented in Chapter 6.

Chapter 6 shows that fully distributed hierarchical clustering protocols should use our *pre-emptive recruiting* mechanism to generate *heterogeneous* networks with small-world properties, which are found in many real world, large scale networks. The main contribution of this chapter is that the scalability protocols for network self-organization should be measured by their ability to generate small-world properties using a mechanism such as pre-emptive recruiting. In addition, while short paths exist in small-world networks, the network must be able to find them. A technique to support such path finding is presented in Chapter 7.

Chapter 7 presents a novel data-centric routing protocol based on compressed Bloom filters. Given a *heterogeneous* network characterized by a small-world topology, the Bloom Gradient Routing protocol distributes information such that queries and sensor data are routed along short paths to relevant destinations. The combination of sensor diffusion and sparse Bloom filters provides a more cost effective technique for query suppression and path finding than offered by comparable protocols reported

in the literature.

1.6 List of Publications

The following publications were generated during the course of conducting the research for this thesis.

1.6.1 Published Papers

- P. Hebden and A.R. Pearce, "Bloom filters for data aggregation and discovery: a hierarchical clustering approach", *Second International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, December 2005.

- P. Hebden and A.R. Pearce, "Distributed Asynchronous Clustering for Self-Organisation of Wireless Sensor Networks", *Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP)*, Bangalore, India, December 2006.

- P. Hebden and A.R. Pearce, "Data-Centric Routing using Bloom Filters in Wireless Sensor Networks", *Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP)*, Bangalore, India, December 2006.

- P. Hebden and A.R. Pearce, "Distributed Asynchronous Clustering for Self-Organisation of Wireless Sensor Networks", *International Journal on Information Processing (IJIP)*, Volume 1, Number 2, 2007.

1.6.2 Papers under Review

- P. Hebden, L. Kulik, and A.R. Pearce, "Self-Organization of Wireless Sensor Networks for Small-World Properties", *Computer Networks*, IEEE , 2009. (in progress)

- P. Hebden, L. Kulik, and A.R. Pearce, “Bloom Gradient Routing in Wireless Sensor Networks”, *Ad Hoc Networks*, Elsevier, 2009. (in progress)

Chapter 2

Wireless Sensor Networks

2.1 Introduction

Wireless sensor networks monitor the physical world. They may be used to detect, measure, and record almost any physical event or property including chemical, image, light, moisture, motion, pressure, radiation, sound, temperature, and vibration.

They have been used to monitor conditions in many environments. For example, wireless sensors have been deployed to monitor the temperature, soil moisture, light, and humidity of large vineyards. The goals included precision harvesting, watering, fertilizing, delivering pesticides, frost protection, predicting insect/pest/fungi development and developing new agricultural models [66]. WSNs should easily outperform traditional methods which reported little more than the average temperature over a wide area [9].

Some networks have been deployed to monitor contaminant transportation, marine microorganisms, seismic activity, and residential homes [18]; monitor equipment in a semiconductor plant and on an oil tanker in the North Sea [78]. Other deployments include habitat monitoring [147, 92, 143], burglar alarms, inventory control, medical monitoring and emergency response [156], and battlefield management [26].

The sensor nodes (motes) are autonomous devices composed of a processor, a sensor or sensors for measuring physical phenomena, a radio transmitter/receiver, and a power source. However, they present fundamental protocol design challenges in terms of scalability and the efficient use of bandwidth and energy.

In terms of node architecture, a number of possible trade-offs can be considered. The TinyOS approach advocates aggressive communication strategy in order to reduce complexity of computation and storage at local sensor nodes [61]. On the other hand,

the sensor-centered approach advocates aggressive sensor data processing, filtering, and compression in order to reduce communication costs [34].

In this chapter we provide details about the hardware used to construct motes, discuss several important issues related to radio communication in sensor networks, and point out that WSNs not only present new challenges, but also they tend to be very application specific.

2.2 Hardware

Sensors and actuators are now being manufactured using micro-electromechanical system technology (MEMS) and low cost processors are being embedded in more devices than ever before. As component costs continue to fall, intelligent devices will become a pervasive and integral part of everyday life. This technology will change the way we interact with the physical environment [29, 114, 154].

In accordance with Moore's Law [105], the number of transistors on a cost effective chip continues to double every 18 months or so. A given computing capacity becomes exponentially smaller and cheaper as each year passes. This scaling is expected to continue for at least another decade or two [16]. However, there is no equivalent prediction for battery technology [150]. Battery capacity is growing at a rate as low as 3% per year [35]. Given that power consumption is likely to remain a serious challenge for the foreseeable future, energy efficient protocols optimized for tiny, battery powered motes are essential for long running WSNs.

2.2.1 Wireless Energy Constraints

Wireless sensor nodes provide advantages and disadvantages. Wireless nodes are easy to deploy in large numbers, and close to their targets. However, for many deployments, wireless also means very low power. Each sensor must be battery powered or make use of "trickle" sources by harvesting energy from the environment. For some applications, there will not be an viable alternative to battery power for many nodes and aggressive power management will be essential. This is likely to be the case even in industrial

environments because operating and safety regulations typically call for each piece of equipment to have a dedicated power circuit where separate power connections are required for each sensor node [78]. Techniques that reduce power consumption must be implemented in order to realize the advantages of WSNs.

2.2.2 Sensor Node Specifications

As of 2003, the Mica platform shown in Figure 2.1 was considered a representative sensor node. It had become the foundation for hundreds of WSN research efforts and had been sold to more than 250 organizations. It was a several cubic centimeter *sensor-actuator unit* similar in size to a pair of AA batteries. The main micro controller was an Atmel ATMEGA103L or ATmega128 running at 4 MHz and delivering about 4 MIPS [62]. As of November 2004, the ATmega128L was an *extremely resource constrained device* with, for example, memory of only 4K Bytes Internal SRAM, 4K Bytes EEPROM, and 128 K Bytes In-System Reprogrammable Flash [7].

For communication, the Mica has a low-powered radio from RF Monolithics (RFM TR1000). This radio was designed for short-range wireless data communications, and applications that require robust operation, small size, low power consumption and low cost. The transmitter has provisions for both ON-OFF keyed and amplitude-shift keyed modulation. The operation frequency is 916.5 MHz, and it can achieve data rates up to 115.2 Kbps [125]. However, in ON-OFF key mode bandwidth is limited to 19.2 Kbps [61]. By using specialized hardware accelerators and amplitude-shift-keying, the radio communication rate can be increased to 40 Kbps. Line-of-sight range is over 30 meters [60].

The MICA2DOT mote, shown in Figure 2.2, is similar to the Mica mote except for its 25mm form factor and reduced input output channel. It was designed specifically for deeply embedded wireless sensor networks [159].

The Mica node was designed to facilitate the exploration of wireless sensor networking, but it does not represent the vision of very small and cheap sensors nodes. The *Spec* node, shown in Figure 2.3, is closer to the vision. It is a single-chip CMOS device that integrates the processing, storage and communication capabilities to form

Figure 2.1: The MICA2 Mote, manufactured by CrossBow Technology, is about the size of a matchbox and uses 2 AA batteries.

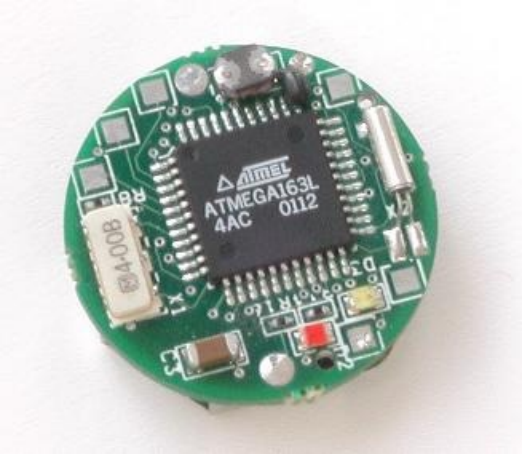


Figure 2.2: The MICA2DOT mote is about the size of a U.S. quarter and uses a "button" battery.

a complete system. It was designed in late 2002 by Jason Hill, fabricated by National Semiconductor, and demonstrated in March 2003. It measures just 2.5 mm x 2.5 mm, contains a micro controller, transmitter, ADC, general purpose I/O ports, UART, memory and encryption engine. This system only needs to be supported by a 32 KHz watch crystal, an off-chip inductor and a power supply, a battery and a 4 MHz clock. "The Spec node represents the coming generation of wireless sensor nodes that will be manufactured for pennies and deployed in the millions" [60].

A more recent development is the matchbox sized Java SunSpot, shown in Figure 2.4, from Sun Microsystems. It uses a 180MHz, 32-bit ARM 920T processor with 512k Bytes RAM and 4MBytes flash memory. Energy is provided by a 3.6 V, 750 mAh rechargeable lithium-ion battery.

Developers will use Moore's law to drive down the cost of these devices, not to increase their storage, computation, and communication capacities [73]. Implementations of WSNs are likely to continue to be subject to severe resource constraints and, moreover, the motes being sold by CrossBow Technology as of January 2009, such as the MICA2, MICAz, and TelosB, are very similar in size and appearance to the vintage 2003 mote shown in Figure 2.1.

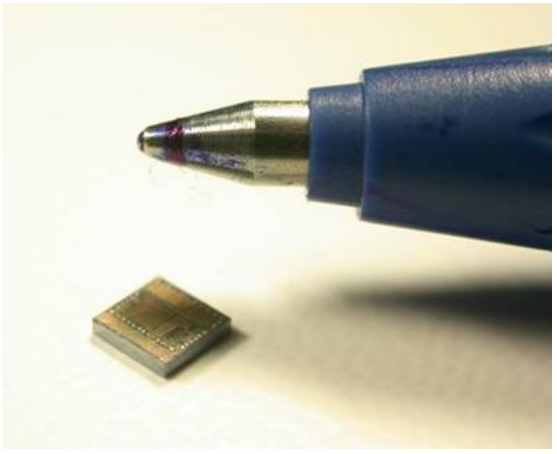


Figure 2.3: The Spec mote pictured beside the tip of a ballpoint pen. This chip contains all of the components found in a MICA mote: microprocessor, memory, A/D converter for sensor data, and a radio. To create a functional node, attach sensor(s), battery, and antenta.

Figure 2.4: Java Sunspot.

2.3 Wireless Communication

Radio communication is by far the most power intensive task, and it is expected to dominate energy consumption calculations for the foreseeable future [25]. While most sensornet devices are disadvantaged by their resource constraints, dense networks of sensors provide redundancy for fault tolerance, and may offer a better signal to noise ratio (SNR) (Appendix C) due to proximity to their targets. Also, large numbers of

sensors facilitate energy efficient multi-hop routing, data aggregation via in-network processing. Nonetheless, WSN protocols must first exploit the favorable computation communication trade-off in order to maximize network lifetime.

2.3.1 Wireless Links

First, wireless communication is notoriously unpredictable. The quality of each link depends on the environment, frequency spectrum, modulation schemes, and the hardware itself. Link quality can vary suddenly with time and small spatial displacements. Zhao and Govindan systematically evaluate packet delivery in [68]. Connectivity analysis, neighborhood management, and routing is explored in [158].

Second, the energy required for radio communication rises dramatically with distance. The power required to transmit over distance d is proportional to d^n where $n >= 2$ and depends on distance and the environment. For most WSNs, radio communication is the most power intensive task, far more than computation [118].

2.3.2 Link Quality

Link quality, as measured by message delivery performance, may be seen as the most basic aspect of wireless communication. Ideally, a simple mechanism would measure actual performance of links, which may be asymmetrical, and discard links to poorly performing neighbors. It has been estimated that, depending on the load, anywhere between 50% and 80% of communication energy is wasted on repairing lost transmissions [68].

2.3.3 Computation versus Communication

Transmitting data even over short distances consumes far more energy than processing it – the ratio has been estimated to be in the range of 1,000 to 10,000 to one. For example, the energy cost of transmitting 1Kb a distance of 100 meters is approximately 3 joules. With the same amount of energy, a general-purpose processor with 100MIPS/W power could execute 3 million instructions [118].

Given this trade-off, we must reduce the amount and distance of communication as much as possible through techniques such as local collaboration, duplicate data suppression, processing and storing data locally, using multi-hop routing, and compression – ultimately to facilitate data aggregation, discovery, and delivery of useful information to the end user over a long period of time.

2.3.4 Multi-hop Routing and Short Hops

When WSNs are deployed in a physical environment, researchers discover that radio communication is far less reliable than lab simulations would suggest [143]. In general the output power required to transmit over distance r is proportional to r^n. The exponent n varies from about 2 to 4, and is closer to 4 for low lying antennae and near ground channels [118]. This suggests that routing over many short hops will be more effective than routing over a few long hops.

Suppose a route of N nodes where the hop distance is r. Let $P_{receive}$ be the minimum receiving power at a node for a given transmission error rate, and P_{send} is the power level required at the sending node. The near ground RF attenuation model is

$$P_{receive} \propto \frac{P_{send}}{r^n}.$$

Without considering other factors, the power advantage of multi-hop over single-hop is

$$\eta_{rf} = \frac{P_{send(Nr)}}{N \cdot P_{send(r)}} = \frac{(Nr)^n P_{receive}}{N \cdot r^n P_{receive}} = N^{n-1}.$$

In practice each relaying of the message will consume power, increase latency, cause interference, and increase the probability that the packet will be dropped. An optimal design would need to consider the costs and benefits of multi-hop routing [32]. Larger networks will require more relaying and, beyond a certain size, they are expected to be heterogeneous [65]. In fact, the benefits of multi-hop have turned out to be much less than originally thought [151].

2.3.5 Multi-hop Routing and Long Hops

In recent years some researchers have claimed that multi-hop routing consumes less energy than single-hop [121], other researchers have claimed the opposite [45, 165]. Since required transmit power P_{tx} is proportional to Euclidean distance d raised to an appropriate path loss exponent n, $P_{tx} \propto d^n$, multi-hop appears to be inherently more efficient. However, it has been shown that whenever source and destination are within range, single-hop is almost always more power efficient than multi-hop routing [151].

When source and destination are not within range, and only motes are available as relay nodes, network topology should limit hop distance to ≈ 5 [75]. In [6], multi-hop reliability at the mote level in a three tier network became insufficient after 5 to 6 hops, so they partitioned their network into subnetworks where each mote was usually closer than 5 hops from a tier 2 device.

2.3.6 Medium Access Control

MAC sublayer protocols such as TDMA, CSMA, and CDMA may be used to arbitrate access to the physical network medium. They have been designed to reduce packet collisions and provide fair access to the channel. However, in WSNs the issue of fairness is less important than overall application performance, and lower performance may be traded for extended network lifetime [32]. For example, sensornet MAC protocols conserve energy by putting nodes to sleep as much as possible. The sleep/awake ratio can optimized for the application. Sensor MAC (S-MAC) [161, 153], Berkeley MAC (B-MAC) [117], and IEEE 802.15.4 [44] trade latency in packet delivery for energy savings.

2.4 Clustering to Reduce Communication Costs

Clustering is generally considered to be the unsupervised classification of unlabeled patterns (data points) into groups based on some distance metric. Clustering techniques fall in two broad categories: hierarchical and partitional. Hierarchical clustering produces a nested series of partitions – at each time step the closest nodes are merged.

This process continues until all nodes have been merged into one top level cluster. K-means is an example of a partitional technique. It partitions the data set (all nodes) into k clusters. It uses iterative relocation of cluster centroids to reduce within cluster variance, and is in the class of iterative optimization procedures [69, 27].

However, in this thesis we do not use clustering to classify unlabeled data points. We use clustering to facilitate in-network processing and data aggregation – two well known techniques that reduce communication costs in resource constrained sensor networks.

2.4.1 In-network Processing

In-network processing may be used to exploit the computation/communication trade-off. For example, instead of thousands of sensing nodes congesting the network with data transmissions, cluster heads aggregate data by performing some computation (max, min, mean, summation, compression, etc.), thereby reducing the quantity of data transmitted from multiple sources to a destination. To exploit the computation/communication trade-off, the network may need to in-network processing, such as compute aggregates, as close to the source as possible. For example, in Figure 2.5, the network on the left routes packets by shortest distance, the network on the right routes packets to facilitate data aggregation, i.e., in this example, to reduce the number of transmissions from 6 to 4.

Furthermore, some applications will require that network nodes process data co-operatively and combine data from multiple sensors. This is because transmitting raw data from all sensors to a base station does not scale to large networks. According to Gupta and Kumar, per node throughput scales as $\frac{1}{\sqrt{N}}$ – it goes to zero as the number of nodes N increases [43]. In other words, as N increases every node spends almost all of its time forwarding packets from other nodes.

Data from sensors close to each other is likely to be correlated, and redundant data can be reduced locally by filtering, in-network aggregation, and compression. After all, applications want useful information, not massive amounts of raw data. It can be shown that a network generates non-redundant data at $O(\log N)$ as N increases [131]. Data

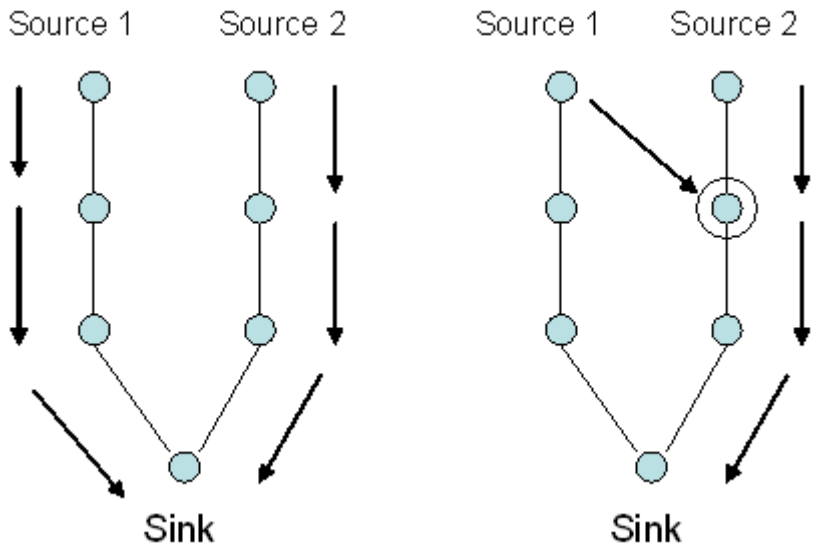


Figure 2.5: The network on the left: packets are routed by the shortest path to the sink. The network on the right: packets are routed to facilitate data aggregation.

generated per node scales as $O(\frac{\log N}{N})$ and is within the theoretical per node throughput capacity derived by Gupta and Kumar.

Given the importance of efficiency in WSNs, sensor nodes that participate in sensor collaboration must be chosen where their information value is weighed against their resource consumption. This is the issue of selecting embedded sensors for in-network processing [31].

2.4.2 Data Aggregation

Data aggregation reduces the amount of data that must be transmitted, and is perhaps the most common operation in WSNs [89]. Hierarchical clustering may provide a framework for data aggregation where sensors send raw measurements to their cluster head for processing, and cluster heads cooperate with each other to provide additional processing and deliver information to the base station.

The original SQL specification offered just five data aggregation functions: COUNT, MIN, MAX, SUM, and AVERAGE. However, WSNs are not limited to these functions and a query may, for example, specify that only a certain region of the network is of

interest. Aggregate functions may be classified according to four properties: duplicate sensitive, exemplary or summary, monotonic, and partial state. For example, COUNT, SUM, and AVERAGE are sensitive to duplicate messages, AVERAGE is summary, and MIN is exemplary [90].

Given a variety of aggregate functions, nodes may need to maintain an abstract record of recent events such as queries, sensor measurements, and topological changes. Individual sensors may need to determine which other sensors are in their cluster in order to make intelligent decisions about forwarding messages and transmitting a current measurement. Cluster heads may need to determine which sensors are in their cluster, and maintain a memory of recent measurement messages in order to detect changes in the environment being monitored, and to discard duplicate messages.

2.5 Routing

Complex queries must be routed from a sink to sensor nodes with matching data. When routing queries, each node needs information about what data or measurements have been recorded in neighboring regions of the sensor network. This requires an efficient mechanism for storing and maintaining historical information. When routing sensor data from source to sink, each node should forward that data along the shortest path to the sink that was the source of the query. To reduce the need for some degree of flooding, the network must maintain space efficient representation of past events, e.g., queries and sensor measurements, and their age. In other words, *WSNs are so fundamentally different from traditional wired and wireless networks that they require a paradigm shift in protocol design.*

The Internet delivers most data from one server to many clients, or between arbitrary pairs of nodes. In a WSN, most data flows from many sources (sensors) to one sink (base station). Also, the data generated by sensors in the same area tends to be based on a common phenomenon, is correlated and redundant to some degree. And, most importantly, many WSNs must operate on very low amounts of power.

Many end to end protocols have been proposed for mobile ad hoc networks, but

WSN nodes are usually stationary. Consequently, for this reason and those above, data aggregation is seen as a useful technique for WSN routing. By summarizing data en route, both redundancy and the number of transmissions is reduced, and energy is saved. This suggests a paradigm shift from traditional address-centric routing to data-centric routing [76]. WSN should use routes from many nodes to one sink that facilitate in-network processing of redundant data, not necessarily the shortest routes.

Address-centric routing is designed to find short routes between pairs of uniquely addressable end nodes. In contrast, *data-centric* routing is designed to aggregate data and find routes from multiple data sources to a destination by using named data. Nodes are addressed by the attributes of their data, location, proximity, or capability rather than a globally unique identifier. It has been shown that data-centric routing offers significant performance gains across a wide range of operational scenarios [76].

2.6 Query Processing

Query processing systems, such as Directed Diffusion [63], COUGAR [12], and TinyDB [91], provide an SQL user interface and gather data from distributed sensor nodes. They must carefully manage power and radio bandwidth to guarantee that essential data is collected and delivered to the user application in a timely fashion. Systems can be classified based on whether they are time-driven or event-driven, and the type of queries they process.

There are three basic types of queries [12]:

- *Continuous queries* commonly span a long period of time. For example: get current temperature every 60 seconds for the next 5 hours.
- *Snapshot queries* collect data about now or some other point in time. For example: get current temperature.
- *Historical queries* collect summary data about the past. For example: get the average temperature for 1999. [1]

[1] This type of query assumes that such data was stored on a relatively large and robust device rather than on a small, fragile sensor node.

Queries for sensor networks tend to be continuous and have long life spans. However, sensing and reporting requirements will depend on the application. Time-driven nodes sense and transmit data at constant periodic time intervals to provide periodic snapshots where latency is not likely to be an important factor. In contrast, event-driven nodes react to certain changes in their sensor measurements an data delivery is likely to be time critical [4].

2.7 Design of WSNs

Unlike traditional networks, WSNs are very application specific and their design should consider trade-offs between several evaluation metrics: lifetime (power consumption), coverage, cost, ease of deployment, response time, temporal accuracy, security, and effective sample rate [62].

Design may be influenced by many factors including fault tolerance, scalability, production costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption [2]. For many applications, the most critical factor is power consumption. Even if some nodes in a network are able to harvest energy from their environment, other nodes may not be able to due to their location, or the additional financial cost of components such as solar panels.

The fundamental challenge is to reduce power consumption by minimizing communication costs. This will help to extend the operational life span of battery powered sensors, reduce congestion, and expedite queries.

Early research projects defined a WSN as a large scale, ad hoc, multi-hop, unpartitioned network of largely homogeneous, low cost, randomly deployed sensor nodes. Some projects assumed that nodes would not even have unique addresses or IDs within their hardware. However, by late 2004 many systems were being built using a variety of devices. Each device may differ in terms of its sensors, processing power, storage capacity, power source, routing ability, location awareness, and so on [128].

2.8 Related Work on Clustering and Routing in WSNs

In this Section we provide a brief summary of related work. In Chapters 5, 6, and 7 we provide a more extensive reviews of closely related work on clustering, the importance of generating network topologies with small-world properties as shown by research on real world complex networks, and routing based on data attributes rather than network addresses.

While clustering for data aggregation and routing has been studied extensively, the clustering protocol presented in this thesis, Distributed Asynchronous Clustering (Chapters 5 and 6), is most comparable to those proposed in [53], [86], and [23]; and the routing protocol presented in this thesis, Bloom Gradient Routing (Chapter 7), is most comparable to Directed Diffusion [30, 63, 64, 137].

The goal of the clustering algorithm proposed in [21] is to maximize network lifetime. However, it assumes that each node is aware of the topology of the entire network. This is not feasible for large WSNs. Other clustering algorithms were designed to generate stable clusters in environments with mobile sensor nodes [23], whereas in this thesis all sensor nodes are stationary.

Heinzelman, et al., [53] proposed a partially distributed algorithm where sensor nodes elect themselves as CHs with some probability and broadcast their decisions. Unclustered sensor nodes bind to the CH that requires minimum communication energy. All clusters are on the same level, and their algorithm allows only 1-hop clusters to be formed. The algorithm is rerun periodically so that none of the sensor nodes are overloaded due to performing CH tasks. However, it is not clear how to compute the optimal number of CHs prior to or during execution, and this protocol fails to generate cluster heads where needed

In [23] Bandyopadhyay and Coyle point out that many clustering algorithms have been proposed, but none aim at minimizing the energy spent by the system to gather data from the network, and an algorithm that generates the minimum number of clusters might not ensure minimum energy usage. This should be obvious as soon as one considers the impact of too few clusters, i.e., the radio links are likely to be too long and result in high packet loss and retransmissions.

An optimal cluster hierarchy should use minimal energy to communicate information from data sources to the local base station. They, [23], proposed a fast, randomized, distributed algorithm for organizing sensor nodes into a hierarchy of clusters. Their objective was to minimize the energy spent during operation, i.e., in communicating the information to the information processing center [23]. They assumed a multi-hop network of nodes equipped with fixed power level transmitters, and designed their algorithm to generate one or more levels of clusters with an optimal number of CHs on each level [23]. However, a protocol designed for a homogeneous network of nodes with fixed power transmitters will not adapt to changing conditions nor will it scale to large geographic areas where more powerful devices provide long range connectivity.

Directed Diffusion [30, 63, 64, 137] is a routing protocol for flat sensor network topologies that routes queries to sensors and sensor data to sinks. This protocol uses an attribute-based naming scheme. The sink injects a query into the network without prior knowledge of which sensor nodes have data of interest. Consequently, it broadcasts the query to the network rather than sensor nodes with specific addresses. The interaction of disseminated queries and matching sensor data establish *gradients* for data to flow toward the sink that expressed that interest. However, it fails on efficiency grounds due to its reliance of flooding and it fails on scalability grounds due to its flat topology and the space required for the routing table at each relay node.

2.9 Conclusion

This chapter provided an overview of WSNs, their limited hardware and the consequent paradigm shift in particular, and suggests three fundamental conclusions.

First, given resource constrained motes, a flat network topology will not scale. If all nodes are tasked with forwarding all received sensor data, many will fail, the nodes closest to the sink will fail first, eventually causing the network to fail. Therefore, hierarchical clusters should be generated by a mechanism such that well separated nodes are identified as cluster heads (centroids) and "similarity" is based on radio distance.

Given a hierarchy of cluster heads, computation in the form of in-network processing and data aggregation are used to reduce communication costs.

Second, given a hierarchy of clusters, a homogeneous network of motes will not scale. The additional burden placed on cluster heads will cause them to fail due to their limited bandwidth and energy. Furthermore, the degree of hardware heterogeneity (in terms of memory, processing, energy, etc.) influences the complexity of software running on the sensors and other devices in the hierarchy, and scalability is limited by node capacity at bottlenecks.

Third, given a heterogeneous network of hierarchical clusters, address-centric routing is not suitable for applications that do not know which nodes in the network can or should be sent selective queries. Therefore, a new energy and bandwidth efficient technique is needed to suppress queries.

In this thesis, a self-organizing network of heterogeneous devices, especially where some nodes are equipped with long range radios and do not rely on batteries as their sole energy source, is used to simulate a scalable wireless sensor network, i.e., one characterized by short distances between motes and their cluster heads and short path lengths between packet sources and destinations. And, routing and query suppression exploit the space efficiency Bloom filters for the storage and transmission of information. In the next two chapters we present Bloom filters, how they work and a brief survey of their applications.

Chapter 3

Bloom Filter Data Structures

3.1 Introduction

Burton Bloom published *Space/Time Trade-offs in Hash Coding with Allowable Errors* in 1970 [11] when computer memory was extremely scarce. In that paper he described what has come to be known as a Bloom filter: a space-efficient probabilistic data structure that represents a set.

This data structure is similar to a conventional hash table, but does not support conventional operations such as *put*, *get* and *delete*. It supports set representation and probabilistic set membership operations with a bit vector and k hash functions. Space requirements are significantly below the information theoretic lower bounds for an error-free data structure [15]. This space efficiency benefit is gained at the cost of a loss of information and errors in the form of false positives where some nonmember elements will test positive for set membership. If an application can tolerate a low rate of false positives and memory is scarce, then Bloom filters may provide a practical optimization technique [15].

The term *standard Bloom filter* is often used to distinguish the original from variations and extensions. For example, a filter with several bits per cell is known as a counting Bloom filter. This makes it possible to keep track of the number of times an element has been stored and to delete an element [15]. Sets of filters can be combined to form a compound structure known as an attenuated Bloom filter (ABF) [126]. An ABF may be evaluated and the result used to determine whether or not an element was probably stored h hops away. More specifically, an ABF can be used to compare adjacent nodes to make routing decisions.

This data structure is relevant today because it provides the designer with a convenient and effective optimization technique that trades memory for accuracy and lower communication costs.

3.2 Standard Bloom Filter Definition

This filter uses a simple bit vector with one bit per cell. To construct a filter, the following procedure is used. Suppose n elements in set S are selected randomly from a large universe U, and we have m bits of memory available. The hash area may be defined as m individual bits with indices 0 through $m-1$. Generate k different indices with k different hash functions applied to each element, and set those bits to 1. Figure 3.1 shows a filter after one element has been stored. To test an unknown element for set membership, follow the same procedure except just compare each of those k bits to 1. If all are equal to 1, then the element is recognized and probably is a member, else it is not a member.

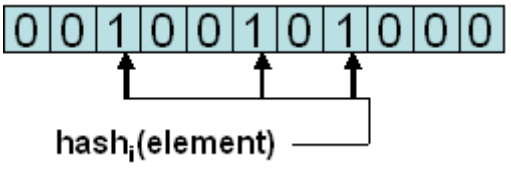


Figure 3.1: **Storing elements in a Bloom filter:** Up to k bits will be set to 1 by k hash functions and one element.

A false positive occurs when we test a new element and it hashes to k bits that have already been set to 1 by one or more other elements in S. For example, given a filter where 50% of the bits have been set to 1 and one hash function, the expected false positive rate would be 0.5. However, if two hash functions are being used, the probability that an unseen element will test positive is 0.25.

In a standard filter, given n elements and m bits of memory, k may be chosen to minimize the false positive rate. An optimal k turns on 50% of the bits when the filter is built [15].

Standard Bloom filters have several interesting properties. Firstly, their false positive rate may be made arbitrarily small by allocating more memory. This property is crucial for applications that can tolerate errors and, for those applications, "false positives may be acceptable as long as their probability is sufficiently small" [15]. If we know the maximum acceptable error rate, then we must determine the amount of memory available, the number of elements to be stored, and the optimal number of hash functions k. Secondly, false negatives do not occur with standard Bloom filters. Bitwise operations are extremely efficient. Filter size does not depend on the size of the element being stored; it is $O(|S|)$.

However, there are several drawbacks. First, except for the trivial case of $n = 1$, an element may not be deleted from a standard Bloom filter (by setting its k bits to zero) without the risk of creating false negatives. Second, information is lost when elements are stored: it is not possible to reconstruct the stored set S because $|U| >> |S|$ and, therefore, the number of false positives could be far greater than the number of true positives, thereby rendering the "recovered" set useless. Third, given a false positive rate f, an optimal Bloom filter requires at least $1.44lg(1/f)$ bits of space per inserted key, which is 44% more space than the $lg(1/f)$ per key required by information theory. Fourth, k is proportional to $1/f$, whereas a hypothetical equivalent optimal data structure would need only a constant number of hash functions independent of the false positive rate [116].

Alternatives have been proposed, such as [116, 46], but they are outside the scope of this thesis. Here we note that while Bloom filters are not ideal, they provide practical trade-offs. They are known for their efficiency and speed [87], and require few assumptions, e.g., they do not assume that the set of elements to be stored is known beforehand and partitioned; and they are relatively easy to implement in hardware [46]. In summary, the literature on their use for various networking problems has been growing [46].

3.3 Standard Bloom Filter Optimization

Usually, given a memory constraint and a set of elements to be stored, the filter is optimized to minimize the rate of false positives (error rate). In standard Bloom filters there are three parameters: m, n, and k where m is the number of bits in the filter, n is the number of elements in set S, and k is the number of hash functions. Given m and n , we can determine the number of hash functions k that minimizes the false positives [15].

After the filter has been built, the probability p of a bit remaining zero is

$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}. \tag{3.1}$$

So the probability of a false positive f is

$$f = (1 - p)^k. \tag{3.2}$$

We use the approximations that p equals the probability that a bit in a constructed filter is still 0 and f is the probability of a false positive.

The derivative of f with respect to k equals zero and is a global minimum when

$$k = \frac{m}{n} \ln 2. \tag{3.3}$$

In practice k must be an integer, and a smaller k may be preferred to reduce the amount of computation. The derivation of Equation 3.3 may be found in Appendix D and [15].

Given an optimal k, half the bits set to 1, and that $\left(\frac{1}{2}\right)^{\ln 2} \approx 0.6185$, the false positive probability f may be expressed as

$$f = \left(\frac{1}{2}\right)^k = (0.6185)^{\frac{m}{n}}. \tag{3.4}$$

Figure 3.2 shows that for a given amount of memory, saturation increases with n and k. As a filter becomes more saturated, it becomes more error prone. A filter with all bits set to 1 will test positive for all elements in U.

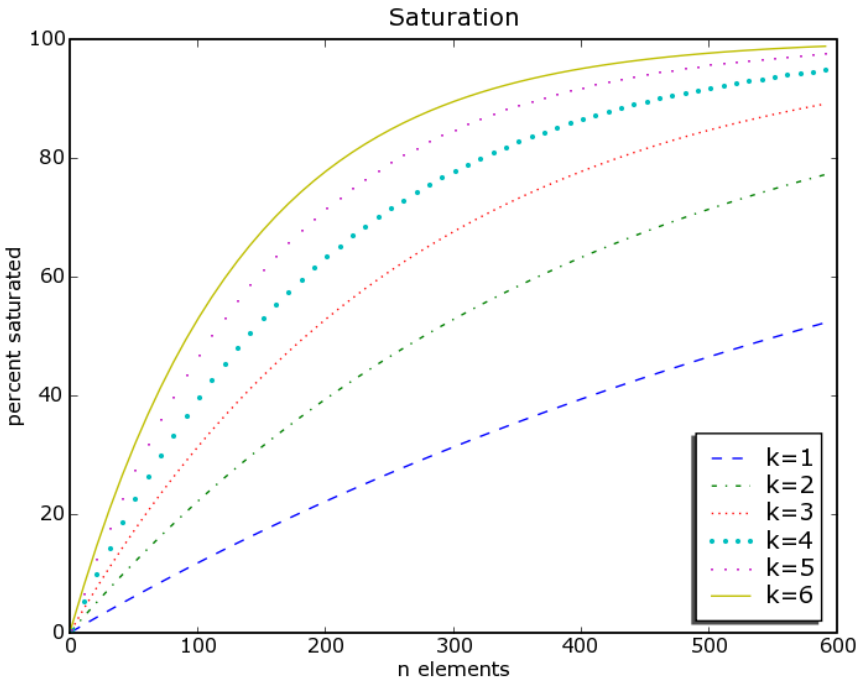


Figure 3.2: Given m, the level of saturation depends on k and n.

Figure 3.3 shows that for a given amount of memory and number of hash functions, the false positive rate increases with n. As more unique elements are stored, more bits are set to 1, the filter becomes less useful. For example, suppose $|S| = 100$ and $|U| = 1000$, and a filter where all bits have been set to 1. Assuming that elements from U are selected at random and tested against the filter, the expected false positive rate would be 90%.

Figure 3.4 shows that for a given false positive rate, the memory requirement m increases linearly with n. On other words, the space requirement is $O(|S|)$.

Figure 3.5 shows that given m and n, the rate of false positives depends on k. Given two parameters, we can optimize for the third. In this case, k.

More hash functions provide more chances to find a zero bit and prove that a random element x drawn from U is not in S. However, using more hash functions sets more bits to 1 in the first place – making a false positive more likely. Given n and m, an optimal k will build a filter that is 50% saturated.

The following equations may be used where we assume that the filter is 50% satu-

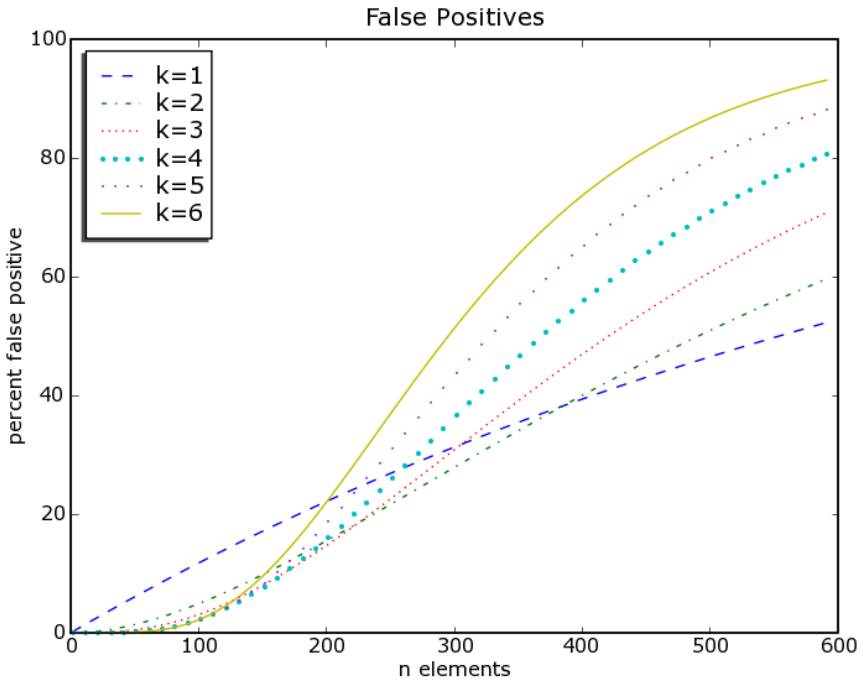


Figure 3.3: Given m, the rate of false positives depends on k and n.

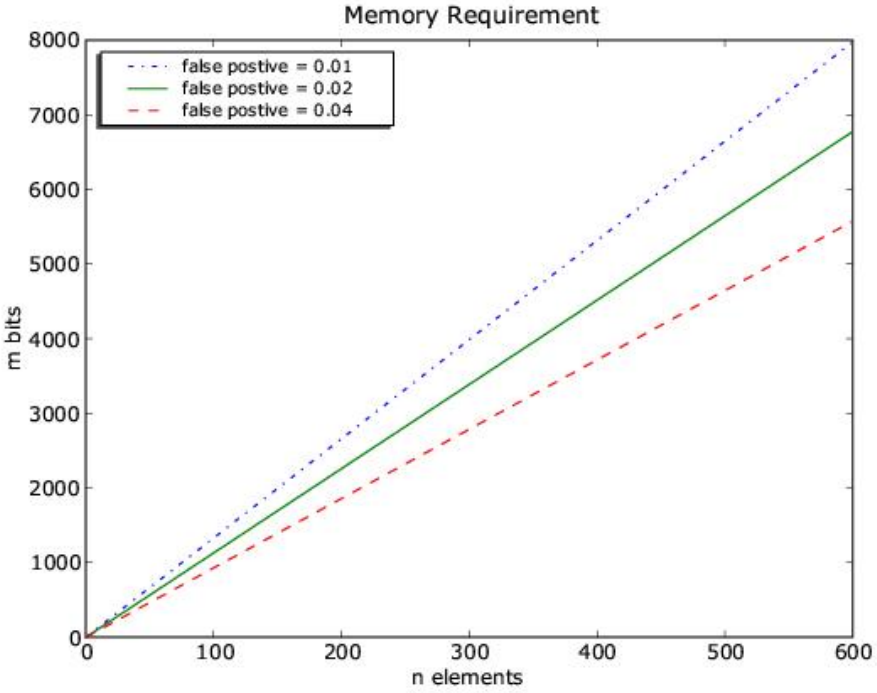


Figure 3.4: Given a rate of false positives, m depends on n.

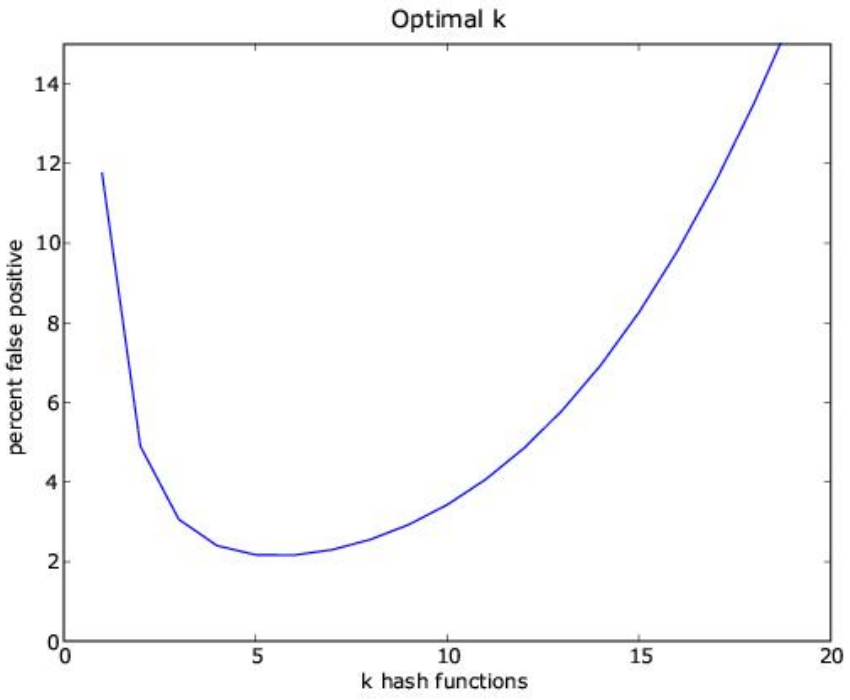


Figure 3.5: Given m and n, the rate of false positives depends on k.

rated. False positive probability f may also be expressed as

$$f = \left(\frac{1}{2}\right)^{\lg \frac{1}{f}}. \tag{3.5}$$

The number of hash functions k must be

$$k = \lg \frac{1}{f}. \tag{3.6}$$

Now we can estimate filter size m given n and f.

$$m = \frac{n \lg \frac{1}{f}}{\ln 2}. \tag{3.7}$$

Figure 3.4 shows that given an acceptable false positive rate f, there is a linear relationship between the amount of memory required and the number of elements to be stored. For each additional element, a constant number of additional bits will be required. However, as Figure 3.6 shows, given a certain number of elements to be stored, the relationship between the amount of memory required and the acceptable

false positive rate is nonlinear. The largest marginal savings in memory occur when the acceptable error rate is increased from a very low rate of less than 1% (where the curve is almost flat).

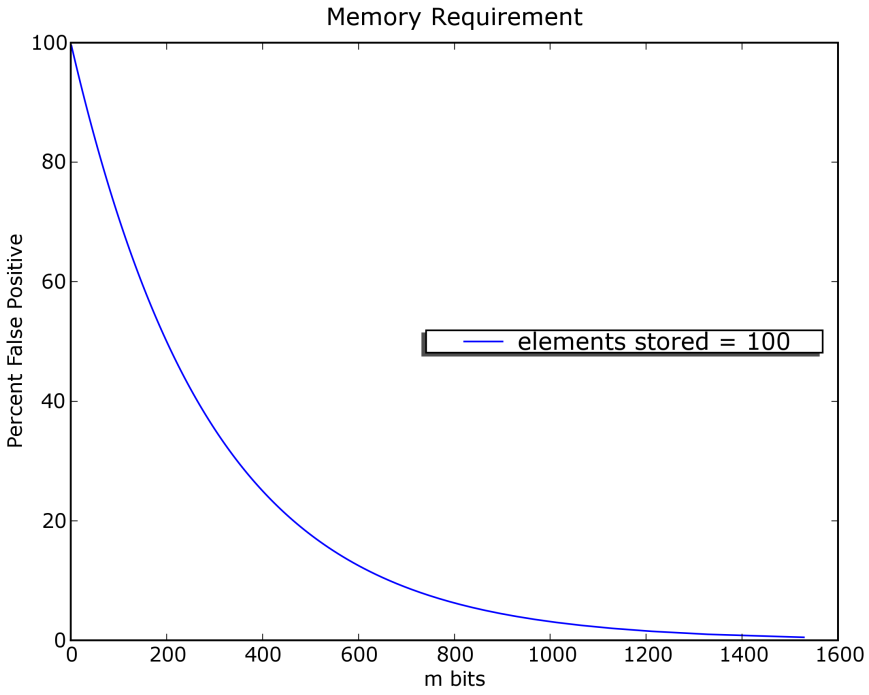


Figure 3.6: Given n elements, a lower false positive rate f requires a nonlinear increase in the number of bits m.

3.4 Compressed Bloom Filters

As discussed in Section 3.3, given memory m and number of elements n, a Bloom filter may be optimized for false positives. By allocating more bits per set element, the false positive rate can be made arbitrarily low. However, a filter optimized for memory is not optimal for communication when the filter itself will be transmitted as a message [102]. A filter optimized for communication will require more memory at the end points, but less energy for transmission and reception.

In the network context, we now consider four fundamental performance metrics: (1) computation time t, (2) probability of error f, (3) storage size m, and (4) transmission size z. Since energy consumed by radio communication is our primary concern,

we will optimize for transmission size z. However, given a maximum transmission size of z, where $z < m$ and an optimal standard filter, information theory tells us that we cannot compress this filter down to z bits.

3.4.1 Entropy

Theoretically, an m bit filter can be compressed to $mH(p)$ bits where p is the probability that any single bit in the filter is 0, and $H(p)$ is the entropy function.

$$H(p) = -p \lg p - (1-p) \lg(1-p) \tag{3.8}$$

An optimal filter where each bit has an equal and independent probability of being a 0 or a 1, has an entropy of 1. As Figure 3.7 shows, entropy is maximized when $p = 0.5$. If p is large enough, $H(p)$ will be significantly smaller than 1. As Figure 3.8 shows, compressed filter size z decreases as p increases beyond 0.5. However, this figure simply shows the impact of saturation on compression, it does not hold the number of elements to be stored, n, constant.

For Internet applications, a *sparse filter* could be compressed by arithmetic coding [103], which is the simple compression scheme recommended by [102]. However, to compress Bloom filters in WSN applications, Golomb-Rice coding [130] has been recommended because of its low computational requirements and effectiveness [19]. If we build, compress, transmit, receive, decompress, and test, then energy consumption will be less when the filter has been optimized for communication.

3.4.2 Optimization

A sparse Bloom filter will yield the same or better false positive rate, and yet fewer bits will need to be transmitted. In other words, the compressed filter is smaller than its conventional cousin, but performs as well or better. An additional benefit is that compressed filters use a smaller number of hash functions, so lookups require less computation.

The trade-off costs are the increased processing requirement for compres-

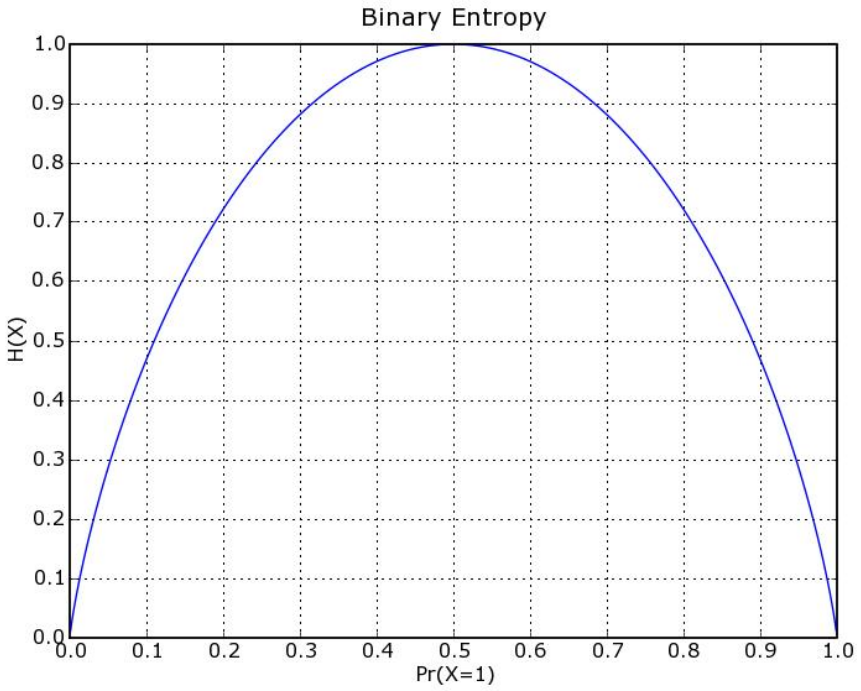


Figure 3.7: The information entropy of a Bernoulli trial X reaches a maximum of 1 when $Pr(X = 1) = Pr(X = 0) = 0.5$.

> sion and decompression and larger memory requirements at the endpoint machines, which may use a larger original uncompressed form of the Bloom filter in order to achieve improved transmission size. [102]

Assuming that transmission size is more constrained than memory and computation, the optimization problem is as follows. Given n and z, find m and k such that f is minimized and the compressed filter size is no more than z. The next question is how much memory, m, can we afford at the endpoints? **A very large filter may be compressed down to z bits if it is sparse enough.** Given m and z, we can calculate p, the probability of a bit being 0 in the uncompressed filter, with the help of the entropy function $H(p)$.

$$\frac{z}{m} = H(p) = -p \lg p - (1-p) \lg(1-p) \tag{3.9}$$

After solving for p, we can solve for k.

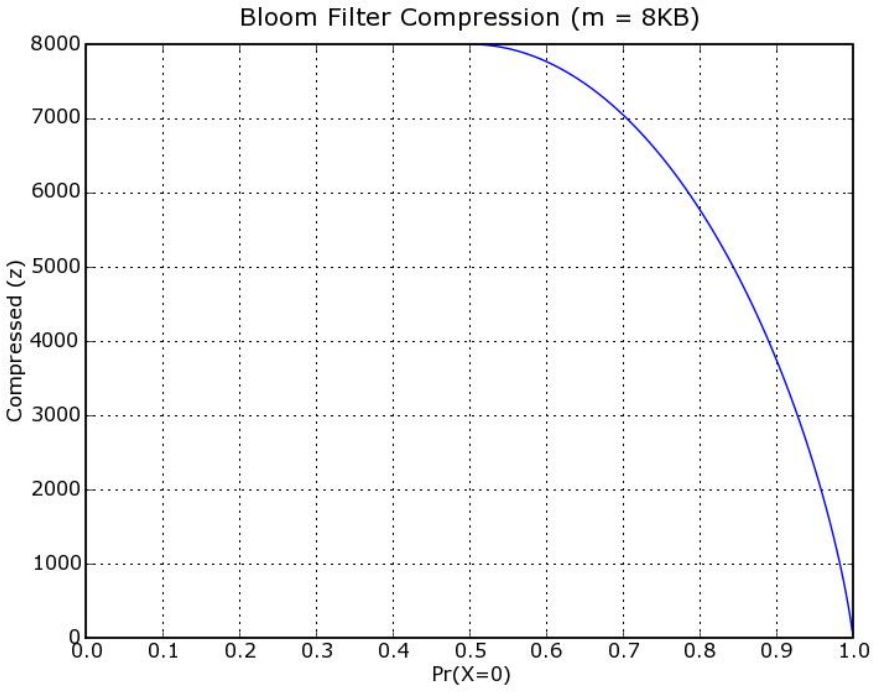


Figure 3.8: A sparse Bloom Filter may be compressed from m to z bits.

$$k = \frac{m}{n} \ln \frac{1}{p} \tag{3.10}$$

Equation 3.10 is really the same as the equation used for uncompressed filters, $k = \frac{m}{n} \ln 2$ where $2 = \frac{1}{0.5}$, except now we allow for the case where $p > 0.5$.

It should be noted that the k that minimizes f without compression maximizes f with compression [102]. Table 3.1 shows that when k is reduced from 6 to 2, or from 6 to 1, the false positive rate is substantially lower. This is because we are using a larger filter for the same n, and we are no longer optimizing k for a filter that will be 50% saturated. A less saturated filter offers more chances for an input to hash to an empty cell. The larger filter is less than 50% saturated and can be compressed down to z.

Bloom filters may be optimized for a limited transmission size z. Given n and z, m and k are chosen to minimize f subject to $mH(p) \leq z$. In the alternative, transmission size z can be minimized given n inputs and a limited false positive rate f.

Under ideal circumstances, the best possible compressed Bloom filter achieving the same f as the standard Bloom filter would have $z = m \ln 2$, a savings in transmission size of roughly 30%, which is "significantly better than what can be realized in practice" [102].

For example, suppose we must limit our transmissions to at most 8 bits per element stored in the filter, then $z/n = 8$. If using a conventional filter, the optimal number of hash functions $k = 6$ and $f = 0.0216$. As shown in Table 3.1, if our device has sufficient memory for 14 bits per set element (75% more memory), then by using a compressed filter, the f can be reduced by almost 20% to 0.0177 and the number of hash functions reduced to two. As m increases and k decreases, the entropy of the uncompressed filter decreases. The result is a larger filter with a lower false positive rate that still compresses to at most 8 bits per element stored.

Table 3.1: **Transmission Size**: 8 bits per element (compressed). Derived from [102].

Array bits per element	m/n	8	14	92
Transmission bits per element	z/n	8	7.923	7.923
Hash functions	k	6	2	1
False positive probability	f	0.0216	0.0177	0.0108
Probability that a bit is 1	s	0.5277	0.133	0.0108
Probability that a bit is 0	p	0.4723	0.867	0.9892
Entropy	$H(p)$	0.9978	0.5656	0.0861

Figure 3.9 provides a graphical view of the information contained in Table 3.1. Both show the impact of m and k given n and limits on transmission size and the expected false positive rate.

3.5 Bitwise Operations on Bloom Filters

Standard Bloom filters contain one bit per cell, so they can only represent whether or not at least one element has hashed to a specific cell. Suppose filters F_A and F_B represent sets S_A and S_B. By the application of bitwise operations on their filters, we can approximate operations on sets S_A and S_B. Since we will be operating on filters, not the sets themselves, the operations are **not** equivalent to deterministic set operations

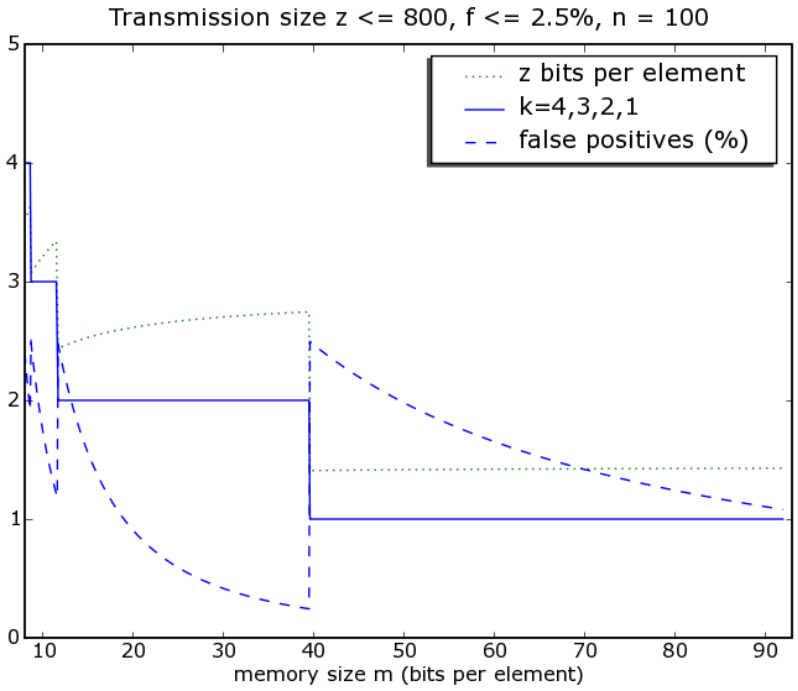


Figure 3.9: **Transmission Size**: Given n elements to be stored and a Bloom filter optimized for storage, allocating more memory m for storage reduces the false positive rate. Using fewer hash functions reduces transmission size.

union $\cup$, intersection $\cap$, and difference $\setminus$.

While bitwise operations OR, AND, and XOR are computationally efficient, they must be performed on standard filters that are of the same size and built with the same hash function(s).

Set Union: The union of two sets, $S_A \cup S_B$, can be approximated by

$$F_{OR} = F_A \; OR \; F_B$$

For example:

$$(1,1,1,0) = (1,1,0,0) \; OR \; (0,1,1,0)$$

This operation is particularly useful if the filters are sparse. For example, given two filters, their union could be stored in the root node of a subtree. If the root tests positive

for an element, then one of its child nodes may contain the element. This technique was used to implement *attenuated Bloom filters* [126], which are discussed in Section 4.5.1.

For other applications, a filter with a size that is a power of 2 can be compressed. Split the filter into two halves and OR them together. When testing for set membership, mask the highest order bit.

Set Intersection: The intersection of two sets, $S_A \cap S_B$, can be approximated by

$$F_{AND} = F_A \; AND \; F_B$$

For example:

$$(0, 1, 0, 0) = (1, 1, 0, 0) \; AND \; (0, 1, 1, 0)$$

Unlike set union, set difference may set some bits to zero and cause false negatives.

Set Difference: Bitwise operations may be used to approximate set difference, $S_B \setminus S_A$, but at the risk of causing false negatives. By setting some bits in filter F_B to zero, some elements in the true difference set may test negative when, in fact, those elements are not in S_A. However, if this operation is being used to determine elements in S_B but not in S_A, then this operation may, in effect, cause false positives.

$$F_{-} = F_B \; XOR \; (F_A \; AND \; F_B)$$

For example:

$$(1, 0, 0, 0) = (1, 1, 0, 0) \; XOR \; ((1, 1, 0, 0) \; AND \; (0, 1, 1, 0))$$

3.6 Bloom Filter Variants

Bloom filters are immensely extensible. The following variants of the Bloom filters have been developed to support specific functionality. Counting filters support insertion and deletion [33]. Attenuated filters store shortest path distance information

[126]. Compressed Bloom filters support message passing [102]. Spectral filters provide support for estimating element frequencies [22]. Bloomier filters provide support for arbitrary functions [20].

3.7 Conclusion

The hash table is a venerable data structure for constant time $O(1)$ lookup, but it may become impractical for very large amounts of data or where the keys are large. When a lookup hash table grows too big, the entire table cannot be kept in core, and even where some form of persistent storage is available, the performance hit may be too great.

The Bloom filter is a space efficient data structure that can do more because it does less. It can perform membership tests in just a fraction of the memory that would be needed to store a full list of keys. The cost is an adjustable rate of false positives, and a loss of information – the order in which a set of keys were hashed to the filter cannot be recovered, the set of keys hashed to the filter cannot be recovered, and a key cannot be removed.

In applications where these costs are acceptable, Bloom filters have been utilized to improve performance. In this thesis, we store sensor meta-data in Bloom filters and then use those filters to reduce communication costs.

Chapter 4

Applications of Bloom Filters

4.1 Introduction

Although Bloom filters were introduced in 1970 [11], and subsequently used in database applications, it was not until the 1990s that they became a significant subject of the networking literature [15]. Information theory provides a lower bound on the extent to which data can be compressed prior to storage or transmission. In other words, the information capacity of any medium has an inherent limit. This limit has motivated the use of Bloom filters in a variety of network applications: (1) distributed caching; (2) summarizing content to help collaborations in overlay and peer-to-peer networks; (3) resource routing: supporting a probabilistic algorithm for locating resources; (4) speed up or simplify packet routing protocols; (5) measure infrastructures to create data summaries in routers or other network devices.

We begin with some early applications before discussing more recent applications, and then we review wireless applications where Bloom filters were used to reduce storage and communication costs, and increase fault tolerance and security.

4.2 Early Applications

Bloom filters were originally used for hyphenation [11], spell-checking [97], and detecting unsuitable passwords [138, 93] where the fundamental task was to filter out nonmembers of a set. For these applications, a great majority of messages to be tested were **not** expected to belong to the given set, and reject time was considered to be a critical performance metric.

For his technique to be effective, Bloom assumed that processing for most inputs

would be very simple. Processing for a small number of cases (set members) would be more complex. If an input is recognized as a member, "it could then be subjected to a follow-up test to determine if it is actually a member of the minority set or an 'allowable error.' By allowing errors of this kind, the hash area may be made small enough for this procedure to be practical" [11].

4.3 Web Proxies

Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol by Fan (1998 and 2000) is one of the most cited papers on using Bloom filters in the Internet context [33]. The authors noted that caching has been recognized as an important technique for reducing Internet traffic, and that caching within Web proxies is very effective. However, gain the full benefit of this technique, proxy caches behind a common bottleneck link should cooperate. By serving each other's misses, they could further reducing the traffic through the bottleneck. The authors called this process *web cache sharing* [33].

As of 1998, sharing caches was not widely deployed due to the overhead of existing protocols. In [33] the authors describe a web cache sharing protocol called *Summary Cache*. Essentially, each proxy stores a summary of the cache directory of each proxy in the group, and checks each summary for a "hit" before sending a query to that proxy. A Bloom filter was used to represent each proxy's cache directory – referred to as a summary cache. [1] Each directory entry could be represented with just 8 bits. This reduced storage requirements and communication overhead.

The following illustrates cache sharing. Proxies cooperate by letting each other know what they have. On a cache miss a proxy, say myProxy, tries to determine if another proxy cache holds the desired Web page. If it does, then a GET request is sent to that proxy rather than to another proxy in the group. For myProxy to make the right decision it needs to know the contents of the other proxy caches. In *Summary Cache*, each proxy periodically broadcasts a Bloom filter that represents the contents of its cache. This means myProxy can test each proxy's Bloom filter until it finds one that

[1] In an application known as *Squid*, such Bloom filters were referred to as *Cache Digests* [129].

tests positive for the URL in question. In the case of a false positive, that proxy will reply that it does not have the requested page, and myProxy will test the other filters. While false positives introduce some delay in the form of a wasted query message, that cost is outweighed by a significant reduction in network traffic.

Performance was shown to be substantially better than for a traditional per request query/reply scheme such as Internet Cache Protocol (ICP). Summary Cache reduced the number of protocol messages by a factor of 25 to 60, reduced bandwidth consumption by over 50%, and reduced CPU overhead by from 30% to 95% [33].

4.4 Traffic Measurement

Per-flow traffic measurement is critical for usage accounting, traffic engineering, and anomaly detection. Previous methodologies are either based on random sampling, or only account for the "elephants". The Space Code Bloom Filter (SCBF) is an approximate representation of a multiset where each element is a traffic flow and its multiplicity is the number of packets in the flow. Through parameter tuning, SCBF allows for graceful trade-off between measurement accuracy and computational and storage complexity [80].

4.5 Peer-to-Peer Applications

In peer-to-peer applications, Bloom filters have been used for routing queries, and meeting space, communication, and privacy constraints.

4.5.1 Probabilistic Location and Routing

The Internet's storage capacity and bandwidth have grown dramatically. Some systems aim to provide cheap, highly available storage without centralized servers. Their designers planned to achieve availability and durability in spite of component failures through replication and coding techniques [126]. In P2P systems it is often necessary to discover the nearest node that holds a file and the route to reach it.

In [126] the authors presented a probabilistic location and routing algorithm that was designed to improve the performance of existing deterministic wide area location mechanisms, i.e., to improve latency in the case where a nearby replica exists. They were motivated by two architectural challenges to increasing availability and durability. First, if replicas may be placed anywhere in the system, how should they be found? Second, given the location of one or more replicas, how should queries be routed to them [126]?

They determined that they could formulate these two operations as a single *location and routing* problem that routes queries from a client to the closest replica with certain properties, for example, the replica with the shortest network path to the client, or the replica residing on the least loaded server. Although their algorithm is probabilistic, **it will succeed or fail quickly** – and if it does fail, then a deterministic algorithm will be invoked so that every document will be found even when replicas are scarce. Conceptually, the trade-off is exactly the same as in classic Bloom filter applications. The authors combined a set a Bloom filters into a compound, topology-aware data structure, which they called an *attenuated Bloom filter* (ABF). An ABF is an array of d Bloom filters, and an ABF of depth d can store information about servers that are i hops away where $i \in \{1..d\}$.

Rhea and Kubiatowicz also explained how the ABFs are updated each time a new data item is added to a server. Essentially, the update is propagated to every server within d hops – like a circular wave spreading out from the source of change. This algorithm was implemented to prevent redundant updates. Figure 16 of their paper's Appendix describes the complete algorithm [126].

4.5.2 Sharing Information

Bloom filters have a great deal of potential for distributed protocols where systems need to share information about what data they have in storage. This implies that Bloom filters play two roles: as a data structure that supports set membership queries and as a message that is transmitted between subsystems. In Compressed Bloom Filters, Mitzenmacher observed: "Transmission size may be of greater importance when the

amount of network traffic is a concern but there is memory available at the endpoint machines" [102].

4.5.3 Self Organization in P2P Systems

Ledlie et al. addressed the issue of dependability and the problem of group formation in peer-to-peer (P2P) systems [82]. Participants may form a local picture of the global state, but this state may not remain stable if nodes enter and leave the system at a high rate. They argued that decentralized systems must explicitly delineate their information dispersal mechanisms, assumed node capabilities, and distribution of information demands.

How should *dependability* be defined in a distributed system? Suppose the type of information exchanged is event-driven. For example, where one node needs to notify another node or collection of nodes about an abrupt temperature change or that a bridge is in danger of imminent collapse. A system where every node has an up to date picture of the entire system may not be feasible.

When designing a dependable system, several components must be considered. What type of information will be exchanged? What are the capabilities of the nodes? What is the distribution of information and requests across the nodes?

The authors divided information exchange into five categories. Most P2P designs focus on **probe queries**, which test for the existence of an object. In contrast, sensor network systems fall into one of four categories: **event-driven point-to-point**: a node registers an interest in an event such as an abrupt temperature change; **event-driven broadcast**: broadcast from one to all, e.g., a software update; **continuous stream point-to-point**: provides a path for streaming data for unknown duration to a node or nodes, e.g., Internet routing; **continuous stream broadcast**: one node continuously updates the entire system [82].

To investigate the issue of dependability, a simulator for a hierarchical grouping scheme was designed and implemented for P2P and sensor network systems. A system consists of many hierarchical groups, each shaped as a tree, and every group has a root node. The summary for each group was represented by a Bloom filter. Each root node

was tasked with:

- Calculating a summary of all objects in the group.

- Maintaining summaries for each of its immediate children (which in turn maintain summaries for their children).

- Directing searches of the group.

A root's workload increases with the size of its group. At some point it must partition the group, i.e., when the root becomes overloaded, it sheds load by partitioning the group. Partitioning and responding to requests to join the group [2] provide dynamism and self-configurability.

The use of Bloom filter summaries served as an efficient mechanism to prune the search space [82]. Essentially, the authors found that search properties of a tree data structure could be combined with two important properties of Bloom filters. They are extremely space efficient and the OR operation can be performed on two filters of the same size.

Given the number of objects in a tree n, the root node can estimate filter size m such that about 50% of bits are set to 1. Each leaf node hashes its objects into its filter and passes the filter up to its parent, which ORs the filters. That parent then passes the ORed filter up to its parent, and so on. This hierarchy of filters means that more bits are set at higher levels than lower levels. At lower levels, filters are more sparse and therefore more accurate and more compressible. This information hierarchy could be used from used both outside the group to determine whether to contact the root and within the group to guide search queries between the nodes [82].

[2] Ideally, group membership is based on low intra-group latency, varied mean time to failure (MTTF), and heterogeneous bandwidth.

4.6 Wireless Applications

4.6.1 Data-Centric Storage

In a sensor network we may be more concerned about the data collected by a node than the identity or IP address of the node. *Data-centric* models have been proposed where the data is *named* based on event type or geographic location. One interesting benefit is that data-centric routing and data-centric storage (DCS) are energy efficient [39].

In the DCS model, all data of a certain event type (temperature, humidity, or pressure, etc.) is stored at the same node, and a query can be sent directly to the relevant node instead of flooding the network. The implementation of DCS uses GPSR geographic routing [72] and peer-to-peer lookup algorithms such as CAN [119], Chord [140], Pastry [127], and Tapestry [8]. This model has been shown to reduce total network load and concentrated traffic at bottlenecks ("hotspots") [120, 136]. In [33] it was shown that Bloom filters offer an efficient way to support attribute based queries.

To reduce data retrieval traffic and increase resilience to node failures, Ghose et al. [39] proposed replication of control and data information in a DCS framework, which they called Resilient Data-Centric Storage (R-DCS). Their system partitions the coordinate space of the network into Z zones. Each zone has a **monitor node** for each event type. These nodes store and exchange information in the form of a *monitoring map* for each event type. This map includes Bloom filters for enabling attribute based queries – these attributes could be *Event Time* and *Temperature*.

The authors in [39] considered a temperature monitoring application where Bloom filters were used to represent which replica nodes contain temperature readings in certain temperature ranges. For example, suppose a node sends the query "Get all event data for temperature readings between 30 and 40" [39]. Bloom filter-based matching would generate a list of replica nodes, and then forward the query to a restricted set of replica nodes – instead of flooding to all replica nodes. However, it was noted that this use of Bloom filters is cost effective only when there are a large number of replica nodes.

4.6.2 Security

Large scale sensor networks tend to be unguarded and vulnerable to denial-of-service attacks in the form of false data reports which cause false alarms, excessive power consumption, and congestion of wireless channels. Such attacks may come from outside the system or from a compromised node within the system. However, providing security in this context is a challenge because a public-key based authentication mechanism based on asymmetric cryptography would exceed the computation and storage constraints of a small sensor node. Statistical En-route Filtering (SEF) [162] was proposed as a mechanism to detect and reject reports from a compromised node.

The Statistical En-route Filtering mechanism (SEF) [162] exploits the properties of a large scale, high density sensor network. Here a large scale network is defined as one where most reports must travel at least 10 hops before reaching a data collection center ("sink"). High density implies that each event ("stimulus") can be detected by multiple sensors and, therefore, SEF relies on the collective decisions of a large number of sensors to detect a false report.

Upon detecting an event, multiple sensors generate a report that carries multiple message authentication codes (MACs) "stored" in a Bloom filter. The report is tested at each node before it is forwarded. The probability of detecting incorrect MACs increases with the number of hops required to get to the sink. The authors found that SEF can dropped up to 90% of false reports within 10 hops and compared to using a list of MACs, using a Bloom filter may reduce overhead by about 70% [162].

In [88], the authors presented MiniSec, a secure network layer that provides low energy consumption and high security. For multi-source broadcast communication (MiniSec-B), Bloom filters were used to track packet history for each epoch and recognize replayed packets. Bloom filters were deemed well suited to the resource constrained environment of sensor networks because of their space and time advantages: $O(n \log n)$ and $O(1)$ respectively.

In [96], the authors presented their protocol for the detection of denial-of-message attacks on sensor network broadcasts. Bloom filters were used to represent a set of nodes, specified by the base station, which must produce an acknowledgment. Al-

though each node receives a copy of the Bloom filter, the set of nodes remains unknown. This ensures that adversarial nodes, eavesdropping on broadcast messages, will not able to selectively forward broadcast messages only to the nodes in the set.

4.6.3 Network Monitoring

In [98], the authors addressed the challenge of monitoring large scale sensor networks which generate spatially correlated data. Their proposed solution is based on contour maps and, specifically, *event contours*. In cartography, a contour is a line in a map that connects points of equal value. These lines, separated by a threshold difference, represent various events such as temperature, altitude, concentration, velocity, et cetera. In this case, it is sufficient if only a subset of sensors report their data to the sink.

They proposed several algorithms. A distributed spatial and temporal data suppression algorithm, and a reconstruction algorithm at the sink that uses interpolation and smoothing. Of particular relevance to this thesis is an algorithm that conveys routing information to enable multi-hop local suppression. It is based on the use of Bloom filters and the knowledge of sensor locations at the sink.

The local suppression algorithm that is based on each node's ability to overhear its neighbors. To enable an even greater trade-off between accuracy and communication cost, suppression must be based on overhearing sensors that are multiple hops away. If sensor u overhears a report that is up to h hops away (a field indicating the number of hops the report has traveled is included in the report), and the report has a value that is within δ of the measurement taken by sensor u, u will not transmit its measurement. However, in order for the sink to interpolate suppressed values, the sink must be able to determine the path taken by each packet received from the contents of each packet.

The authors proposed a Bloom filter technique. The data source node creates a filter and adds it to the packet. Each node on the route "stores" its node ID to the filter, so the filter represents a subset s of the set S of all node IDs. When the data sink receives the packet, it extracts the number of hops h from the report, and it uses the same hash functions to test the elements of S for membership in the subset s. If an element tests positive, then it is very likely that that node is on the route. However, if a node in s is

too far away, the sink will know it's a false positive.[3] After some false positives have been removed from s to yield $s\prime$, the sink can reconstruct the route from $s\prime$ [98].

Given a maximum path length of $n = 20$, $k = 1$ hash functions, and a $m = 32$ bit Bloom filter, it was calculated that all false positive paths of $n = 10$ or more hops could be eliminated with high probability [98].

4.6.4 Reducing Routing Overhead

Ad hoc networks are wireless networks of mobile nodes which have limited power and transmission range. Due to their lack of a fixed infrastructure, each node also acts as a relay to provide communication to other nodes.

However, many ad hoc routing protocols assume that there is always a connected path from source to destination. If the network suffers a temporary partition between source and destination, which is likely in short-range wireless networks, packets will be lost. Consequently, the authors in [148] developed *Epidemic Routing* to deliver messages even in a network where there is never a connected path from source to destination. This protocol provides routing in partially connected ad hoc networks by ensuring the eventual delivery of messages via random pair-wise exchanges of messages among mobile hosts ("carriers").

Vahdat and Becker tested their protocol in a simulator. Each host maintained a buffer of messages which originated from itself or another host. The messages were stored in a hash table, keyed by a unique identifier for each message. While not utilized here, the authors noted that a Bloom filter would substantially reduce the space overhead [148]. When two hosts come into communication range, the host with a smaller identifier initiates an *anti-entropy session*. [4] Each host maintains a cache of hosts that it has had a session with recently. A Bloom filter representing recent hosts could save space here also.

In the context of connected networks, Helmy et al. [58] considered the problem that while many routing protocols for ad hoc networks are designed to find the optimal or

[3] Assumptions: the topology is fixed and the sink knows the location of each node in the network.

[4] During anti-entropy, the two hosts exchange information to determine which messages stored remotely have not been seen by the local host. Each host requests unseen messages.

shortest path, but they tend to incur significant route discovery overhead. For dynamic systems where small amounts of data are transferred and connections may be very short in duration, such routing protocols are too costly.

The authors in [58] focused on reducing the overhead of resource (or route) discovery for short flows because nodes in ad hoc networks are usually portable devices with limited battery power. To save power they determined that the resource discovery mechanism should be efficient in terms of communication overhead. Hence, they designed an architecture to for power-efficient resource discovery and small data transfers in large-scale ad hoc networks.

They implemented *contacts* which act as short cuts in a highly clustered multi-hop wireless network [58]. The idea of using contacts was first introduced in [56], and the initial work on the CARD architecture was presented in [57]. This architecture is based on the concept of *small worlds* – the addition of a small number of short cuts results in a significant reduction in average path length (or degrees of separation). Each node stores a path to several nodes (contacts) beyond its vicinity. Each path, which is a list of edges, was stored in a Bloom and added a Contact Selection message [58].

4.7 Conclusion

Bloom filters were originally used in Unix spell-checkers [97], and for detecting unsuitable passwords [138, 93]. They were subsequently found to be useful in databases to speed up semi-join operations [83, 107].

More recently they have been used in many Internet applications [15]. Two important examples are cooperating web proxies and peer-to-peer (P2P) systems. In [33] the authors describe a web cache sharing protocol called *Summary Cache*. In [126] the authors presented a probabilistic location and routing algorithm that was designed to improve the performance of existing deterministic wide area location mechanisms, i.e., to improve latency in the case where a nearby replica exists. Their results indicate that using Bloom filters local decision making should deliver a net performance gain for the system.

Bloom filters have been applied in many resource constrained domains where their limitations were acceptable. In this thesis, we exploit the properties of Bloom filters to support WSNs where resources are even more constrained than has been the case for past applications.

Chapter 5

Distributed Asynchronous Clustering

5.1 Introduction

Distributed Asynchronous Clustering (DAC) is a hierarchical clustering protocol designed for a large scale Wireless Sensor Network (WSN) of thousands of power constrained data sources, one base station ("sink"), and a data gathering application that continuously monitors the network for an aggregate value such as the mean temperature recorded by all sensors. Sensor network design may be influenced by many factors [2], and each device may differ in terms of its capabilities [128]. However, the fundamental challenge is to reduce the energy consumed by radio communication such that network lifetime is maximized given the application's quality of service (QoS) guarantees.

The Distributed Asynchronous Clustering (DAC) protocol provides an effective, low cost solution to an essential problem: how to generate a near optimal number of well separated cluster heads in a wireless deployment. DAC solves this problem by using local knowledge of the network and its environment to distribute cluster heads and terminate their generation. As a result, this protocol has many desirable properties. *Fully distributed:* Autonomous cluster heads use a limited range radio beacon to recruit members. By utilizing local information, such as the relative strength of radio signals from nearby cluster heads, sensors are able to make decisions that move network topology toward a global optimum. Such decentralized systems are more scalable and more robust against individual node or link failures [32]. *Adaptive:* The probability of a node volunteering for cluster head duty depends on its power level, its beacon range may depend of factors such as network density, and the shape of each cluster depends on the radio propagation environment. *Self-terminating:* It stops after all nodes have

decided to be either a cluster head or a cluster member. *Low overhead:* Clustering requires minimal communication of infrastructure data.

We implemented a simulator to model the impact of clustering on communication costs. In particular, we modeled the impact of clustering on transmission distance for the benchmark task. We assumed that a protocol that reduced distance more effectively in our simulator would also reduce the energy used by radio communication more effectively in a real deployment.

We compared the performance of DAC with another low overhead protocol, LEACH [55, 53], and a hypothetical k-means [69, 27] protocol. We found that DAC required fewer cluster heads to cluster all sensors, and generated a superior topology where connectivity provided a far greater reduction in transmission distance.

The primary advantages/characteristics of this protocol are: (1) *pre-emptive recruitment* promotes well separated cluster heads; (2) the number of cluster heads can be optimized; (3) it provides a solution that is fully distributed, scalable, adaptive, and robust; (4) facilitates deployment of devices – especially over irregular terrain; and (5) provides substantial performance advantages over existing protocols.

In Section 5.2 we discuss clustering for self-organization and in-network processing. In Section 5.3 we present results from our simulations. In Section 5.4 we present related work. And finally, our conclusions.

5.2 Hierarchical Clustering for In-network Processing and Routing

A WSN is a *complex system* composed of a large number of nodes and their interactions and, consequently, many researchers have taken a hierarchical clustering approach to self-organizing sensor networks. Hierarchical architectures may take advantage of device heterogeneity and create opportunities to exploit the computation - communication trade-off. For the energy cost of transmitting 1Kb a distance of 100 meters, a general purpose processor with 100MIPS/W power could execute ≈ 3 million instructions [118]. As the scale of a deployment increases, it becomes more important for

network topology to provide well separated nodes that facilitate in-network processing and reduce transmission distance.

5.2.1 Hierarchical Clustering

For large scale deployments, where fine grained sensing covers a wide area, networks will include tens of thousands or even millions of nodes. "Protocols will have to be inherently distributed, involving localized communication, and sensor networks must utilize hierarchical architectures in order to provide such scalability" [77]. Indeed, hierarchical clustering and in-network processing are well known and scalable techniques for reducing communication costs. They may reduce transmission distance and the amount of data transmitted. In general, motes are partitioned into a set of clusters where each sensor may only transmit data to its cluster head. Cluster heads process member data and forward the result to the base station. However, existing clustering protocols tend to suffer from limitations related to latency, overhead, centralization, reliance on global knowledge of the network, lack of scalability, and suboptimal numbers of poorly distributed cluster heads.

5.2.2 In-network Processing

One of the key motivations for clustering protocols such as DAC is to facilitate in-network processing, i.e., perform computation, such as data aggregation, as close to the source as possible to reduce the amount of data transmitted. Further, some applications will require that network nodes process data cooperatively and combine data from multiple sensor types. Transmitting raw data from all sensor nodes to a base station does not scale well because per node throughput scales as $\frac{1}{\sqrt{N}}$ – it goes to zero as the number of nodes N increases [43]. In other words, as N increases every node spends more time forwarding packets of other nodes.

In the next section we provide a summary of LEACH [55, 53] before presenting DAC because it highlights several important problems and is most comparable to our protocol.

5.2.3 LEACH Clustering Protocol

Low-energy adaptive clustering hierarchy (LEACH) is an application specific protocol architecture for homogeneous WSNs [55, 53]. To develop their clustering protocol they assumed that all nodes are powerful enough to transmit directly to the base station when necessary. Their network self-organizes into clusters using a semi-distributed algorithm. Individual nodes use global knowledge of the network to make autonomous decisions about whether or not to volunteer for cluster head duty and which cluster to join without any centralized control. The probabilities of volunteering during set-up are intended to generate an optimal number of cluster heads.

At the beginning of self-organization, nodes may autonomously volunteer for the role of cluster head. The probability P_i of sensor S_i volunteering depends on its energy level and whether or not it has volunteered in any of the previous $(r \ mod \ (N/C))$ rounds. The expected number of cluster heads C depends on the number of nodes N and the set of probabilities P at time t [53]. However, the probability P_i does not depend the decisions made recently by other sensors in the network.

After all nodes have decided their role, cluster heads broadcast an advertisement message (beacon) to the WSN using a Carrier Sense Multiple Access (CSMA) protocol. According to their simulations, the range of this beacon is **not limited** [55, 53]. Free sensors associate with the cluster head that requires minimum communication energy – the closest based on *received signal strength*.

5.2.4 DAC Clustering Protocol

The design of the DAC protocol is based on two observations: (1) a cluster head's beacon range and the number of clusters formed are very closely related; (2) multiple proximate nodes should not volunteer for CH duty. However, many comparable protocols [53, 86, 23, 95, 94, 106, 163] rely primarily on residual energy to influence the probability of a node volunteering for cluster head duty, ignore decisions made by proximate nodes, rely to some degree on global knowledge of the network, and fail to adapt to local conditions in the environment.

DAC was designed such that nodes may adapt to local environmental conditions. In DAC, whether or not a free sensor volunteers depends on nearby events and its capacity. All else being equal, higher capacity nodes are more likely to volunteer for cluster head duty.

After a sensor volunteers for cluster head duty, it immediately broadcasts a beacon to recruit nearby sensors. All sensors that hear the beacon are no longer free to volunteer – they must wait for a period of time, then associate with the closest cluster head. The shape and number of clusters formed may be strongly influenced by the radio propagation environment. Clustered sensors are not be free to re-associate until their cluster head fails or resigns.

DAC relies primarily on the strength of a cluster head's beacon and *pre-emptive recruiting* rather than the probability of a sensor volunteering, to generate a near optimal set of *well separated* cluster heads. According to *pre-emptive recruiting*, when a sensor volunteers for cluster head duty, it transmits a limited range beacon and recruits neighbors as soon as possible to prevent them from volunteering. In contrast, we would say that LEACH uses *non pre-emptive recruiting*. These differences enable DAC to consistently outperform LEACH without incurring more overhead.

As the pseudo code in Algorithm 1 shows, during the DAC set-up phase, free sensors volunteer to be cluster heads with a certain probability, each new cluster head emits a beacon according to CSMA, free sensors within beacon range are "recruited" and set themselves to *not free*, record the received signal strength (RSS) and identity of each beacon heard. Given enough time, every sensor will hear a beacon or volunteer. After a waiting period, during which no more beacons are heard, each sensor associates with the nearest cluster head based on RSS.

Fault tolerance may be provided by a variation of the DAC set-up phase. When a cluster head fails, the sensors associated with it will gradually become aware that their cluster head no longer transmits and change their status from clustered to free. After a waiting period, a qualified free sensor will volunteer to be a cluster head, broadcast a beacon, and free sensors within range will associate with it. Over time the network will degrade gracefully, reporting less detailed information about its environment.

Algorithm 1: Pseudo code for pre-emptive recruiting, which runs on each sensor node during set-up phase of DAC. Free sensors volunteer to act as cluster heads with a small probability p and transmit a beacon.

```
initialization;
while sensor.isFree do
    sensor.listen();
    if sensor.hearsBeacon( ) then
        sensor.isFree ← false;
        sensor.recordRSS( );
    else if sensor.random(0..1) p then
        sensor.isFree ← false;
        sensor.isClusterHead ← true;
        sensor.setBeaconRange( );
        sensor.broadcastBeacon( );
    end
end
sensor.listen(time_period) ;
clusterHead ← sensor.maxRSS( ) ;
sensor.associate(clusterHead) ;
```

5.2.5 Deployment

DAC facilitates deployment of devices in several ways. First, a large number of low cost homogeneous sensors may be deployed in an approximate manner over an irregular terrain. Clusters should adapt naturally to the radio propagation contours of the environment. Second, fault tolerance reduces the need for maintenance and deployment of new devices. Third, some applications will require greater in-network processing and/or longevity. *We can use the sensor network itself to facilitate an upgrade.* This insight will be reinvoked in Chapter 6 where cluster heads, which are relatively few in number and well separated, are automatically located and replaced by more powerful devices to facilitate small-world properties [59].

5.3 Simulation and Results

We compared the performance of LEACH, DAC, and a hypothetical k-means clustering protocol. The most important metric was the *total transmission distance* required for

our benchmark task after clustering. In our simulations, each protocol was subjected to the same constraints. These included a limited cluster head beacon range and a limited number of cluster heads C.

We also investigated the relative importance of the following network parameters: radio model, base station location, and beacon range. However, we tried to keep our model as simple as possible [149]. Our simulations were intended to present a simplified representation of reality – just enough detail to unambiguously reveal the relative impact of three clustering protocols on total transmission distance. We used simple models, as recommended in [67], to provide a basic proof of concept.

5.3.1 Network Model

In most simulations the network was composed of 875 sensors deployed approximately 30 units apart on a grid of 25 rows and 35 columns. Each sensor's x, y coordinates were perturbed slightly by a small Gaussian. The base station was located at $x = -50$, $y = -50$. Cluster head radio beacon range was limited to a Euclidean distance d of 150 units (distance is relative). The path loss exponent n depended on assumptions about distance and the environment.

Radio communication is highly variable and difficult to model [53]. However, since quantitative estimates of packet loss and expected network lifetime for a deployment of specific radio propagation environments and hardware are outside the scope of this paper, we made a number of simplifying assumptions. To compare protocols, each node was assumed capable of direct communication: sensor to the nearest cluster head within range and cluster head to base station. In the case of unclustered sensors, the cost of direct communication to the base station was, in effect, a penalty.

5.3.2 Radio Model

The strength of a transmitted signal decreases in proportion to d^n, where d is distance and n is the path loss exponent. Depending on multi-path and other interference, n is typically in the range of 2 to 5 [32]. The actual value strongly depends on the radio

propagation environment [104].

We assumed a simple radio model where the radio consumes a fixed amount of energy per bit to run the transmitter or receiver circuitry, and a variable amount per bit to run the transmit amplifier. In LEACH [55], the variable amount is proportional to transmission distance d^2. In their later work [53], the variable amount is different for intra-cluster communication and base station communication, d^2 and d^4 respectively. However, in both [55] and [53], it was found that approximately 5% of sensors should be cluster heads. If communication with the base station consumes energy proportional to d^4 instead of d^2, then this should reduce the optimal number of cluster heads.

So we tested LEACH using different radio models and unlimited beacon range. For d^1 path loss, the optimal percent of cluster heads was $\approx 5\%$. For d^2 path loss, the optimal percent of cluster heads was $\approx 2.5\%$. However, in other simulations where we assumed a higher path loss exponent for cluster head to base station communication, such as the d^2, d^4 model used in [53], **one cluster head** was optimal, i.e., one cluster head minimized clustered network distance. This indicates that it is not possible to generate useful results where unlimited beacon range is assumed.

These preliminary tests suggested several implications. The optimal number of cluster heads is not necessarily $\approx 5\%$, rather it depends on the radio model, path loss exponents in particular, and the beacon range. Given a limited beacon range, the optimal number of cluster heads should be inferred by the network itself.

5.3.3 Cluster Quality

In our simulated network, an ideal set of clusters would have a uniformly and evenly distributed set of cluster heads. Figure 5.1 shows a representative set of poorly separated LEACH cluster heads. Large numbers of sensors (small black dots) have not associated with a CH because they are out of beacon range Many cluster heads (large white squares) are much too close together to be fully utilized.

LEACH clusters tend to be highly variable in terms of the number of clusters, cluster size, the spatial distribution of cluster heads, and the percent of sensors clustered. Figure 5.1 shows the most obvious problem: cluster heads are not well separated and

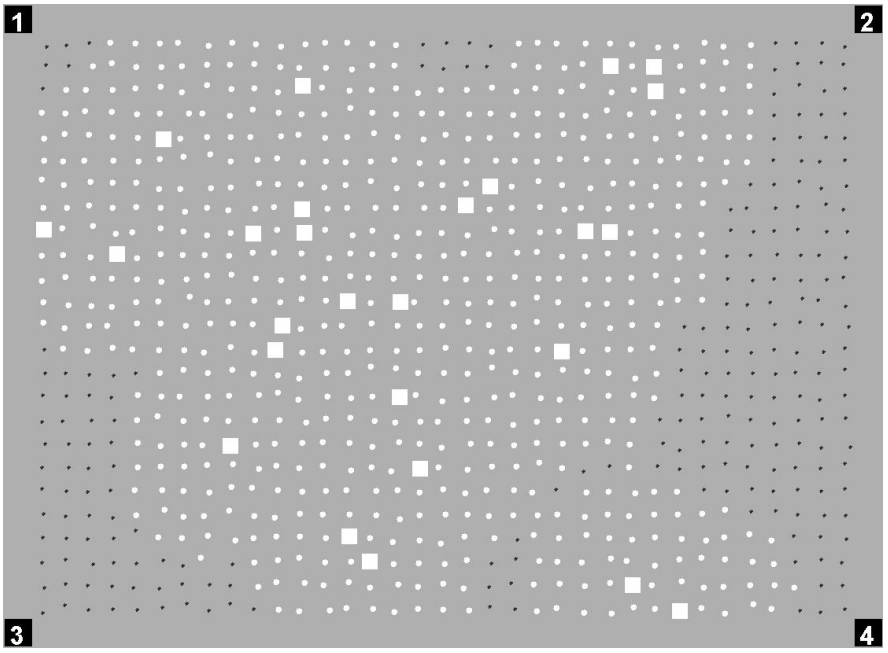


Figure 5.1: **LEACH Clusters:** 26 randomly distributed cluster heads, 74% clustered. Sensors are small dots and CHs are large squares. Clustered sensors are white. Unclustered sensors are black. (Black squares numbered 1 to 4 are sinks.)

many sensors are not clustered.

DAC clusters tend to be consistent in terms of the number of clusters, cluster size, the spatial distribution of cluster heads, and the percent of sensors clustered. As Figure 5.2 shows, the cluster heads are evenly distributed, and all sensors are within cluster head beacon range. We compared the distance between each cluster head and the nearest adjacent cluster head for each protocol. The minimum distance between a DAC cluster head and its closest neighbor cluster head was consistently much larger.

5.3.4 Number of Cluster Heads and Coverage

The desired number of cluster heads C must be known *a priori* for both LEACH and k-means in order to calculate the probability of a sensor volunteering. In contrast, DAC relies primarily on beacon range R in its attempt to generate an optimal number of clusters – it does not require *a priori* knowledge of the network. This difference means that DAC stops generating new clusters automatically. In contrast, LEACH can easily generate too few or far too many clusters because its sensors do not use

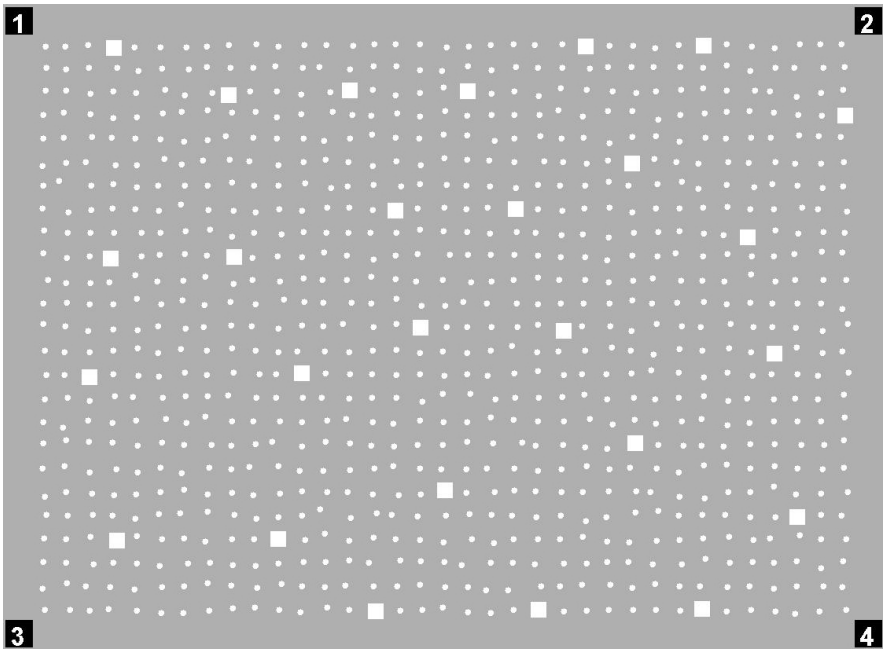


Figure 5.2: **DAC Clusters:** 26 evenly distributed cluster heads, 100% clustered. Sensors are dots and CHs are squares. (Black squares numbered 1 to 4 are sinks.)

local information, i.e., whether or not a nearby sensor has already volunteered. If too few cluster heads are generated, then some sensors will not be clustered ("orphans") because they are out of beacon range. If too many, then channel contention is increased unnecessarily.

All else being equal, a smaller number of cluster heads is preferred because limited wireless channel bandwidth must be shared by all sensors and cluster heads in the network. After clusters have been formed, each cluster head creates a TDMA schedule which tells its members when they can transmit. To reduce inter-cluster interference, each cluster communicates using direct-sequence spread spectrum (DSSS), also known as direct sequence code division multiple access (DS-CDMA), where each cluster communicates using a different CDMA code. All nodes will receive better communication channels when there are fewer clusters [53].

As the chart in Figure 5.3 shows, DAC consistently clustered more sensors with a limited number of cluster heads than LEACH. DAC is likely to cluster nearly 100% of sensors with only 28 cluster heads. In contrast, LEACH's clustered only 88%. and required at least 60 cluster heads to cluster nearly all sensors consistently. The implication is fairly obvious. If a protocol elects cluster heads at random locations, then

clustering the last few sensor nodes will require a disproportionate number of volunteers for CH duty in order to get them where needed.

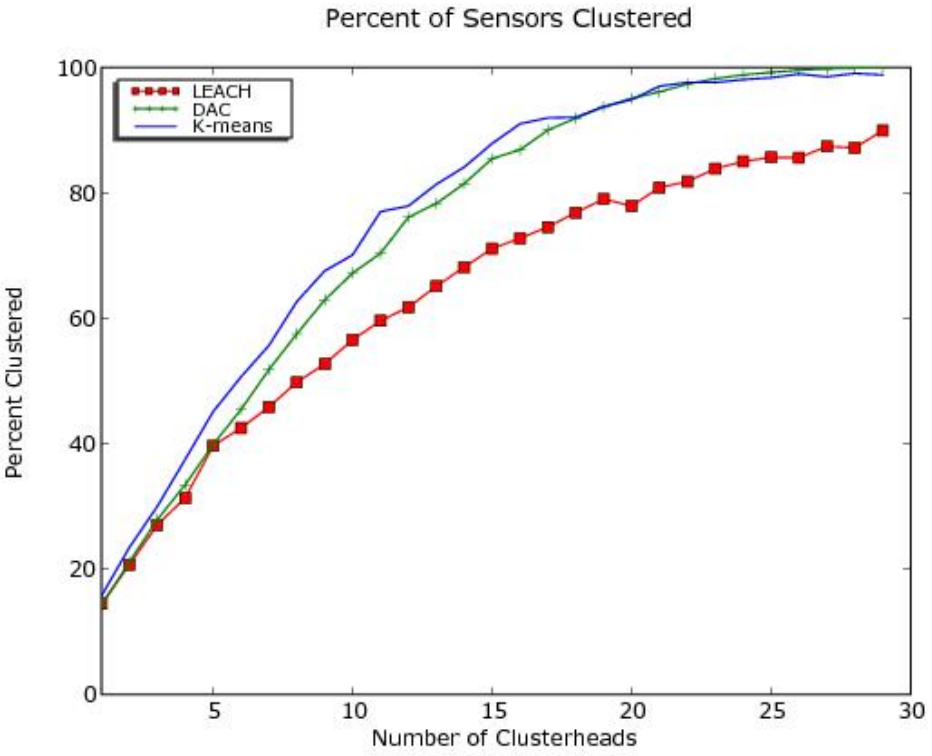


Figure 5.3: Percent of sensors clustered by three protocols. DAC clusters 100% of sensors with less cluster heads than LEACH and k-means.

5.3.5 Metrics

We simulated a *data gathering network* where all nodes sense their environment at a steady rate and report their data periodically. The benchmark task was to determine the average sensor measurement. For this task, the communication cost of delivering data to the base station dominates. Since the energy consumed by radio communication can be approximated by a function of Euclidean distance d, we used d^n as a metric for evaluating clustering protocols.

We defined *clustered network distance* (CND) as the sum of d^n distances: each clustered sensor to its cluster head, each unclustered sensor to the base station, and each cluster head to the base station. *Maximum network distance* (MND) was defined as the case where each of the N sensors must transmit raw data directly to the base station :

$\sum_{i=1}^{N} d_i^n$. For each protocol we calculated clustered network distance as a percent of maximum network distance: $p_{max} = 100 * CND/MND$. As an indication of relative expected energy consumption we used the ratio of $LEACH.p_{max}/DAC.p_{max}$.

5.3.6 Clustered Network Distance and Radio Model

An estimate of energy consumption depends on the radio model, transmission distance, and the path loss exponent n in particular. So we ran simulations given $C = 28$ and different values for n to determine its relative impact on the clustering protocols.

First we assumed that all communications were over similar distances under similar conditions. As n increased, the expected relative energy consumption of LEACH versus DAC increased. As shown in Figure 5.4, from $n = 1$ to $n = 5$, DAC was 1.9, 3.9, 4.8, 4.9, and 5.5 times better respectively. Second we assumed that intra-cluster communications were over relatively short distances and under ideal conditions, d^2 power loss, and base station communications were over relatively long distances and subject to interference, d^4 power loss. According to this model, and given our parameters, a DAC network would be expected to last ≈ 5 times longer than LEACH.

5.3.7 Clustered Network Distance

The clusters generated by each protocol are intended to reduce communication costs by reducing the total transmission distance for our benchmark task, which is to compute the average sensor measurement at the base station. If a sensor is associated with a cluster head, then it only needs to transmit a short distance to its cluster head - where its measurement will be aggregated with other measurements and forwarded to the base station. Our metric for comparing protocols in this context was clustered network distance expressed as a percent of maximum network distance.

As Figure 5.5 shows, DAC reduced clustered network distance by a greater amount than LEACH – especially in the $2 - 4\%$ cluster heads range. Assuming d^2 path loss, clustered network distance decreased rapidly as the number of cluster heads increased from zero to 28 or $\approx 3\%$ of sensors. DAC was expected to cluster all sensors with

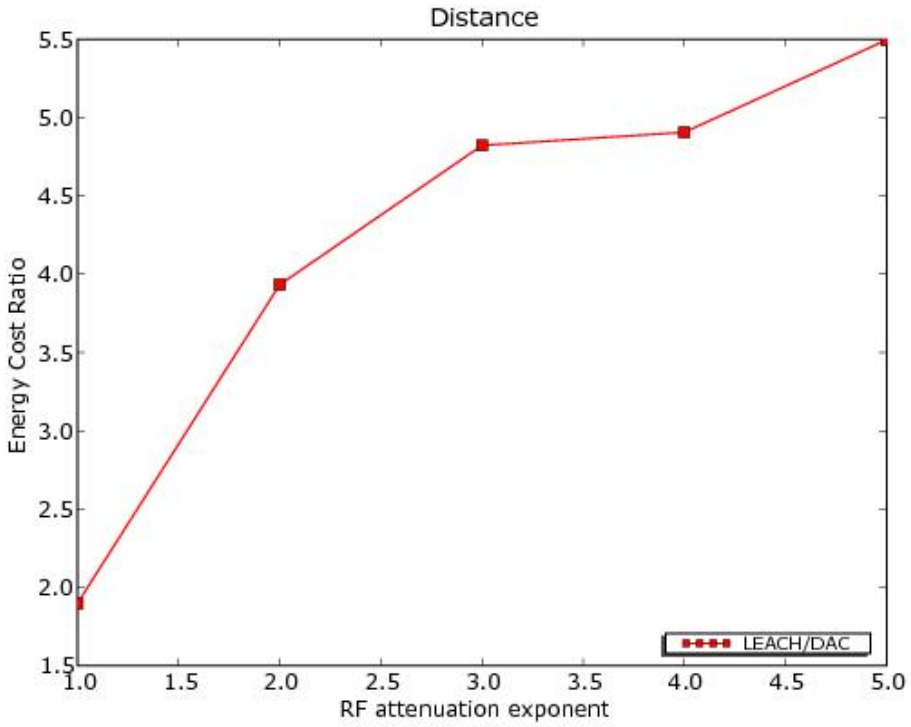


Figure 5.4: Expected energy consumption ratio: as the path loss exponent n increases, the relative energy consumption of LEACH versus DAC is expected to increase.

about 3% cluster heads, and since *no sensors were free to volunteer at that point*, the number of cluster heads was limited. However, according to LEACH all sensors decide whether or not to volunteer first, which means any number could decide to volunteer.

Although a hypothetical k-means clustering protocol would require far more communication overhead than DAC, it did not perform better in our simulations. Based on 100 trials using DAC needed an average of 27.37 cluster heads to cluster 100% of sensors, so we allowed k-means to use 28 (a slight advantage). Assuming d^2 path loss, network distance after clustering, as a percent of maximum network distance, was 4.17% for DAC, better than the 5.05% for k-means.

Furthermore, even when LEACH was allowed to generate an optimal number of cluster heads according to its protocol, more than twice as many cluster heads as for DAC, it was still less effective. Based on 100 trials and d^2 path loss, as shown in Figure 5.5, LEACH required 7.4% cluster heads to cluster 98.61% of sensors and reduced network distance to 9.29%. DAC required about 3.1% cluster heads to reduce clustered

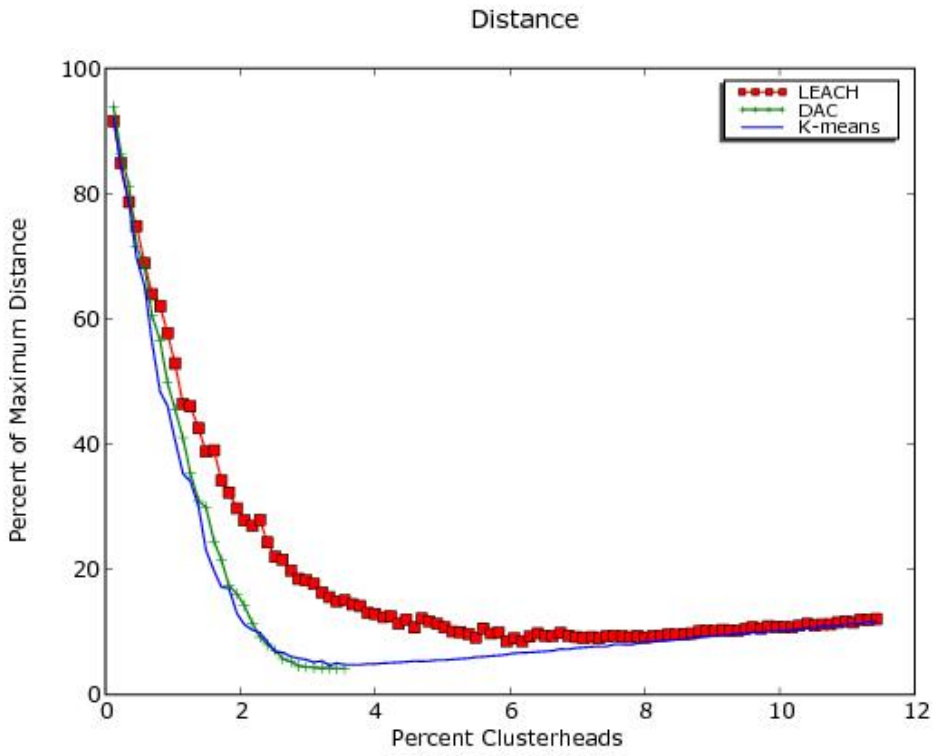


Figure 5.5: Minimizing clustered network distance: Given a 150 beacon range and d^2 radio attenuation, DAC required $\approx 3\%$ cluster heads, LEACH $\approx 6\%$.

network distance to 4.19% , still much better than LEACH.

Table 5.1 provides a qualitative summary of the clustering protocols and their relative effectiveness.

5.3.8 Number of Cluster Heads

An optimal number of well separated cluster heads should minimize clustered network distance. For each protocol we recorded the average number of cluster heads that minimized clustered network distance based on 100 trials. We observed that the number of cluster heads that minimized clustered network distance depended on several network parameters. For example, we tested LEACH and DAC with different values for the radio model, base station location, and beacon range.

For LEACH, given the d^2 radio model, an average 64 cluster heads was optimal, but when the d^2, d^4 model was used, 50 cluster heads was optimal (a 22% reduction). This suggests that when the cost of communication with the base station is greater than

intra-cluster communication, less cluster heads are needed. However, this parameter did not have a significant impact on DAC, where we observed 31 and 30 cluster heads respectively.

Next we moved the base station further away from the sensor network, from (-50,-50) to (-500,-500). For LEACH, 56 cluster heads was optimal. The decrease was limited because the protocol needed enough cluster heads to cluster almost all sensors, which depends on beacon range. For DAC, 29 cluster heads was optimal.

And finally, we increased beacon range from 150 to 300 units. For LEACH, 22 cluster heads was optimal (a 66% reduction). For DAC, 9 cluster heads was optimal (a 71% reduction). While the radio model and base station location have an impact on the optimal number of cluster heads, beacon range is clearly the parameter that should be given the most attention.

Clustered network distance for in-network processing depends on number of sensors acting as cluster heads. LEACH may generate any number of cluster heads, DAC stops after all sensors have been clustered.

Protocol	distributed	separation	distance
LEACH	no	poor	poor
DAC	yes	excellent	excellent
k-means	no	good	good

Table 5.1: A comparison of clustering protocols during each *round* of operation. DAC is fully distributed, and exhibits better CH separation and distance reduction.

5.3.9 Scalability

We simulated networks of 200 to 14,000 sensors, using a beacon range of 150 units, and a d^2 radio model. As network size increased, the percent of cluster heads required for DAC to cluster all sensors decreased slightly. In a small network of 200 sensors, we needed about 4% cluster heads, in a network of 875, 3.07%, and in a large network of 14,000 sensors, 2.85%. This reduction is not possible where the number of cluster heads is an explicit parameter.

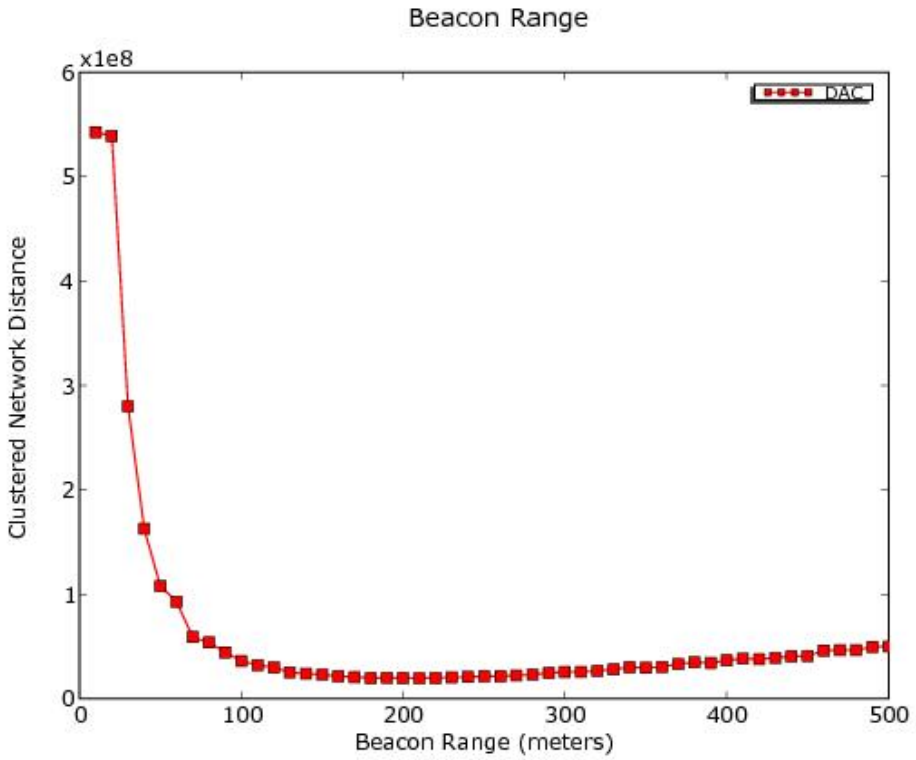


Figure 5.6: Optimal beacon range for DAC: 210 units, generated 15 cluster heads, $\approx 1.7\%$ of sensors.

Next we tested networks of 875 to 14,000 sensors. As network size was increased from 875 to 14000, LEACH's clustered network distance grew from ≈ 4 times greater to ≈ 5.5 than DAC's. DAC also performed better than a hypothetical k-means protocol.

5.3.10 Beacon Range

To estimate the optimal beacon range in our simulations, DAC was tested using beacon ranges from 10 to 500 in 10 unit increments and d^2 path loss. As shown in Figure 5.6, clustered network distance was minimized when beacon range was set at 210 units, which generated 15 cluster heads.

Then we compared LEACH with DAC where both used a beacon range of 210 units, and LEACH also used the explicit parameter of 15 cluster heads. DAC was still ≈ 4.2 times better at reducing clustered network distance.

5.3.11 Data Aggregation Tree

In this section we briefly consider the data aggregation trees that are likely to be generated in a deployment where radio communication range is limited and packets must be forwarded in a multi-hop fashion via CHs to the sink.

Figure 5.7 shows a typical data aggregation tree that forms when CHs are randomly located. The path from source to sink tends to be longer in terms of hops and Euclidean distance, resulting higher expected latency and energy consumption.

Figure 5.8 shows a typical data aggregation tree that forms when DAC is used. The path from source to sink tends to be shorter in terms of hops and Euclidean distance, resulting in lower expected latency and energy consumption.

5.4 Related Clustering Protocols

Many research efforts have addressed the problem of reducing communication costs in data gathering networks. By optimizing trade-offs network lifetime can be increased while still meeting a specific application's quality of service guarantees.

Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [95] assumes a clustering protocol similar to LEACH, and Adaptive Periodic TEEN (APTEEN) [94] uses LEACH centralized. Both TEEN and APTEEN were designed for *reactive networks*. In such networks, nodes are expected to respond immediately to sudden changes. Hard and soft thresholds were used to reduce message transmission. Naturally, this reduced the amount of data transmitted and extended network lifetime *vis-a-vis* LEACH, which is better suited for applications that require information based on periodic data from all nodes.

TEEN uses a hard threshold H_T and a soft threshold S_T to reduce message transmission. For each cluster set-up, each new cluster head broadcasts the two thresholds to its members. The first time each member node detects the attribute value $V >= H_T$, it switches on its transmitter and transmits the sensed value. Second and subsequent transmissions only occur when $V >= H_T$ and the change $v >= S_T$. The authors noted the possibility that a number of nodes would never transmit, so in APTEEN they

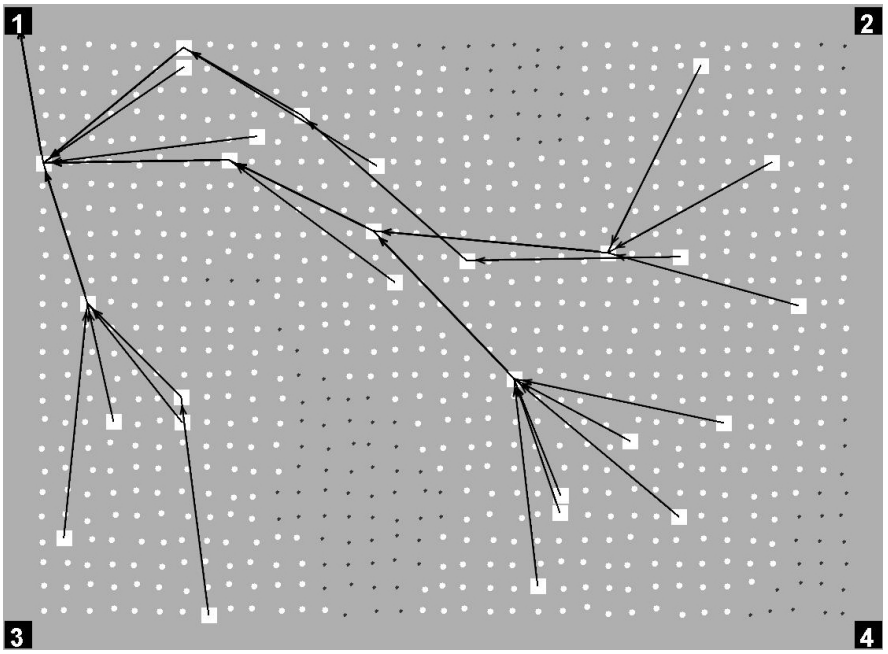


Figure 5.7: **LEACH Data Aggregation:** Data flows to sink #1, black square in upper left. Black dots are unclustered sensors, white dots are clustered sensors, and white squares are cluster heads.

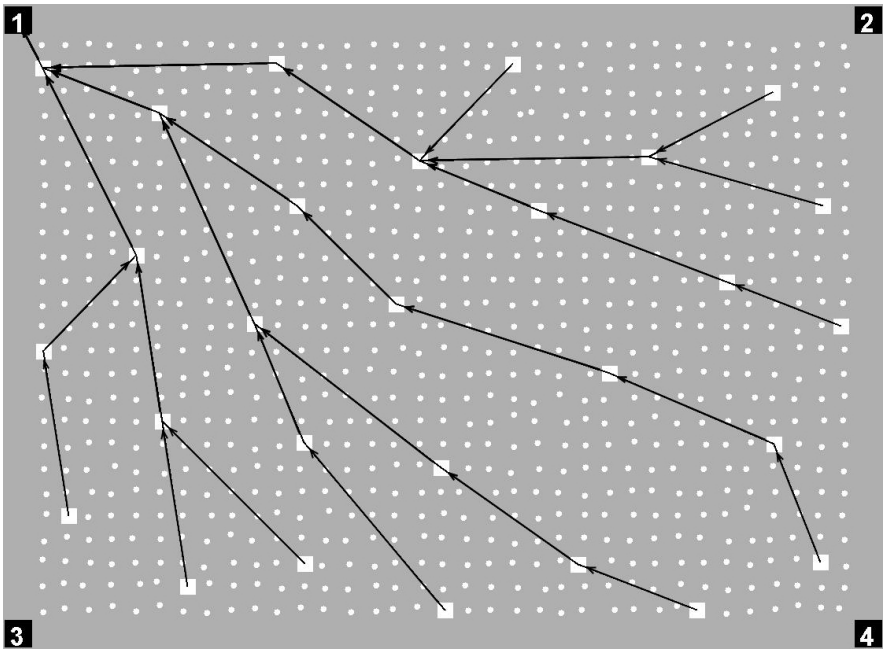


Figure 5.8: **DAC Data Aggregation:** Data flows to sink #1, black square in upper left. There are no unclustered sensors (black dots), white dots are clustered sensors, and white squares are cluster heads.

added a time out feature where T_C is defined as the maximum time period between two successive reports sent by a node.

Power-Efficient Gathering in Sensor Information Systems (PEGASIS) [86] was designed to be near optimal in terms of energy cost for data gathering applications. Given a trade-off between energy spent per packet and delay, an *energy x delay* metric should be minimized while meeting the application's QoS requirements. In their protocol, chain(s) of communication are built from neighboring nodes. During each round, if each node communicates only with close neighbors, and only one designated node sends aggregated data to the base station, then the total energy consumed by communication will be reduced. When the authors compared protocols, PEGASIS used CSMA and while LEACH used TDMA. We note that to make a valid comparison, both protocols should be tested using the same MAC layer protocol. In summary, PEGASIS assumes that all nodes have global knowledge of the network, and energy savings come with a penalty of increased delay.

Bandyopadhyay and Coyle [23] proposed a fast, randomized, distributed algorithm for organizing mobile sensors into a hierarchy of clusters. Their algorithm was designed to generate one or more levels of stable clusters with an optimal number of cluster heads on each level.

Distributed Energy-efficient Clustering Hierarchy Protocol (DECHP) [106] outperformed LEACH [53] in simulations partly because LEACH cluster heads communicate directly with the base station, while DECHP used multi-hop routing. Energy consumption was assumed to be proportional to distance d^4 – an assumption which strongly favors multi-hop routing. However, LEACH and multi-hop are not mutually exclusive, so a better comparison would require that both protocols use the same form of routing. While DECHP is an interesting protocol that addresses some of the problems associated with LEACH and other clustering protocols, its setup phase requires a large amount of communication overhead.

The Hybrid Energy-Efficient Distributed clustering protocol (HEED) presents fully distributed approach to promoting network lifetime [163] where each node uses local information to generate well separated cluster heads and elects cluster heads based on

their residual energy.

5.5 Conclusion

We have presented DAC, a low overhead, energy efficient approach to distributed asynchronous clustering. In order to make straightforward and fair comparisons with other protocols, we assumed direct communication between nodes, and hierarchical clusters where all CHs are at the same level. Multi-hop routing and n level hierarchical clusters, where some CHs always report to a higher CH, may provide additional energy savings and, depending on the application, many other optimizations are possible and compatible with DAC.

Based on our simulations, we observed that the number of cluster heads that minimizes transmission cost depends on a number of parameters including the radio model and base station location, but cluster head beacon range had by far the greatest impact. We noted that because limited wireless channel bandwidth must be shared by all clusters, a protocol should self-organize the network such that clustered network distance is minimized with as few cluster heads as possible. In light of these observations, similar protocols may need about twice as many cluster heads as DAC where our standard network parameters, the d^2 radio model and a limited beacon range are assumed.

The benefits of DAC increased with the scale of the network, and with the cost of radio communication. DAC builds an infrastructure for in-network processing and routing via cluster heads [47] that extends the lifetime of large scale networks.

DAC uses a range limited cluster head beacon and pre-emptive recruitment to generate well separated cluster heads; it automatically stops generating cluster heads after all sensors have been recruited. This reduces communication costs between sensors and their CHs, which are guaranteed to be relatively close, between CHs and the sink by data aggregation; and it promotes connectivity between CHs and the sink. Well positioned cluster heads may be replaced by more powerful devices to facilitate small-world properties – which we discuss in Chapter 6.

Chapter 6

Clustering for Small-World Properties

6.1 Introduction

In this chapter we tackle the problem of designing protocols for the self-organization of large scale wireless sensor networks (WSNs). We compare Distributed Asynchronous Clustering (DAC) [49], presented in Chapter 5, using two mechanisms, *random recruiting* and *pre-emptive recruiting*, for generating energy efficient wireless networks built from motes, gateways and hubs.

Deployments of many thousands of resource constrained sensor nodes (*motes*), which are low cost, stationary, and designed to do little more than take measurements and transmit, have been envisioned [30], but they are not scalable due to the limited battery power and radio range of motes. In contrast, heterogeneous WSNs are scalable if a relatively small number of progressively more powerful devices from *gateways*, *hubs*, and so on, to nodes with satellite uplink capability, are deployed in locations that provide efficient connectivity.

Protocols for self-organization should use pre-emptive recruiting, not random recruiting, to generate networks with small-world properties. Such networks are highly clustered and provide short paths between arbitrary nodes and, consequently, information spreads more easily in small-world than random or regular networks [152]. In the sensor network context, they provide energy efficient, low latency communication.

In our simulations, we generated tiered networks of N heterogeneous devices: motes at level 1, gateways at level 2, and hubs at level 3; a level $n+1$ device was assumed to be at least an order of magnitude more powerful than a level n device. Motes transmitted data of interest over short distances to gateways, and the long range radio links provided by gateways and hubs enabled *short cuts*. The resulting network

had *small-world properties*: motes were highly clustered, and the number of hops between any mote and any sink was $O(lgN)$. As a result, the network required less energy to report measurements, and communication latency from source to sink was $O(lgN)$.

The importance of small-world networks and their models is not reflected in the sensor network literature. For example, the well known protocols defined in [53] and [64] do not consider small-world properties, the applicability of the small-world concept to WSNs was investigated only briefly in [59], and the utility of a few powerful nodes was not considered until very recently in [42]. *In view of past research and our experimental results, we present a mechanism for generating small-world wireless sensor networks.*

This chapter is organized as follows. Section 6.2 provides some background on complex networks. In Section 6.3 we discuss the importance of hierarchical architectures in large networks, distributed clustering for self-organization, and in-network processing. Section 6.4 describes the experimental setup, and Section 6.5 presents our results. Section 6.6 briefly summarizes closely related work, and Section 6.7 concludes this chapter.

6.2 Complex Networks: Small-World

The structure and function of complex networks, where many non-identical elements are connected by diverse interactions has been the subject of an enormous amount of research. An extensive review of the fundamental principles that govern the formation and evolution of various complex technological and social networks may be found in [112]. In this section we review complex networks and suggest that insights gained from the study real small-world networks and their models can be used to inform the design of protocols for self-organizing WSNs.

6.2.1 Network Analysis

Networks, such as regular, random, small-world, and scale-free, may be analyzed in terms of degree distributions, clustering, and the distance between nodes [152].

In network analysis, the simplest characteristic of a node is its degree k. The *degree* k_i of node i is defined as the number of edges incident on i. The *average degree* of a network is the average k_i over all i and is denoted $\langle k \rangle$. $P(k)$ is the distribution function of node degrees. It is the probability that a randomly selected node has exactly k edges. If the distribution is *exponential*, then $P(k)$ becomes vanishingly small for large k. In contrast, if the distribution is *power-law*, then $P(k)$ does not become vanishingly small for large k, i.e., the distribution has a long right tail of values far greater than the mean [112]. All scale-free networks [10], where power-law degree distributions are of interest, are believed to display small-world properties [133].

The *clustering coefficient* C of a network is the average fraction of pairs of neighbors of a node that are also neighbors of each other. C indicates the cliquishness of a typical neighborhood. *Average path length* L is the average geodesic distance between all pairs of nodes. The *distance* d_{ij} between nodes i and j is defined as the number of edges along the geodesic path connecting them. L is relatively small for many real world complex networks and this is consistent with the *small-world effect* many people have experienced in their social networks. Watts and Strogatz refer to L as the *characteristic path length* of a network [152]. *Diameter* D is the maximum geodesic distance between any pair of nodes.

We note that distance may be measured in terms of hops or Euclidean distance; the preferred measure depends on the network [37]. In a WSN, both Euclidean distance and number of hops over unreliable radio links may have a large impact on QoS and communication costs. A data gathering WSN is dominated by many-to-one traffic, so the metrics used in our simulations measure the distance from sensor nodes to the sink, rather than the distance between arbitrary nodes.

6.2.2 Random Graphs

Modeling of complex systems and processes has often assumed that interaction patterns can be mapped onto a regular structure. In the late 1950s Erdös and Rényi (ER) made a breakthrough in classical graph theory when they described a complex network topology with a random graph [28].

The most commonly studied model is denoted $G_{n,p}$ where graph G is comprised of n nodes which are linked independently with probability p, and not linked with probability $1 - p$. The average number of links is $m = \frac{1}{2}n(n-1)p$. The average degree of a vertex is

$$z = \frac{n(n-1)p}{n} = (n-1)p \simeq np$$

where np is good for large n. When the number of links is small, there are many small components, which are clusters of connected nodes. As more links are added, the components grow in size, first by connecting to isolated nodes, and then by connecting to other components. A phase transition occurs at $m = \frac{n}{2}$ when a giant component forms, and for $m > \frac{n}{2}$ the giant component contains $O(n)$ nodes, i.e., scales linearly with n [141].

While random graphs reproduce one of the properties of real world networks, short average path lengths, in almost all other respects they do not. The probability that two nodes are linked is p regardless of whether they have a common neighbor, so $C = p$, and they have a Poisson degree distribution. In short, the random graph makes a good straw man [112].

6.2.3 Small-World Networks

Small-world networks have their roots in studies of social networks where most people only know their friends and perhaps a few of their friends' friends. In such networks most pairs of vertices seem to be connected by a short path. One of the first demonstrations of this *small-world effect* was by Milgram [99]. In his experiment, participants were asked to pass a folder of documents to one of their acquaintances in an attempt to

deliver it to the assigned target (identified by name, profession, and city). The results showed the existence of short paths (about six hops), and a more surprising result – that ordinary people are good at finding them [112]. This becomes more interesting when we consider that short paths also exist in random networks, but such paths could not be found by people using realistic information [112].

Regular networks are highly clustered and, therefore, provide high levels of local interaction and small path lengths between spatially correlated nodes. Random networks provide small path lengths between arbitrary nodes. Small-world networks provide both properties: they are highly clustered and have small average path lengths.

The fundamental idea behind small-world networks is the notion that networks gain efficiency by having a large number of local links and a few global links connecting local clusters together. In the WSN domain, distance metrics should consider not only Euclidean distance and their impact on the energy required for radio communication, but also the number of hops and their impact on latency.

6.2.4 Small-World Models

The *small-world effect*, that a small number of acquaintances may connect unrelated people, was studied by Milgram in the 1960s [99, 146]. More than three decades later, Watts and Strogatz proposed a model of small-world networks, and speculated that the small-world phenomenon is probably generic for many large, sparse networks found in nature [152].

Both the Watts and Strogatz (WS) and the Newman and Watts (NW) small-world models start with a regular network (lattice) and introduce a number of long range links [152]. A review of small-world models is provided in [110].

The WS small-world model was introduced in 1998 [152]. This model assumes a one dimensional network ring of n nodes of degree $k = 4$. Each node has four edges connecting it to its four nearest neighbors. For each node, each of its edges is subject to a small probability p of being rewired and attached to a distant node. By varying p from 0 to 1, they interpolated between a regular and a random network. When $p = 0$, the

network is regular, highly clustered and large-world: average path length $L(0) \sim \frac{n}{2k}$ grows linearly with n. When $p = 1$ the network is random, poorly clustered and small-world: average path length $L(1) \sim \frac{\ln(n)}{\ln(k)}$ grows only logarithmically with n [152].

The WS model showed that by adding a few random long range links, average path length L was drastically reduced, but had little impact on the network's local connectivity C. They found a broad interval of p over which $L(p)$ is only slightly larger than L_{random} yet $C(p)$ is much greater than C_{random} [152].

$$L(p) \gtrsim L_{random} \sim \frac{\ln(n)}{\ln(k)} \tag{6.1}$$

$$C(p) \gg C_{random} \tag{6.2}$$

In the Newman and Watts (NW) small-world model [111], edges are not rewired. Rather, an edge between two nodes is added with a probability p. When $p = 0$, the NW model reduces to the original nearest k neighbors model. If $p = 1$, then the network is fully connected.

6.2.5 Applications of Small-World Research

Watts and Strogatz computed L and C for the power grid of the western United States, the neural network of roundworm *Caenorhabditis elegans*, and the collaboration graph of film actors. All three were found to be small-world networks. They also investigated the functional significance of small-world connectivity for dynamical systems, and concluded that models of systems with long range links tend to show enhanced signal-propagation speed, synchronizability, and computational power [152]. **Paths over a small number of links may provide high-speed communication channels between distant clusters, and facilitate dynamical processes (such as synchronization and computation) that require global coordination and information flow.** Strogatz provides an excellent review research on complex networks in [141].

Recently, networks are said to show the *small-world effect* if the value of L "scales logarithmically or slower with network size for a fixed mean degree" and, for networks

with power-law degree distributions, L scales $\frac{\log n}{\log \log n}$ [141].

6.3 Scalable Wireless Sensor Networks

Sensor networks are usually comprised of many power constrained nodes deployed inside or very close to a phenomenon of interest, and their protocols and algorithms must possess self-organizing capabilities [2]. They should be inherently distributed, use localized communication, and their network architectures must be hierarchical for scalability [77, 40].

While hierarchical clustering simplifies routing and facilitates data aggregation, the radio range of cluster heads should be commensurate with their level. In this section we discuss some design goals and present a mechanism for topology generation, pre-emptive recruiting, which generates a hierarchy of well separated cluster heads, i.e., well separated devices with long range radios.

6.3.1 Cluster Formation

The performance of a flat multi-hop network will deteriorate rapidly as network size N increases if all nodes send raw data to a sink because per node throughput scales as $\frac{1}{\sqrt{N}}$ [43]. Therefore, one of the key motivations for clustering protocols is to facilitate in-network processing, such as data aggregation, close to the data source to reduce the amount of data transmitted.

Cluster-based routing protocols naturally organize sensor nodes such that sensed data may be efficiently relayed by gateway and hub devices to the sink. This raises the research question, how should the clusters be formed such that the energy consumption and communication metrics such as latency are optimized? The factors affecting cluster formation and cluster-head communication have been identified in [1] as open questions for future research.

6.3.2 Single-Hop versus Multi-Hop Routing

In recent years some researchers have claimed that multi-hop routing consumes less energy than single-hop [121], other researchers have claimed the opposite [45, 165].

Given a model for energy consumption per bit, $\alpha + \beta d^n$, where α represents the distance-independent energy consumed by transmitter and receiver, and βd^n represents the distance-dependent energy consumed by the transmitter, [101] found that α was the dominant cost in most cases. "For virtually all of today's short-range radios" α substantially exceeds βd^n, "*even at the radio's maximum output power*" [100].

More recently, it has been shown that whenever source and destination are within range, single-hop is almost always more power efficient than multi-hop routing [151]. When source and destination are not within range, and only motes are available as relay nodes, the network topology should limit hop distance. In [6], multi-hop reliability at the mote level in a three tier network became insufficient after 5 to 6 hops, so they partitioned their network into subnetworks where each mote was usually closer than 5 hops from a tier 2 device.

6.3.3 Distributed Clustering

Clustering and in-network processing are well known and scalable techniques for reducing communication costs. They may reduce transmission distance and the amount of data transmitted. In general, sensor nodes are partitioned into a set of clusters where each sensor may only transmit data to its cluster head. Cluster heads process member data and forward the result to the sink. However, existing clustering protocols tend to suffer from limitations related to latency, overhead, centralization, global knowledge of the network, lack of scalability, and suboptimal numbers of poorly separated cluster heads.

In our simulations, we compared two versions of the DAC clustering protocol [49]. The first used the random recruiting mechanism, which is the same as Algorithm 2 except that nodes within beacon range are still free to volunteer for cluster head duty. The second used the pre-emptive recruiting mechanism defined by Algorithm 2.

Both versions of DAC used the sensor network itself to generate two and then three tier heterogeneous networks. Deployment proceeded as follows. A large number of low cost homogeneous motes were deployed in an approximate manner – possibly over irregular terrain. Some motes volunteered for gateway duty and recruited cluster members. Gateways were deployed automatically to replace the volunteers. Some gateways volunteered for hub duty and recruited nearby gateways as cluster members. Hubs were deployed automatically to replace those volunteers.

Algorithm 2: Pseudo code for the pre-emptive recruiting mechanism where each node is a mote, gateway, hub, and so on, as required by the application.

```
initialization;
while node.isFree do
    node.listen();
    if node.hearsBeacon( ) then
        node.isFree ← false;
        node.recordRSS( );
    else if node.random(0..1) p then
        node.isFree ← false;
        node.isClusterHead ← true;
        node.setBeaconRange( );
        node.broadcastBeacon( );
    end
end
node.listen(time_period) ;
clusterHead ← node.maxRSS( ) ;
node.associate(clusterHead) ;
```

At this point we note that in a real deployment the effective range of a device may drop precipitously. For example, a weather change from dry to rainy/foggy conditions could cause range to decrease by ≈ 80 percent [5]. In order to maintain connectivity with the sink under adverse conditions, gateways and hubs should be positioned, at least approximately, where they are most needed. Otherwise the network will not scale.

6.3.4 Network Topology

The importance of a node depends on its centrality, and four measures are widely used in network analysis [109]. The *degree centrality* k of vertex i is the number of links

attached to it.

$$k_i = \sum_{j=1}^{n} A_{ij}$$

However, the importance of vertex i may depend on both the quantity and quality of its links. The *eigenvector centrality* x of i is a weighted average of adjacent vertices.

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j$$

where λ is a constant, $\mathbf{x} = (x_1, x_2, ...)$ is a vector of centralities, can be rewritten in matrix form as

$$\lambda \mathbf{x} = A \cdot \mathbf{x},$$

where $\mathbf{x}$ is an eigenvector of the adjacency matrix with eigenvalue λ. A variant of this measure is used by the search engine Google to rank Web pages.

The next two measures are based on shortest paths in terms of links traversed from source to destination. *Closeness centrality* cc_v is the mean geodesic distance between a vertex v and all other reachable nodes V

$$cc_v = \frac{\sum_{t \in V} d_G(v, t)}{n - 1}$$

where $n \geq 2$ is the size of the set of vertices V reachable from v. This measure indicates, for example, how long it would take for information to spread from a given vertex to other vertices in the network.

Betweeness centrality bc_v is the fraction of shortest paths connecting other vertices s and t that run through vertex v.

$$bc_v = \sum \frac{\sigma_{st}(v)}{\sigma_{st}}$$

This may be normalized by dividing by the number of pairs of vertices not including v, which is $(n-1)(n-2)$ [109].

In our simulations, the highly resource constrained motes connect to the network infrastructure via gateways, which means that many more links are incident on gateways than hubs. Hence, gateways link to hubs, not because of a mechanism such as preferential attachment, where new nodes prefer to link to high degree nodes [10], but because they lie in the shortest path between many sources and destinations. *In other words, hubs are not central in terms of their node degree distribution, they are central in terms of their betweeness.* We expect large scale WSN topologies to be somewhat similar to world wide airline networks where the nodes with the highest degree centrality do not necessarily have the greatest betweeness centrality.

6.4 Experimental Setup

We implemented a wireless sensor network simulator to test the relative performance of a network self-organized by DAC using random recruiting (RR) to elect nodes for cluster head duty, versus DAC using pre-emptive recruiting (PR). Although both mechanisms can be used to generate tiered networks with small-world properties, at least to some extent, the metrics of network analysis suggested that PR would generate topologies characterized with lower communication costs.

The topology generated by each version of DAC was evaluated with metrics based on Euclidean distance (a proxy for energy consumption), and hop count (a proxy for latency).

6.4.1 Network Model

The parameters for our *standard network* are shown in Table 6.1. Nodes were deployed on a regular two dimensional grid of rows and columns at 30 unit increments. Sinks were deployed off the grid at each corner, e.g., sink #1 was located at $x = -30, y = -30$.

The parameters for our *standard nodes* are shown in Table 6.2. The gateway, hub and sink nodes were assumed to be equipped with GPS and, therefore, aware of their own coordinates, and all nodes within range. The motes were not assumed to be aware

Table 6.1: Standard Network Parameters.

Sink 1 (x, y)	Grid	Gap Size	Path Loss
$(-30, -30)$	regular	30 units	d^2

of their coordinates, instead they simply determined which cluster head was closest by received signal strength.

Table 6.2: Standard Node Parameters: mote and gateway devices recruit, hub devices do not.

	Number	Beacon Range	Comm Range
Mote	≤ 900	150	150
Gateway	≤ 28	300	300
Hub	≤ 8	NA	600
Sink	≈ 4	∞	∞

The distance between almost all motes and the sinks is *far greater* than a single mote's radio range. The path loss exponent n was assumed to be 2 for all mote to CH transmissions. Beacon range was assumed to be 150 and 300 units (distance is relative) for mote and gateway devices respectively. In this simulation, hub devices do not volunteer and use a beacon to recruit because they are already at the top level (there are no "super hubs").

At this point we note that while a protocol may offer large performance gains under ideal conditions, it may be the case that it fails badly under more realistic conditions. In a real deployment where radio range may vary greatly over time, the beacon range should be set at significantly less than the expected communication range for link stability. Nonetheless, we found that adding greater detail to our simulations, such as variable conditions, unreliable radio links, a random deployment of nodes instead of a regular deployment, and obstacles to radio propagation did not have a material impact on our results and, therefore, will not be discussed further.

6.4.2 Radio Model

In general, the power required for a successful transmission P_t depends on Euclidean distance and path loss. $P_t \propto d^n$ where d is distance, and n is the path loss exponent. Depending on the radio propagation environment, n is typically in the range of 2 to 5 [32].

It should be noted that radio range is non-uniform, non-circular [75], highly variable and difficult to model [53]. Consequently, in [53] the radio was assumed to use a fixed amount of energy per bit to run the transmitter or receiver circuitry, and a variable amount per bit to run the transmit amplifier. The variable amount of energy consumed by their radio was proportional to d^2 and d^4 for intra-cluster and CH to sink communication respectively. Our simulations used an equivalent radio model because a more complex model would not more useful.

6.4.3 Distance Metrics

We simulated a *data gathering network* where all nodes sense their environment at a steady rate. The benchmark task was to periodically determine the average sensor measurement. For this task, the communication cost of delivering data from many sources to the sink over a number of radio links dominates. The primary metrics of interest are based on Euclidean distance (a proxy for radio cost), or hop count (a proxy for latency).

6.4.3.1 Euclidean Distance

The energy consumed by radio communication can be approximated by a function of distance d and a path loss exponent of n: $P_{tx} \propto d^n$. The total energy required to transmit a packet from all motes to the sink using data aggregation is proportional to $\sum d_i^n$ for all links in the data aggregation tree..

However, this chapter considers a three tier network of heterogeneous devices where gateways are less energy constrained than motes, and hubs are not subject to significant energy constraints. Motes are by far the most energy constrained devices –

they are the weakest link in the data gathering chain – and the cost of radio communication from the motes to their cluster heads is critical. This metric is an abstraction because in practice, some motes will be out of range and their packets would have to be delivered via multi-hop routing to the nearest CH. The energy required for all motes to report to their CHs is proportional to $\sum d_i^n$ for all mote to CH links.

6.4.3.2 Hop Distance

This metric was used to measure the number of hops from each mote to the sink where data packets must be delivered multi-hop from motes via gateways and/or hubs. Two metrics used hop counts. (1) *Diameter* D: the maximum number of hops from a mote to the sink. (2) *Average Path Length* L: the average number of hops from all motes to the sink.

We used hop count as a proxy for latency and reliability. As Figure 6.1 shows, the expected diameter of networks with a regular lattice structure ("mesh"), where only adjacent nodes are linked, increases much faster than is the case for a small-world networks. Next, each link was assumed to be 95% reliable given a limited number of retries. As Figure 6.2 shows, the expected probability of data delivery decreases much faster for regular networks than for small-world networks.

The energy consumed by radio communication and latency depends on the number of hops from source to destination and the probability of a failed transmission on each link. For example, if links are established such that the probability of a failed transmission is 0.50, then each hop would require on average two tries for each data packet and two tries for each acknowledgment. The implication is that in a real deployment, effective path length may be much longer and packet loss much higher than indicated by hop distance alone. In other words, for scalability, large WSNs must have small-world properties in order to meet QoS requirements.

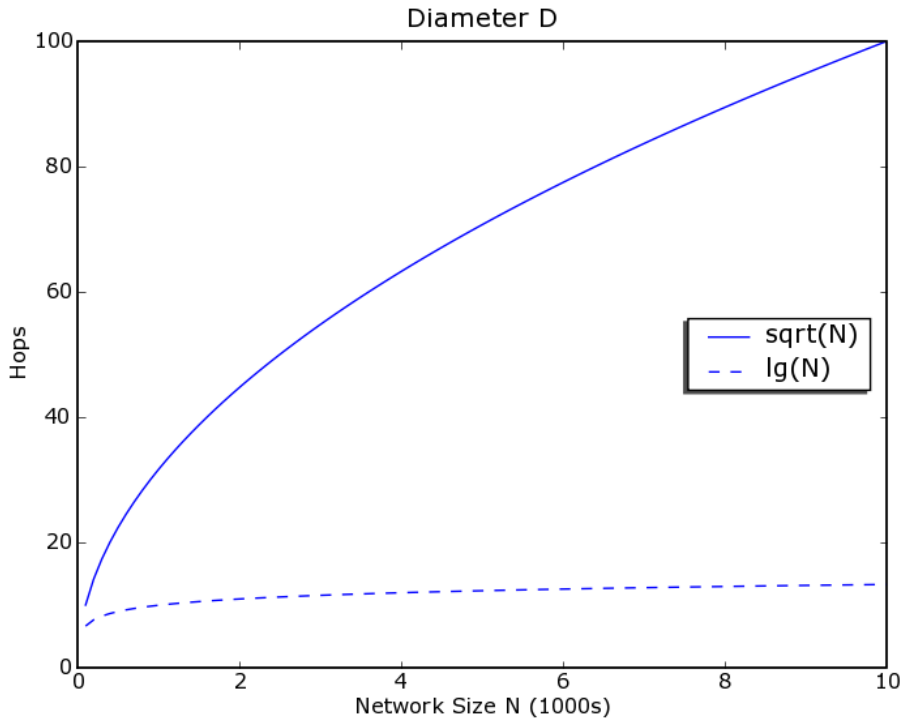


Figure 6.1: **Regular versus Small-World Networks:** As network size increases, the relative efficiency of a small-world topology increases in terms of diameter.

6.5 Experimental Results

The results reported in this section are based on 100 trials. We simulated networks where nodes were deployed onto a regular two dimensional grid such that the horizontal and vertical distance between each mote was about 30 units. A small percent of the motes were replaced by gateways and hubs. Each device was assigned a range based on the expected range for that device type. Motes: $R1 = 150$. Gateways: $R2 = 300$. Hubs and sinks: $R3 = 600$.

6.5.1 Euclidean Distance

In this section we consider a three tier hierarchical network of 875 nodes where each packet is routed from mote to gateway to hub to sink.

Figures 6.3 and 6.4 show data aggregation trees. Some gateways (large white cir-

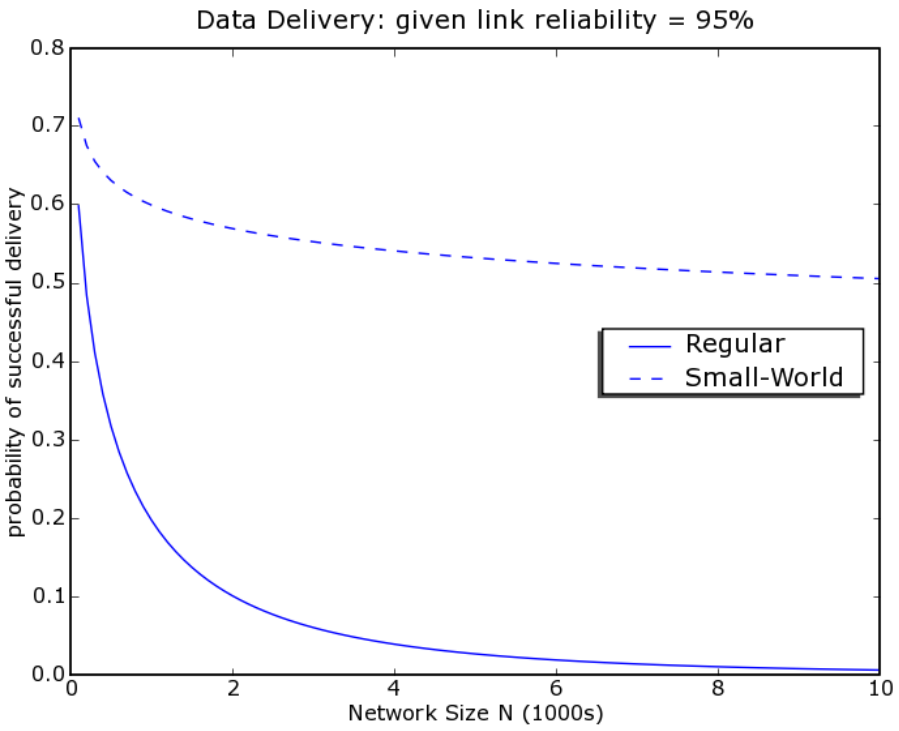


Figure 6.2: **Regular versus Small-World Networks:** As network size increases, the relative efficiency of a small-world topology increases in terms of the probability of successful delivery.

cles) have been replaced by hubs (large white squares). All motes (small dots) are assumed to be linked their gateway by one or more hops. Thin lines indicate established radio links between gateways and hubs. Thick lines indicate established radio links between hubs and the sink.

Random locations for gateways and hubs (data aggregation and long range links) are less effective than well separated locations – it appears from the Figures 6.3 and 6.4 that the PR mechanism generated a more efficient topology. However, in this section we focus on the mote to nearest CH link. For a path loss of $n = 2$, PR generated clusters that required $\approx 50\%$ less energy for all motes to report to their CHs. For a path loss of $n = 4$, PR generated clusters that required over 99% less energy. Obviously, in radio propagation environments characterized by high path loss, a network with randomly located CHs is very likely to fail.

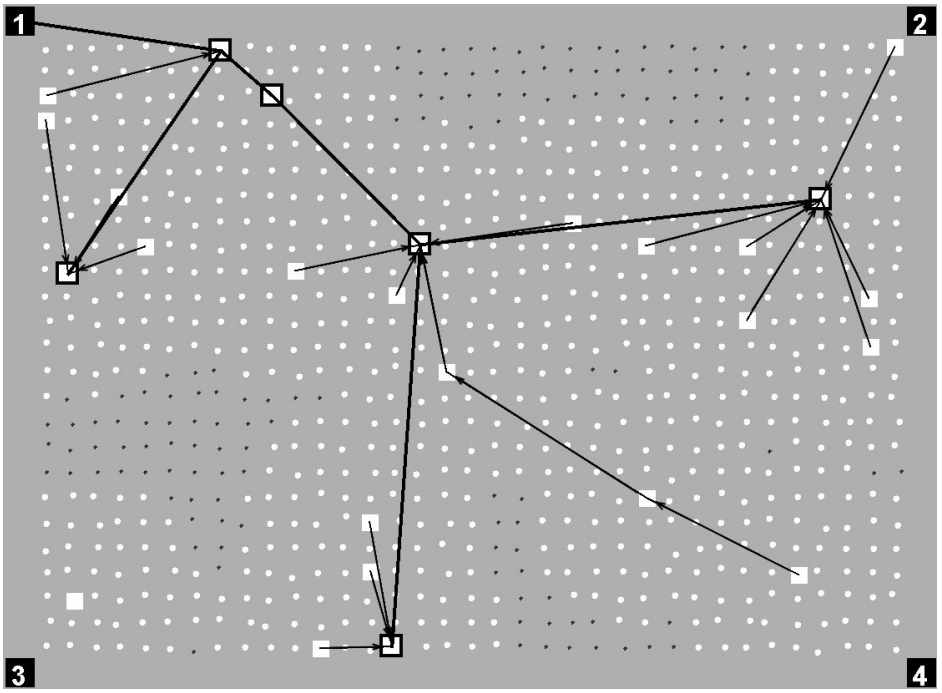


Figure 6.3: **Three Tier Grid Deployment:** Data flows to sink #1, black square in upper left. Black dots are unclustered motes, white dots are clustered motes, white squares are gateways, and black outlined squares are hubs.

6.5.2 Hop Distance

As we discussed in Section 6.2, small-world networks are characterized by high local connectivity and short path lengths where even random long range links reduce expected path length. In this section we discuss the impact of links between well located gateways and hubs on hop distance from motes to the sink where packets are routed multi-hop via gateways and hubs.

Section 6.5.1 assumed limited beacon range, but all nodes were presumed capable of direct communication with the sink. **In this section radio communication range is limited and, therefore, multi-hop paths are used in topologies where there are no gateways or hubs within range.** Normally, the next hop within range on a path to the sink, in order of preference, is the best hub, gateway, or mote, where best means closest to the destination.

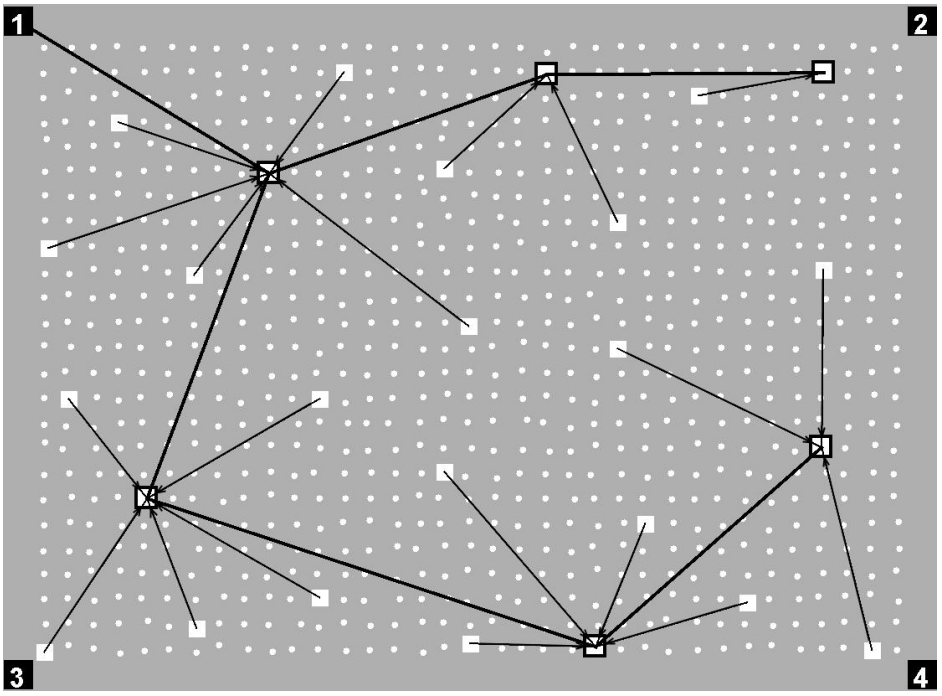


Figure 6.4: **Three Tier Grid Deployment:** Data flows to sink #1, black square in upper left. Black dots are unclustered motes, white dots are clustered motes, white squares are gateways, and black outlined squares are hubs.

6.5.2.1 Small-World Properties in WSNs

During self-organization, the WSN must decide where to add long range links. In [59], Helmy examined an architecture designed to reduce resource discovery overhead. He found that given a network of 1000 nodes and about 5000 links, adding just 25, 80, and 150 links achieved a $40 - 60\%$ reduction in average path length L. Random long range links that were limited to $25 - 40\%$ of network diameter minimized L. A higher limit actually increased L. In this section we consider slightly different metrics based on path lengths from any node to the sink.

6.5.2.2 Scalability

We used two metrics to compare the network topologies generated by the mechanisms RR and PR: average path length from the motes to the sink L, and diameter, the longest path length from a mote to the sink, D.

Figure 6.1 shows, in theory, that network diameter D grows much more slowly in

a small-world than in regular networks. In our simulations, which we discuss in the following sections, compared to RR, PR generated networks with better small-world properties in that they scaled better with network size N based on metrics L in the average case and D in the worst case.

6.5.2.3 Path Lengths

In this section we are primarily interested in the number of radio links from source to the sink (hop distance) in our simulated network where 875 nodes are deployed at regular locations on a two dimensional grid. We define diameter D as the maximum geodesic path from any mote to the sink, and L as the average path length from the motes to the sink. Route discovery was implemented with a greedy depth first search algorithm where all radio links were required to be symmetrical.

First we note that in three tier networks, the ideal path from a mote to a sink would proceed via one gateway and one or more hubs. However, if the preferred next hop is out of range, the network must resort, at least temporarily, to mote or gateway multi-hop. For DAC-RR, the addition of eight long range hubs was not enough to guarantee preferred connectivity. For DAC-PR, the addition of about seven or eight hubs guaranteed preferred connectivity.

Second, to the extent that ideal paths were realized, path lengths were reduced. For a homogeneous mesh topology, where motes only have radio links with adjacent nodes, the network had a diameter of about 30 hops, and a characteristic path length of about 21 hops. Table 6.3 shows that the addition of gateways and hubs reduced maximum D and average path length L in two and three tier networks. Table 6.3 also shows the substantial advantage of pre-emptive recruiting over random recruiting. In addition, pre-emptive recruiting was far more reliable – it only required 103 attempts to generate 100 connected three tier networks while random recruiting required 163 attempts.

Next we note that for our 875 node network, ranges $R2 = 300$ and $R3 = 600$ are about 23% and 45% of the networks Euclidean diameter respectively, and consider how well this empirical data agrees with research on complex networks.

Table 6.3: Diameter D and average path length L are the number of hops from network sources to the sink.

	Mechanism				
	Mesh	Random	Pre-emptive	Random	Pre-emptive
Tiers	1	2	2	3	3
Diameter D	30	18	11.4	15.1	6.6
Length L	21	8	6.9	5.3	4.6

In the context of a two tier network, considering Helmy's results [59] and assuming at most two nodes per link, we expected DAC-PR to need a minimum of 26 to 50 gateways, i.e., $2.6\% - 5\%$ of nodes, to get a 40% reduction in average path length. We found that where 3.2% of nodes had links of up to 23% of maximum, the number of hops from any mote via its CH to the sink in the worst case was reduced by $\approx 80\%$, and average path length by $\approx 77\%$. This is somewhat consistent with [59] where it was suggested that random long range links should be limited to $25 - 40\%$ of network diameter in order to yield a $40 - 60\%$ reduction in L.

In the context of a three tier network, the next hop for data aggregation in the simulator was, for most cases, defined as the best hub within range, or the best gateway within range. We made two observations. First, DAC-PR consistently generated a network characterized by paths length approximately $lg(N)$ hops long. Second, increasing the communication range of the hubs was only helpful up to a point. Ranges longer than necessary, such as $R3 > 600$, did not change the number of hops on a path from source to destination because the best next hub was almost always in range at $R3 \leq 600$.

6.5.2.4 Radio Ranges for Small-World Properties

Based on the empirical data, PR is an effective mechanism for generating large scale WSNs given the modest assumption we are not limited exclusively to motes. If we are not limited, then WSNs may be self-organized for small-world properties if a set of devices is deployed with radio ranges $R = \{150, 300, 600, ..., max\}$ such that R_{max} is about 50% of network diameter D.

When delivering data from a mote to the sink via progressively more powerful devices, the length of each hop toward the sink doubles every hop, e.g, $150 + 300 + 600 = 1050$ and where $D = 1320$, path length is just four hops. In other words, we can expect average path length L to scale logarithmically with network size.

6.6 Related Work

In this section we briefly review closely related work on self-organizing wireless sensor networks for small-world properties.

Wireless networks belong to the category of *spatial* graphs, not *relational* graphs, so the small-world concept as presented in [152] may not be directly applicable [24]. The primary concern has been that the links in WSNs are determined by radio connectivity, which is a function of distance, the environment, and other factors. For example, the radio range of a Crossbow MICAz mote is specified as $\approx 20 - 30$ meters indoors, $\approx 75 - 100$ meters outdoors, and the range may be much less in real deployments [81].

Assuming that "short cuts" could represent physical links, or logical links that translate into multiple physical hops, [59] considered whether or not it is possible to develop network architectures for resource discovery. They proposed a contact-based architecture for resource discovery in large scale wireless networks where the goal is to reduce the number of resource discovery queries. However, they did not propose a mechanism to generate networks with small-world properties. Dixit, et al., investigated whether small-world and scale-free models are applicable to cellular wireless networks, "which typically do not exhibit self-organizing or scalability properties due to the limited range of the wireless nodes" [24]. Their goal was to design self-organizing mechanisms that would generate robust, reliable, scalable, efficient wireless networks, but their design assumed the careful placement of identical fixed relay nodes to improve service to mobile users [24]. Guidoni, et al., [42] applied small-world concepts to the design of heterogeneous sensor networks. They added a small number of powerful high-end sensors, but did not consider a hierarchy of devices and their placement.

For some applications it may be feasible to augment the WSN with a small number of wired links. In [135] the authors investigated the use of wired short cuts, equipped with radio transceivers at each end, to reduce energy consumption. Given a greedy geographic routing protocol, they wanted a network design where the number of wired short-cuts is much less than the number of nodes that would reduce the average number of hops and energy dissipation skew as much as possible. They considered the placement of wires in two networks. First where the sink is static, and second where the sink is mobile.

For the static sink WSN, they found that the deterministic placement of a small number of wired short-cuts, where all wires originate from the cell containing the sink node, was best for reduced energy consumption and load balancing. For the mobile sink WSN, their findings were consistent with [74]. An inverse square distribution, where the probability that u and v are connected by a wire depends Euclidean distance, $d(u, v)^{-k}$, strikes a nice balance at $k = 2$ between the number of short cuts connecting "nearby" and "faraway" nodes such that the average hop count and nonuniform energy consumption is minimized.

6.7 Conclusion

In this chapter we simulated the self-organization of a WSN for small-world properties. Motes were deployed randomly in a field, and some nodes were replaced by less resource constrained nodes called gateways, and powerful long range nodes called hubs. These nodes were identified by the DAC hierarchical clustering protocol. During our trials, DAC used either random recruiting or pre-emptive recruiting. We found that the pre-emptive recruiting (PR) mechanism generated networks with superior topologies.

In effect, the PR mechanism identifies well separated locations for cluster heads (gateway and hub devices) and, consequently, generates a more scalable topology in terms of connectivity, latency, and energy efficiency. In contrast, randomly located cluster heads provided a much less effective topology as indicated by the Euclidean distance of motes to their CH, and the characteristic path length and diameter of the

networks generated.

Simulation results consistently showed that well separated cluster heads reduced the load on motes, and facilitated higher utilization of more expensive devices such as gateways and hubs. Expected radio transmission costs due to Euclidean distance and number of hops was reduced; expected network lifetime was increased. Given gateways and hubs with sufficient radio range, and the small-world effect where $L \propto \lg(N)$, data gathering applications may scale to cover very large geographic areas when PR is used. In Chapter 7 we show that, not only do short paths exist, our data-centric routing protocol will use them for query dissemination.

Chapter 7

Bloom Gradient Routing in a Small-World

7.1 Introduction

This chapter presents our protocol, Bloom Gradient Routing (BGR), in the context of a large scale three tier wireless sensor network (WSN) of heterogeneous devices. Bloom filter representations of sensor meta-data are diffused "up" to provide gradients that guide queries "down" to *a priori* unknown sensors with matching data.

In general, different applications may require different communication protocols and hardware to meet quality of service guarantees and maximize network lifetime. Applications that gather named data from the network rather than from individually addressable nodes are known as *data-centric*, but existing protocols are not scalable. Sensor Protocols for Information via Negotiation (SPIN) [79, 51] and Directed diffusion [30, 63, 64, 137] are two well known data-centric routing protocols. According to SPIN, sensors exchange meta-data before transmitting to reduce communication costs, but data delivery becomes less reliable as network size increases. Directed Diffusion was designed for flat network of homogeneous nodes and, therefore, does not scale to large networks – especially where many sinks disseminate many unique queries.

Here, we consider a data gathering application, built on the query-driven data delivery model, where independent sinks inject *snapshot* queries into the top level of a hierarchical network and the query flows down to the motes. After each mote receives a query packet, it extracts the instruction, then senses its environment and transmits matching data up to its gateway CH, which forwards the packet primarily via hubs toward the query source.

The network itself is composed of very resource constrained battery powered sensor nodes referred to as *motes* at level one, resource constrained nodes referred to as

gateways at level two, and powerful nodes referred to as *hubs* at level three. Given a large universe of sensor types, each mote is equipped with a relatively small subset of sensors. Only gateways and hubs play the role of cluster heads (CHs) – their primary tasks are to forward packets and provide in-network processing.

Network traffic is dominated by packets that flow periodically from many sensors to a small number of sinks, and this traffic pattern may exist simultaneously in many regions of the network. At any given time, there could be one or several sinks, at random locations, for random durations, with the same or different queries, but motes should be allowed to sleep as much as possible. Furthermore, as the scale of the network increases in terms of the number of sinks, queries, and nodes, the costs in terms of energy, bandwidth and memory at the CHs increases. In a large complex network, flooding the entire network with every query would reduce the network's capacity to deliver data from the motes and, where few motes actually have matching data, would force the motes to expend energy without producing useful information for the application.

To reduce communication costs in large data gathering sensor networks, we use proactive and reactive approaches, and a combination of techniques based on clustering, data-centric routing, and Bloom filters. We use our Distributed Asynchronous Clustering protocol [49] presented in Section 6.3.3 with pre-emptive recruiting (DAC-PR) to generate a three tier network of motes, gateways, and hubs. This protocol generates networks with a topology, as previously shown in Figure 6.4, that naturally supports the in-network processing of named data from motes, and the routing of queries by Bloom filter enabled CHs to motes with matching data.

CHs use Bloom filters to represent sensor meta-data and provide a gradient for current queries to follow and, therefore, make better routing decisions in large data-centric networks [48]. Gradients reduce demands on bandwidth, memory, processing, and on the critical radio link between each mote and its cluster head. Consequently, given resource constraints, the network can be expected to serve more sinks for longer and with lower latency.

Section 7.2 discusses WSNs and some design considerations. Section 7.3 provides background information on routing, and in Section 7.4 we present our Bloom Gradient

Routing (BGR) protocol. Sections 7.6 presents our experimental setup and 7.7 presents our experimental results. And finally, our conclusions.

7.2 Congestion in Wireless Sensor Networks

The most essential function of a network layer protocol is to route packets from source to destination. However, unlike traditional address-centric networks, most routing in WSNs is data-centric, highly resource constrained, and vulnerable to congestion.

When the number of packets in a network is less than capacity, the number delivered is proportional to the number sent. However, when too many packets are transmitted, congestion occurs at one or more routers and performance declines; at very high traffic levels almost no packets will be delivered [144]. From the application's point of view, as the network becomes more congested, latency increases. The underlying cause may be related to several constraints: bandwidth, memory, and processing speed. For example, if packets arrive from multiple sources faster than they can be processed, the length of input and/or output queues will grow. If queue lengths exceed available memory, packets will be dropped, but simply adding more memory may make matters worse. By the time a packet gets to the front of its output queue, the sender will have timed out, perhaps several times, and sent one or more duplicates. In fact, Nagle showed that networks with infinite storage, first-in-first-out queueing, and finite packet lifetime will drop all packets when overloaded [108].

Simply upgrading the processor or increasing bandwidth often just shifts the bottleneck [144]. In BGR, we allocate memory for sparse Bloom filters to reduce the load on network components – especially each mote's energy source.

7.3 Routing in WSNs

For most data gathering applications, the dominant communication pattern is expected to be from many data sources to one sink via a data gathering tree, but entire subtrees may be excluded from some queries and, thereby, reduce costs associated with query

dissemination [89].

While communication between one sink and addressable motes may be useful for some applications, e.g., automobile parking lots [142], this chapter tackles the problem of efficient query dissemination in the data-centric context – sinks are interested in specific sensor data and its location, but not the address (ID number) of its source.

BGR combines aspects from *proactive*, *reactive*, *address-centric*, and *data-centric* routing with Queued Bloom Filters. Next we provide some background on these routing techniques before presenting BGR in Section 7.4 and Queued Bloom Filters in Section 7.5.

7.3.1 Proactive versus Reactive Routing

A central focus of research on ad hoc networks has been the design of *proactive* and *reactive* routing protocols. A proactive protocol would continuously compute routes between all sinks and all motes. A reactive protocol would only start a route computation when a source needs to send a packet to a known destination [30]. Proactively maintaining routes between all sinks and all motes is pointless for data-centric applications because when the sink injects a new query into the network, it does not know the destination addresses. On the other hand, reactively computing the routes between each sink and all motes with matching data for every query would be costly in terms of repeated flooding and latency. In [30] the authors suggested that a combination of proactive and reactive protocols may overcome the disadvantages of each approach.

7.3.2 Address-Centric versus Data-Centric Routing

Address-centric routing routes packets from one addressable node to another via the shortest path according to a distance metric. For example, consider Table 7.1. In a flat network, where the number of sinks is assumed to be proportional to the number motes N, the cost of route computation grows $O(N^2)$ and routing table size at each CH grows $O(N)$. Even if motes have globally unique identifiers, the sink has no way of knowing which of the thousands, or more, of nodes have data of interest and, therefore,

address-centric routing is not feasible for query dissemination in large scale WSNs.

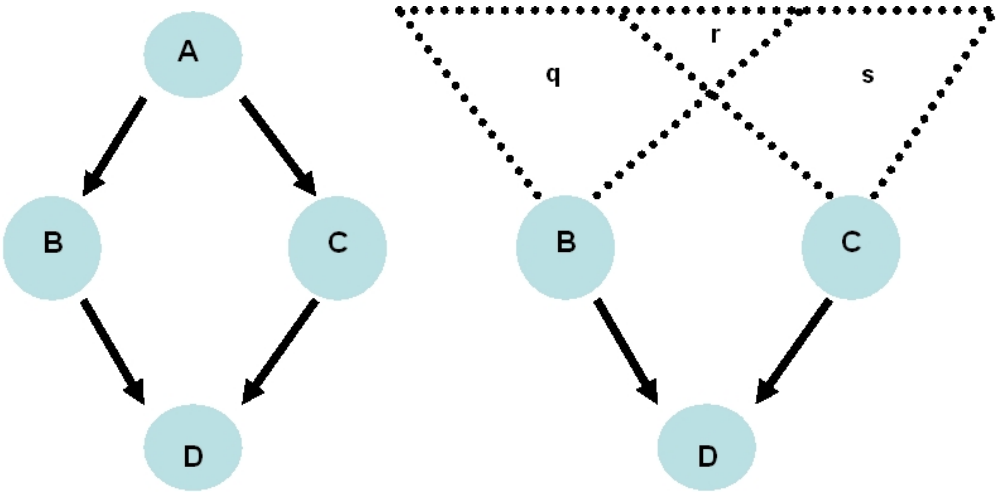
Data-centric protocols are better suited to one-to-many networks where the application injects queries and gathers data from many unknown sources in the network, and where traffic may be reduced by data-aggregation. However, in order to build routing tables, the protocol must first discover which nodes are likely to have data of interest.

Data-centric protocols such as Directed Diffusion [64] use *flooding* to discover nodes with matching data where application queries in the form of *named data* are sent to all sensor nodes. For example, $query = \{type = thermometer, operator = GT, value = 25, location = lab\}$. Matching data flows back along reverse paths to the sink, which reinforces the best paths. The resulting reverse multi-cast trees provide paths that facilitate in-network processing such as data aggregation.

However, flooding the network with each new query (or sensor data) does not scale to large WSNs. First, it is not efficient due to implosion, overlap, and resource blindness [79]. Figure 7.1(a) shows that implosion occurs when a node receives multiple copies of a packet from the same sender, and Figure 7.1(b) shows that overlap occurs when a node receives partially redundant sensor data. Resource blindness occurs when packets are forwarded without regard to energy constraints. Second, after route discovery, each CH must maintain information about each unique query to make forwarding decisions; most nodes will have limited memory for routing tables regardless of the number of unique queries and network size.

Query Dissemination: Table 7.2 shows a hypothetical data-centric routing table where the next hop en route to the motes with matching data depends on the sink's query rather than on the destination's address. Table size increases with the number of unique queries, which could be extremely large. Given a network of N motes and a set of valid queries proportional to N, the routing table at each CH grows $O(N)$ to provide next hop information for each unique query.

Data Delivery: Data-centric routing could also be used for data delivery to multiple sinks. For example sensor data $T = 22$ would first be translated to $T = room\_temperature$, then mapped to the corresponding table entry, which points a list of next hops and forwarded to each of them – a technique similar to multicast [144].



(a) (b)

Figure 7.1: **(a) Implosion**: After node A sends a packet to all of its neighbors, node D receives two copies. **(b) Overlap**: Sensors B and C take measurements in an overlapping area and flood their data. Node D receives two copies of data named r.

7.3.3 Data-Centric Routing with Bloom Filters for Query Delivery

In our network, after sensor data has been reported, translated, stored and diffused, we use data-centric routing for query dissemination. Each query packet

$$packet = \{sinkID, query\}$$

contains an interest in named data sent from a sink address to **unknown destinations**. Each CH tests this named data against Bloom filters to identify zero or more next hops. As Table 7.3 shows, we use one Bloom filter for each neighbor. If the named data tests positive against its Bloom filter, then the CH forwards the query to that neighbor.

7.3.4 Address-Centric Routing for Data Delivery

We use address-centric routing tables for sensor data delivery. Each data packet

$$packet = \{sinkID, data\}$$

contains sensor data sent from a mote or motes to a **known destination**, i.e., a sink

with a known address. Query dissemination has already left a reverse path to that sink, and each CH on that path uses the destination sink ID to identity the next hop.

Table 7.1: Address

destination	**next hop**
Sink1	neighbor1
Sink2	neighbor2
Sink3	neighbor3
Mote1	neighbor4
Mote2	neighbor5
Mote3	neighbor6

Table 7.2: Data

destination	**next hop**
Query1	neighbor1
Query2	neighbor2
Query3	neighbor3
Sink1	neighbor4
Sink2	neighbor5
Sink3	neighbor6

Table 7.3: Bloom Filtered

destination	**next hop**
BloomFilter1	neighbor1
BloomFilter2	neighbor2
BloomFilter3	neighbor3
Sink1	neighbor4
Sink2	neighbor5
Sink3	neighbor6

7.4 Bloom Gradient Routing

Bloom Gradient Routing (BGR) is a data-centric protocol that proactively diffuses sensor meta-data to build gradients at each CH (gateway or hub), reactively disseminates queries to *a priori* unknown motes such that matching data flows back to that sink. For query dissemination, each CH needs just enough information to transmit queries toward motes with data that matches the query; it does not need to maintain the end-to-end path or even explicit next hop information for each mote.

To support BGR, we use a form of Bloom filters which are built at gateways and forwarded to hubs. We discussed Bloom filters extensively in Chapter 3. In this chapter, where bandwidth and energy consumed by radio communication are our primary concerns, we use compressed Bloom filters (CBFs), presented in Section 3.4, which are sparse and optimized for transmission size z, not memory space m. An additional motivation for using CBFs is that after a hub receives two or more CBFs, it can decompress and OR them such that the result is a Bloom filter with an acceptable false positive rate.

To provide additional support for BGR, we maintain Bloom filters in a queue. This data structure, which we call a Queued Bloom Filter (QBF), is used by CHs to discard stale data and indicate the relative age of recent Bloom filters.

This section provides conceptual view of BGR, whereas Section 7.5 will discuss Queued Bloom Filters.

7.4.1 Named Data for BGR

Each mote offers a set of services defined by its sensors, but we assume that only a small subset of all valid sensor types (possibly thousands) are installed on each mote. For example, in [38] each mote was equipped with a weather board composed of six sensors: light, temperature, humidity, solar radiation, photosynthetically active radiation, and air pressure.

Motes report their data in the form of attribute-value pairs. For example, after sampling its environment (an office), and measuring an air temperature of 22 degrees centigrade, it would report $T = 22$ to its CH. However, queries for specific measurements are not likely to be useful for most applications – especially not for monitoring temperature in an office building – and Bloom filters require named ranges, which are part of the universe of meta-data elements that may be stored in them.

In order to implement range queries and categorical queries, an application specific vocabulary must be defined such that sinks and CHs use the same *vocabulary* for useful ranges of values. For example, in an office building monitoring application, room temperature might be defined as $room\_temp : 20-24$ degrees centigrade, and $query = \{T = room\_temp\}$ or, for measurements outside of room temperature range, $query = \{T \neq room\_temp\}$. In the latter case, the CH tests its Bloom filters for temperature ranges other than $room\_temp$ by iterating over its temperature vocabulary.

In addition, a vocabulary should be defined for time. For example, if the sink wants "hot" temperature data from the current epoch "zero", the query would be formulated as $query = \{T = hot \wedge TIME = zero\}$. Naturally, a more fine grained naming system could be defined, depending on the application, but the details of such a system are not essential to this thesis. BGR simply requires that ranges for measurements, time, and other named data, have standard definitions and query formats in order to make effective use of Bloom filters.

Rather than simply flood the network with queries, BGR proceeds in three phases: sensor diffusion, data dissemination, and data delivery.

7.4.2 Phase One: Sensor Diffusion

The first phase of BGR is *sensor diffusion*: sensor meta-data is proactively diffused "up" the hierarchy. This phase starts when each mote transmits named data to its gateway. Figure 7.2 shows a gateway receiving named data from three motes. The gateway translates this sensor data into meta-data. Each measurement value is binned and tagged – the source mote id number is been prepended, .e.g., $id : T = 22$. At this point the data is in two forms: meta-data and tagged meta-data. Then the gateway uses Algorithm 3 to store both forms in its local filter, which is queued, and the untagged meta-data in its sparse Bloom filter that represents all motes in its cluster.

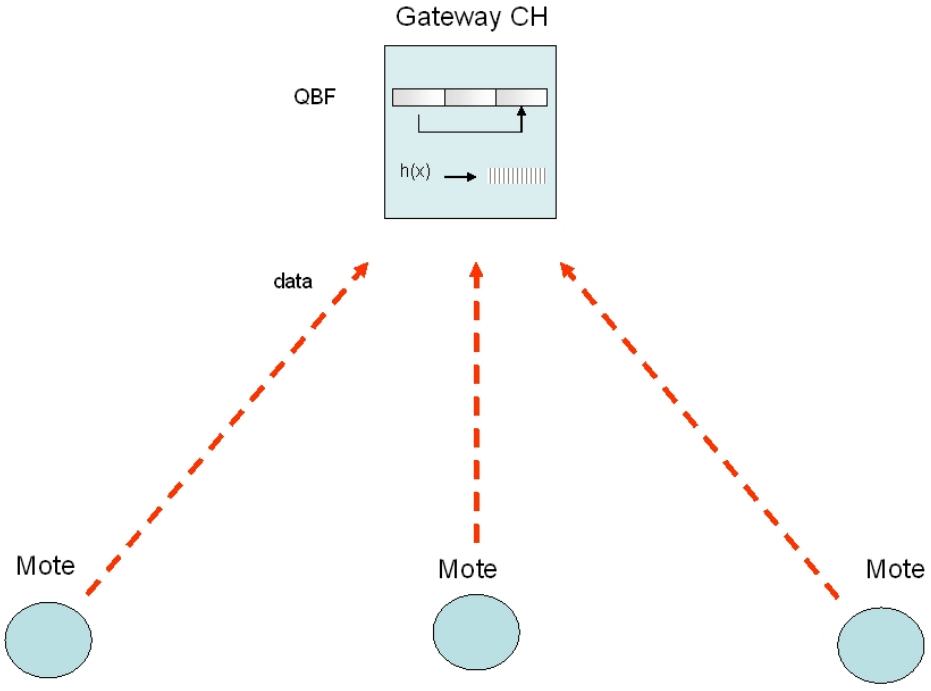


Figure 7.2: **Motes and a gateway enabled with Bloom filters**: Motes generate named data to represent their services and sensor measurements. Each gateway stores information in a Bloom filter at the back of its QBF, and in a sparse Bloom filter which is compressed and forwarded to its hub.

At the end of the current epoch, the gateway forwards a compressed copy of the sparse filter to its hub. Figure 7.3 shows a hub receiving compressed Bloom filters from its four gateways. These filters represent untagged meta-data. Then the hub multicasts a Bloom filter for the most recent epoch to all other hubs. The hubs exchange

Algorithm 3: Pseudo code for storing a set of meta-data in a Bloom filter.

Input: Set of elements: meta_data, Bloom filter: bloom_filter

```
foreach element in meta_data do
    foreach hash in Hash do
        bloom_filter[hash(element)] = 1;
    end
end
```

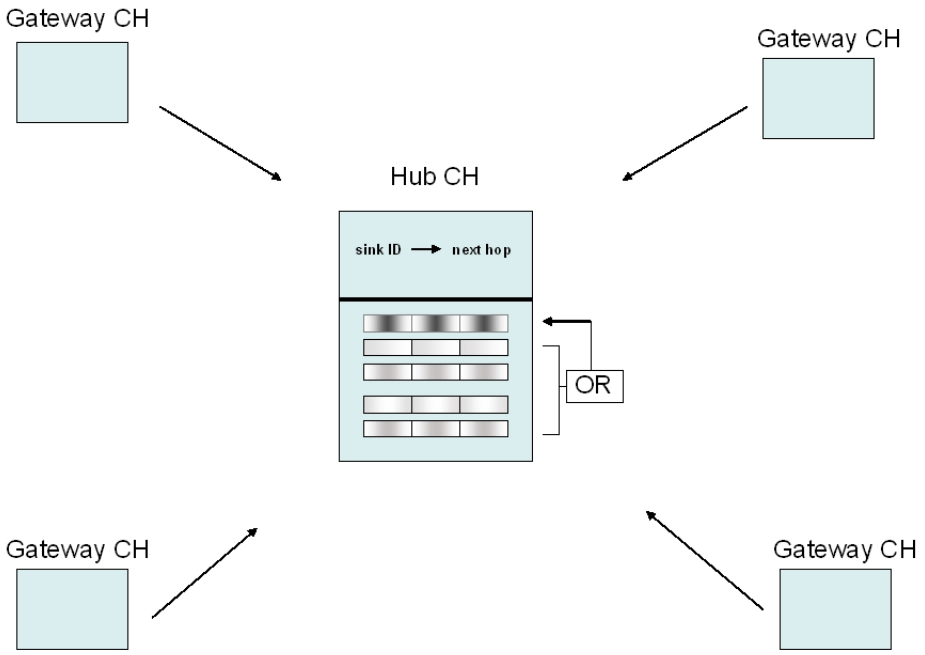


Figure 7.3: **Hub and Gateway CHs:** Gateways transmit compressed Bloom filters to their Hub.

compressed Bloom filters such that all hubs have probabilistic information about what named data is available via any other hub.

7.4.3 Phase Two: Query Dissemination

The second phase of BGR is *query dissemination*: queries are reactively disseminated along Bloom filter gradients to motes with matching data. Figure 7.4 shows a three tier network where sink $S1$ sends a query to nearby hubs, and the query follows a path based on gradients at H1, H2, H3 and G1 to mote M6.

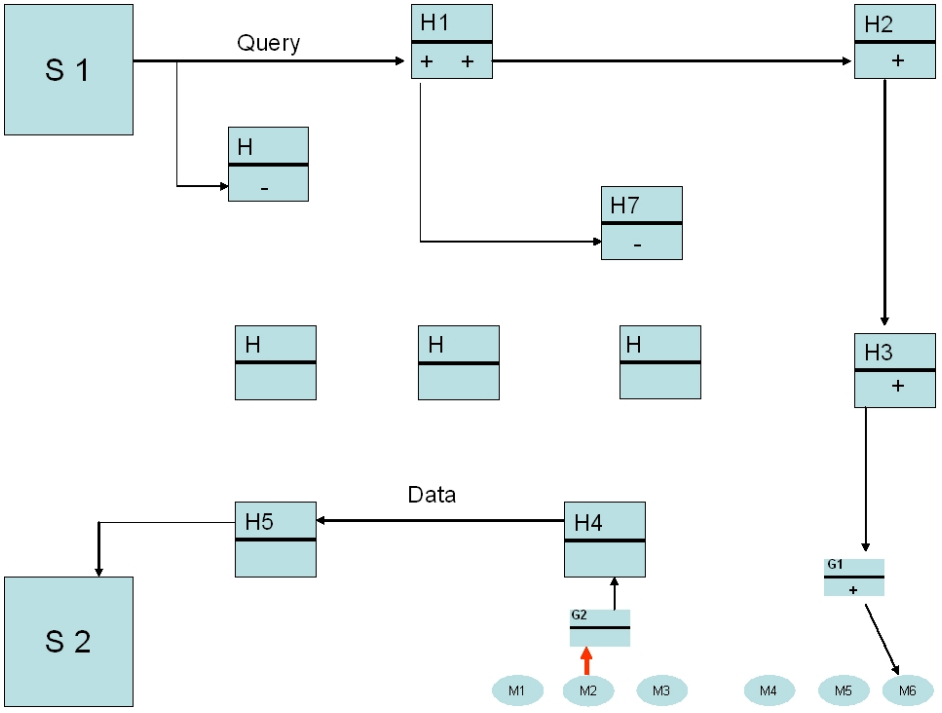


Figure 7.4: **Hub Backbone:** Hubs use Bloom filters to route queries to other hubs. Each hub also uses Bloom filters to route queries to its gateways. Hubs use address-centric routing to deliver data packets received from their gateways to the query source (sink).

Dissemination proceeds in two basic steps. First, when each hub receives a query, it must decide which neighbors (hubs within range) to forward the query to. For example, given $query = \{M = low \wedge T = high \wedge S = high\}$, the receiving hub uses Algorithm 4 to test all elements in the query against the Bloom filter for each neighbor hub.

Algorithm 4: Pseudo code for testing a Bloom filter.

Input: Set of elements: query, Bloom filter: bloom_filter
Output: Boolean

```
foreach element in query do
    foreach hash in Hash do
        if bloom_filter[hash(element)] == 0 then
            return false;
        end
    end
end
return true;
```

The order that elements are tested does not matter, and the query tests negative as soon as an element hashes to a zero in the BF. If the query tests positive against a neighbor hub's BF, then data of interest probably exists in that direction, and the query is forwarded to that hub.

Second, each hub tests its gateway BFs, and forwards the query to each gateway whose BF tests positive. However, the gateway may receive many similar queries from its hub and, in some sense, provides a *gatekeeper* service to the motes in its cluster. This service filters out redundant transmissions to the motes, and only transmits queries to a mote when the query asks for something more or different from the service currently provided.

Although queries are expected to test negative for most neighbors, false positives will cause some queries to be forwarded in error. For example, when $f = 0.05$, about 95% of query packets are forwarded toward motes that sent matching data, while only $\approx 5\%$ of unique queries are forwarded toward a random subset of motes that did not. This would equate to a low rate of flooding.

The expected false positive rate f is a tunable parameter and may be set arbitrarily low by allocating more memory to the Bloom filters. Nonetheless, we note that for some applications the trade-off between memory cost and the transmission cost of false positives should be considered. First, given a fixed amount of memory, allocating memory for Bloom filters reduces the amount of memory available to other tasks. Second, queries that test positive are unlikely to travel more than a few hops (due to error) because the probability of h false positive hops, $p_h = f^h$, decreases exponentially with h.

7.4.4 Phase Three: Data Delivery

The third phase of BGR is *data delivery*: matching sensor data is delivered via reverse address-centric paths from motes to query sources.

In the absence of queries from one or more sinks, which provide instructions, each mote samples its environment and transmits data to its gateway according to default parameters. To maximize network lifetime, the defaults for sampling and reporting are

set at the minimum rate necessary to support sensor diffusion.

The sink address for each query packet is stored in a table at each CH en route to motes with matching data. After a mote receives a query from its CH, it examines the query and replies with matching data. The CH may do some in-network processing on data received from its motes, update its Bloom filter, then forward data of interest to its hub. Hubs use address-centric routing to forward data packets to the sink that injected the query into the network.

7.4.5 Space Required for Named Data

Although the number of unique queries may be extremely large, they are constructed from a limited vocabulary of named data. For Bloom filters, the set of named data to be stored equates to a set of elements S drawn from the universe of elements U, where S is a small subset of U. The space required for each Bloom filter at the CHs is $O(|S|)$, the number of elements we expect to store, and does not depend on network size or element size.

After sensor diffusion, query elements are tested against Bloom filters at hubs and gateways. For example, suppose a query expresses an interest in measurements from sensors with high carbon monoxide (CO), humidity (H), and temperature (T), but low light (L) readings: $query = \{CO = high \wedge H = high \wedge T = high \wedge L = low\}$. According to BGR, this query must be decomposed into its "atomic" elements in order to be tested against a Bloom filter.

If queries may be built from $1..n$ unique elements, then $2^n - 1$ unique queries are possible, but at most n elements of sensor meta-data need to be stored in each Bloom filter. Also, many queries will have common elements. For example, thousands of queries might include an interest in high temperature data $T = high$, plus a few other attribute-value pairs, but the interest in sensor data $T = high$ would be just one element for the Bloom filter – there may be any number of queries for that one element but very few for a specific combination of elements in the form of attribute-value pairs.

7.4.6 Routing Summary

Sensor diffusion optimizes Bloom filters for transmission, not storage. They are transmitted "up" from gateways to hubs where they are exchanged. During query dissemination, sink ID and previous hop information may be stored at each CH in a routing table or Bloom filters optimized for storage. Data delivery uses address-centric routing where sensor data follows a reverse path of sink addresses.

Table 7.4 provides a summary of using Bloom filters for address-centric and data-centric routing. Standard Bloom filters, as presented in Section 3.3, are optimized for storage in memory. Compressed Bloom filters, as presented in 3.4, are optimized for transmission. Where resource constraints have no impact on performance, we can use conventional routing tables. If resources constraints cause performance to drop below an acceptable level, a form of Bloom filter may be used.

Table 7.4: Optimal BGR query dissemination must consider the number of destinations. Routing tables (RT) or Bloom filters (standard BF, compressed CBF) may be used for address-centric (AC) or data-centric (DC) routing.

optimize	**destinations**	**AC**	**DC**
storage	few	RT	RT
	many	BF	BF
transmission	few	RT	RT
	many	CBF	CBF

7.5 Queued Bloom Filters

To efficiently route queries to motes that provide a specific service, instead of all motes, timely information about each mote must be available at all gateways and hubs. Conventional routing tables are updated periodically such that new entries are created and old entries are either refreshed or deleted. In contrast, a standard Bloom filter does not support the delete operation. Add is an idempotent operation – no matter how many times an element is added (stored), the result is the same and there are no side effects, but delete would make false negatives possible.

Since standard Bloom filters do not support delete, they may represent stale information. After changes in the physical environment, network connectivity, sensor measurements, sensor availability, and so on, Bloom filter gradients will become a source of misinformation. One option is to reset them periodically, but that would discard fresh data along with the stale data.

Counting Bloom filters [15], which have more than one bit per cell, support the delete operation but the counters may overflow and the delete operation may cause false negatives. In BGR, such counters could easily overflow because the same elements of meta-data may be reported many times by motes to their cluster head, and Bloom filters are unioned at the hubs. Even if the risk of false negatives is ignored, the meta-data that was stored in the Bloom filter no longer exists, which means there is no way to determine which bits are stale and should be set to zero.

In this application, queries are for all matching data, regardless of distance from the sink – measurement data from nearby motes is not necessarily preferred over data from faraway motes. Consequently, attenuated Bloom filters, presented in Section 4.5.1, are not suitable. BGR disseminates queries based on the existence of matching data, not quantity, and only utilizes one bit per cell.

BGR uses a Queued Bloom Filter to solve the stale data problem. A Queued Bloom Filter is a compound data structure where a set of Bloom filters is ordered according the epoch during which they were built. Algorithm 5 shows the pseudo code for creating a QBF composed of q Bloom filters with m bits of memory each.

Algorithm 5: Pseudo code for creating a new QBF.

```
Input: Integer: q, Integer: m
Output: Queue of Bloom filters: QBF
for i=0 to q do
|   QBF[i] = new BF(m);
end
return QBF;
```

The queue of q Bloom filters is managed such that the oldest filter is at the front and the current filter (newest) is at the back. New elements are only stored in the current BF. At each time step, the oldest BF is removed from the front, existing BFs advance

toward the front, a new BF is created, initialized to all zeroes, and added to the back of the queue.

At any point in time, we can query the queue for measurements of a certain age. Algorithm 6 shows the pseudo code for testing a queue of Bloom filters. Each BF is associated with statistics including, the number of elements stored n, and their min, max, mean and standard deviation. Min and max are especially useful for reducing the rate of false positives at gateways, which also reduces latency by making the system more responsive.

Algorithm 6: Pseudo code for testing a QBF.

```
Input: Set of elements: query
Output: List of integers: list

initialization;
max ← QBF.length - 1;
for epoch=0 to max do
    bloom_filter = QBF[epoch];
    found ← test(query, bloom_filter);
    if found == true then
        list.add(epoch);
    end
end
return list;
```

Both gateways and hubs server as CHs for the motes their cluster (hubs also serve as "gateways" for their own cluster of motes). Epochs are ordered in terms of recency. Therefore, the most recent data is generated in epoch 0 and stored in each CH's current Bloom filter, QBF_0. Previous data, generated in epochs $1..q-1$, is stored in $QBF_{1..q-1}$.

Each hub maintains a set of QBFs to represent data received from its gateways and other hubs for epochs $1..q-1$. In contrast, each gateway maintains a QBF to represent data from the motes in its cluster only. At the end of each current epoch, gateways send a copy of QBF_0 to their the hubs. The hubs exchange those filters (from the previous epoch) and store them in QBF_1.

QBFs represent sensor-data received during each epoch, and they are used to *suppress queries at the hubs* where there is no evidence in QBF_1 of matching meta-data

at other the hubs or the hub's gateways, and to *suppress mote queries at hubs and gateways* unless QBF_0 or QBF_1 tests positive. Thus, query suppression may be adjusted. For example, it may be reduced by also accepting older evidence in QBF_2.

In addition, Bloom filters built during past epoch provide probabilistic evidence of historical conditions in the network. Here are two examples. (1) If a hub QBF tests positive for the existence of high temperature for each of the last three epochs, then forward. (2) If a hub QBF tests positive for the existence of low temperature in one epoch followed by high temperature in the next epoch, then forward.

7.6 Experimental Setup

In our three tier network of $N = 875$ nodes, about 95% of the nodes are motes, about 5% of the nodes are gateways or hubs, and about 1% of the nodes are hubs. The network has self-organized such that short paths, $O(lgN)$ hops, exist between any mote and any sink and, consequently, the small-world effect is possible assuming that each node has enough information to identify the next hop. While the number of sinks $S = 4$ is relatively small, BGR has been designed to scale to larger numbers of sinks and/or motes by exploiting space and bandwidth efficient techniques.

7.6.1 Queued Bloom Filters

In our simulations, each gateway received data from an average of 30 motes once every time period. For most Bloom filter applications, the goal is to minimize the false positive rate f given an available memory space of m bits and the number of elements to be stored n, where n depended on the number of sensor types on each mote and whether or not each measurement was also stored with the mote's id number. The optimal number of hash functions $k = \frac{m}{n} \ln 2$, as presented in Section 3.3.

BGR uses compressed Bloom filters, as presented in 3.4, which require more memory than necessary for a Bloom filter optimized for storage given a maximum acceptable false positive rate. In fact, this application favors the use of sparse Bloom filters for several reasons.

First, the goal is to optimize the Bloom filters for transmission size, no greater than z bits compressed, and with a false positive rate no greater than f. This means using using a smaller number of hash functions and a larger m such that the Bloom filter is as sparse as possible without exceeding the limit on false positives. Given m and z, Equation 3.9 determines how sparse the uncompressed filter must be in order to compress down to no more than z bits. Equation 3.10 determines the number of hash functions that will minimize f given these constraints.

Second, n may vary from gateway to gateway but their Bloom filters must be the same length to be bitwise OR'd at the hubs. The implication is that even if we did not want to compress Bloom filters for transmission, we would still want them to be sparse.

Third, the union of received Bloom filters must not be too saturated otherwise an unacceptably high false positive rate will result. Memory should be allocated for q Bloom filters such that application specific requirements for epochs of historical data q, false positive rate f, and transmission size z are met at hubs and gateways.

7.6.2 Searching Hierarchical Clusters

Hubs receive sparse Bloom filters from their gateways and other hubs and combine them with bitwise OR operations. Therefore, Bloom filters at the hub level are more saturated than filters at lower levels, and the probability of a false positive decreases as the descent proceeds [82]. The descent down a branch stops when the target is found or when a filter tests negative. In other words, filters are used to perform a probabilistic search of a hierarchical structure.

7.6.3 Using Memory to Reduce Communication Costs

In this context, trading memory for energy and bandwidth is an effective strategy for the following reasons. (1) The trend for mote technology indicates that memory capacity is expected to grow faster than energy capacity [88]. (2) Sensor diffusion entails a much smaller number of destinations than flooding queries. (3) Computing and transmitting

Bloom filters requires less energy than transmitting many queries. This efficiency gain is critical at the gateway to mote link because receiving a query packet requires almost as much energy as transmitting one over a short distance [101, 100]. (4) A WSN that serves complex queries implies that most Bloom filters will test negative during query dissemination, thereby greatly reducing energy and bandwidth consumption by radio transmissions. (5) Furthermore, radio range understates the benefit of not transmitting a query. The radius of interference is about two times radio range [151]. (6) Fewer transmissions means energy constrained motes can sleep more or execute a useful task, and other nodes are free to do in-network processing, etc., and thereby reduce latency.

7.7 Experimental Results

The design of BGR was driven by the utility of pervasive information about sensor types and recent measurements. This information, represented by Bloom filters, is much more compact than named data or meta-data. After Bloom filters have been proactively diffused in a hierarchical manner, a query can be answered by nearby proxy nodes, or forwarded without the delay and overhead of computing routes between all sources and destinations.

7.7.1 Query Suppression

Sensor diffusion, where sensor meta-data is sent by gateways to hubs, will be more energy efficient than flooding the entire network with a large number of queries if the cost of sensor diffusion is less than the benefit of query suppression. It is worth noting that the number and specificity of queries does not depend on the cost of diffusion. Therefore, given no false negatives, the net benefit of BGR is proportional to the degree of query suppression, which depends on how selective the queries are in terms of sensor types, measurement ranges, and freshness of the data as represented by the QBFs.

In this section we consider three scenarios executed on a simulated WSN of 875 randomly deployed nodes. The air temperature T and humidity values H at each node are unknown to the sink but are, in fact, spatially correlated and normally distributed

within each cluster. The statistics for the air temperature of the entire network are mean 23.67, minimum 0.67, maximum 48.97, and a standard deviation of 11 degrees centigrade. The statistics for the humidity of the entire network are mean 47.29, minimum 3.38, maximum 100, and a standard deviation of 22.08. However, all queries are for integer measurements taken during the current epoch, i.e., a query for $T = 22$ should be forwarded to all motes with recent temperature measurements in the range of 22.0 to less than 23.0.

First, a query for all data – the query should be sent to all motes. Figure 7.5 shows the data aggregation tree that results when no queries have been suppressed. This is equivalent to all filters testing positive.

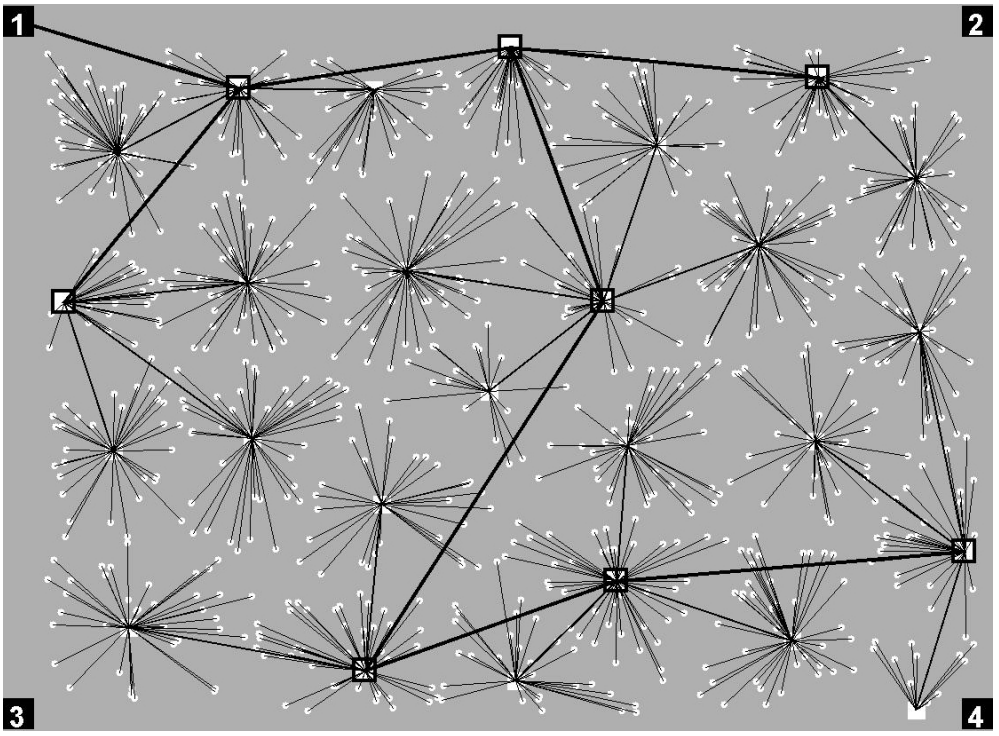


Figure 7.5: **Three tier network of motes, gateways, and hubs:** The data aggregation tree shows that the query has been forwarded to all hubs and gateways (CHs), and to all motes.

Second, a query for a range of temperature data. Figure 7.6 shows the aggregation tree generated for the query $\{T = 21 \vee T = 22 \vee T = 23\}$. The Bloom filters suppressed the query such that it was forwarded to a small subset of cluster heads and motes.

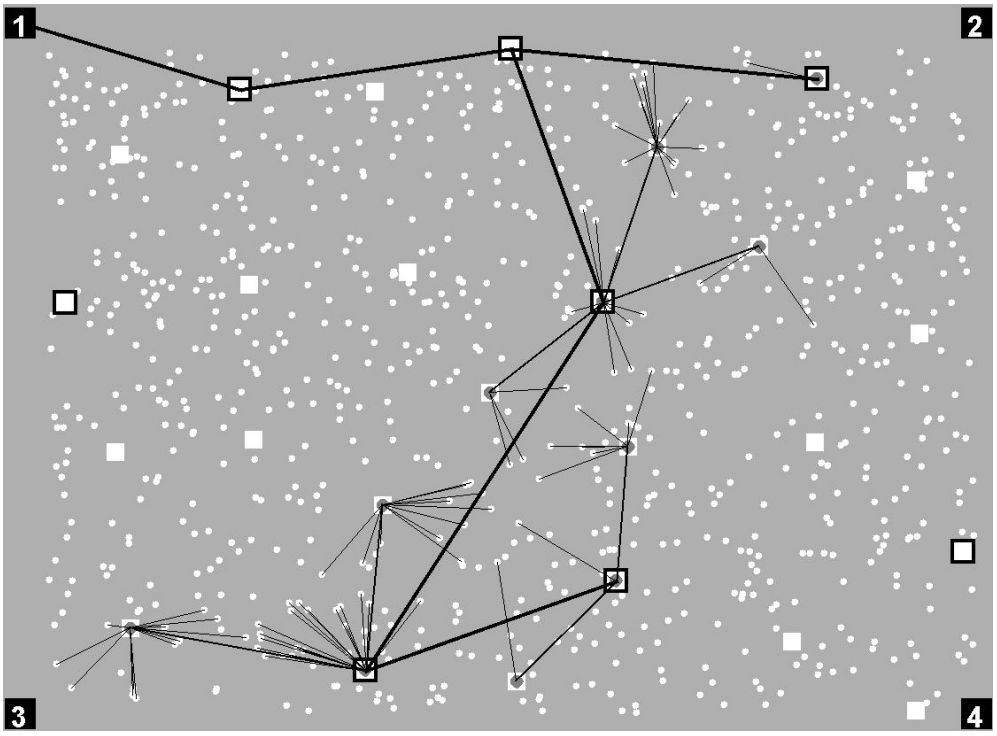


Figure 7.6: **Three tier network of motes, gateways, and hubs:** Bloom filters are maintained at gateways and hubs. This query has been forwarded to a small subset of CHs (squares with dark center), and a small number of motes.

Third, a complex query for a specific combination of temperature and humidity data. Figure 7.7 shows the aggregation tree generated for the complex query $\{(T = 21 \vee T = 22 \vee T = 23) \wedge H = 46\}$. The Bloom filters suppressed this query such that it was forwarded to an even smaller number of cluster heads and motes.

Naturally, the size of the data aggregation tree for each query depends on the distribution of queries and the spatial distribution of measurements in the network. Therefore, as shown above, the cost savings provided by BGR will vary greatly – from 0%, where the query is forwarded to all hubs, gateways, and down to all motes because all motes have matching data, to nearly 100% where very few motes have matching data. Given a realistic set of queries for specific data and a large network equipped with many sensor types, BGR will be far more efficient than any protocol that relies on flooding for query dissemination.

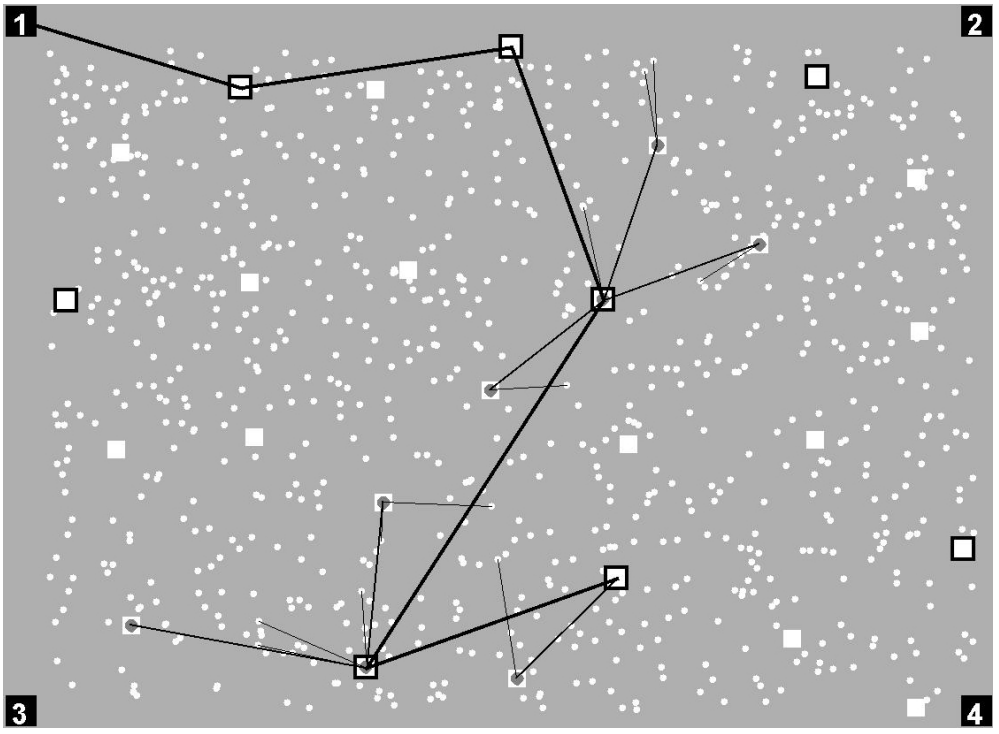


Figure 7.7: **Three tier network of motes, gateways, and hubs:** Bloom filters are maintained at gateways and hubs. This query has been forwarded to a smaller subset of CHs (square with dark center), and a smaller number of motes.

7.7.2 Net Benefits of Query Suppression

Suppose one subtree with a hub at its root. The cost of flooding may be defined as the number of links times the number of bits per transmission. However, since flooded queries and Bloom filters must be transmitted over the same number of links, whether we consider the entire subtree or just the CH to mote links, the number of links can be ignored. Therefore, given a fixed cost for Bloom filter diffusion, we only need to consider the number of queries that would be sent to motes without matching data, in other words, the number of nonmatching queries.

Optimal filter size m depends on the number of unique elements n to be stored and desired false positive rate f. Suppose one sensor measurement from each mote in a cluster of $n = 30$ motes, and a maximum acceptable false positive rate of $f = 0.04$.

$$m = \frac{n \lg(f)}{\lg(0.6185)}$$

A probabilistic representation of this set of elements would require $m \approx 201$ bits.

Bloom filter diffusion will be more efficient than flooding queries when the number of nonmatching queries x is greater than a threshold number, i.e., the cost of diffusion in terms of queries. If each query requires $q = 10$ bits and each Bloom filter requires $m = 201$ bits, and the false positive rate $f = 0.04$, then

$$m + (f * x * q) = x * q$$

and at least $x \approx 21$ nonmatching queries must be sent to a CH during each epoch in order for the Bloom filter diffusion yield a net reduction in the number of bits transmitted on the CH to mote radio links.

7.7.3 Query versus Transmission Suppression

Bloom filters save energy by reducing radio transmissions and receptions, between hubs, hubs and their gateways and, most critically, between gateways and their motes. In BGR, forwarding decisions are based on whether or not each neighbor tests positive for a query. If none test positive, then the packet is dropped. If at least one neighbor tests positive, then the packet is forwarded to those neighbors only. However, in a shared medium, a transmission to one is a transmission to all neighbors. This suggests that no savings are possible, energy or otherwise. In practice, there are three cases; we list their efficiency gains from largest to smallest.

The first case: no neighbors test positive, and no transmissions or receptions occur. **The second case**: a subset of neighbors test positive. Efficiency gains are possible because the neighbors that overhear a transmission, which is not addressed to them, can ignore it and do some useful work or sleep; in addition, the CH will only need to transmit at a power level sufficient to reach the most distant neighbor in the subset. **The third case**: all neighbors test positive, and the CH transmits to all neighbors. No efficiency gains are possible in this case because no queries were filtered out and, therefore, no transmissions were suppressed.

7.8 Related Work

The characteristics, applications, and communication protocols of WSNs are surveyed in [2, 145]. They cover design issues and techniques proposed for all layers of the network stack. In contrast, [3] is devoted to routing and the network layer – it describes and categorizes the many approaches to routing.

Sensor Protocols for Information via Negotiation (SPIN) [79, 51] is a family of low-energy protocols that extend LEACH [55] to disseminate information. SPIN was designed to address problems caused by flooding and is based on two ideas: (1) communicating raw sensor data is much more expensive than sensor meta-data; and (2) nodes should be energy aware and energy adaptive [54].

In SPIN, large data messages are named using high-level data descriptors, called meta-data, which are assumed to be application specific. Before data transmission, the sensors exchange messages of three types. (1) ADV: advertises meta-data. (2) REQ: requests specific data. (3) DATA: delivers the requested data. This high-level negotiation that examines meta-data to ensure that only data that provides new information is transmitted. They also allow nodes to base routing decisions on application-specific information about the data, which enables large energy savings compared with conventional approaches [52]. However, the advertisement mechanism does not guarantee data delivery because only single-hop neighbors exchange messages and, therefore, the probability of data loss increases with network size.

Directed Diffusion [30, 63, 64, 137] is an important data-centric routing protocol for flat sensor network topologies. It provides mechanisms for routing queries and sensor data. Each sensor uses an attribute-based naming scheme that names the data it generates according to one or more attributes. Initially the sink has no prior knowledge of which nodes probably have matching data. It broadcasts the query to the network, and nodes that receive the query forward it to all adjacent nodes except for the sender, thereby flooding the network.

The interaction of disseminated queries and matching sensor data establish *gradients* for data to flow toward the sink that expressed that interest. For example, a sink may flood the network with an interest in the form:

$type : temperature, op : GT, value : 20$

Sensors with matching data generate attribute-value data pairs in the form:

$type : temperature, id : 12, value : 21$

The data is routed along reverse paths to the query's source. After the sink receives this data, it reinforces some paths to the data sources. Based on the strength of the gradient, intermediate nodes forward sensor data and queries for data along efficient paths. This particular version of Directed Diffusion is known as two-phase pull [50].

A number of other researchers have also found the gradient concept useful [132, 160, 84]. Messages flow through a multi-hop network according to the gradient stored in each node. In some implementations, gradient is based on the "height" of a node, which is proportional to its distance from the sink [132]. In that case, data flows downhill toward a sink, and queries flow uphill toward sensor nodes. However, regardless of implementation, gradient indicates direction, i.e., the next node on a multi-hop path.

Gradient Broadcast (GRAB) addresses the problem of robust data forwarding to the sink via unreliable sensor nodes over error-prone wireless channels [160]. Their sink maintains a *cost field*, and each node maintains an indicator of the cost of forwarding a packet from itself to the sink. Data flows toward lower cost nodes, so the cost field implies a gradient because direction is implicit. However, the cost value at each node is not the same as the gradient (vector) defined in Directed Diffusion. For robust delivery, GRAB uses a mesh of multiple paths from source to sink. The source assigns a *credit* to each report it transmits to control the degree of path redundancy, i.e., more credit translates into a wider mesh of multiple paths.

Gradient-Ascending Stateless Protocol (GRASP) was presented in [84]. In GRASP, a forwarding history is stored at each node on the path from source to sink – each history is represented by a Bloom filter. When a source node transmits a packet, the packet's *origin address* is hashed to a Bloom filter at each node along the multi-hop path to the sink. GRASP was primarily motivated by the theory that membership-based broadcast is more energy efficient than flood-based dissemination. However, because GRASP defines a protocol for routing queries from a sink to specific, addressable sensor nodes, we see it as an *address-centric* protocol. This is in contrast with many

research efforts which emphasize the *data-centric* nature of WSNs.

An interesting paper by Braginsky and Estrin presents *Rumor Routing*, which is proposed as a compromise between flooding queries and flooding sensor data [13]. *Gossip Routing* also provides a scheme that reduces flooding – nodes flood by sending the message to random neighbors and redundant connectivity allows most nodes receive to it [85]. *BARD* [139] uses Bayesian estimates based on prior routing information to limit flooding. Acquisitional Query Processing (ACQP) uses *semantic routing trees* to enable each node to determine whether or not any nodes below need to participate in a query [89].

7.9 Conclusion

We have presented a scalable paradigm for messaging in a WSN where the DAC protocol was used to self-organize the network into clusters and BGR was used for routing queries to unknown motes with matching data without flooding. The sinks, hubs, and gateways used a hierarchy of Queued Bloom Filters filter out unpromising transmissions. In this network, short paths existed and the network was able to find them using BGR – a fully distributed protocol.

The BGR protocol entails three phases. Sensor diffusion proactively pushes sensor data and meta-data stored in Bloom filters to establish probabilistic data-centric gradients. Query dissemination reactively disseminates queries along these gradients to motes with matching data. Data delivery is a process whereby data of interest flows along reverse paths to the sink that sent the query.

BGR was extremely cost effective in the case of selective queries, where little or no matching data existed in the network. Query suppression is valuable because, in terms of energy consumption, *the best transmission is no transmission*. When BGR is implemented on an infrastructure of well separated cluster heads such as generated by DAC, the result is a network that is self-organizing, scalable, adaptive, reliable, and more energy efficient than other data-centric protocols that rely on flooding or random forwarding.

Chapter 8

Conclusions and Future Work

8.1 Summary of Contributions

Large scale WSNs require fully distributed protocols for self-organization and communication. The Distributed Asynchronous Clustering (DAC) protocol was applied to a homogeneous network of resource constrained sensor nodes ("motes"). Although it generated energy efficient clusters, its scalability was limited. The DAC protocol was also applied to heterogeneous networks. It generated hierarchical clusters characterized by well separated cluster heads, and a topology that moved the in-network processing and radio communication burden from battery powered motes to the more powerful gateways and hubs. Consequently, each mote was tasked with duties that matched its capabilities and recent measurements.

In our research on clustering, we found that the key to scalable self-organization is a local feedback mechanism, which we call pre-emptive recruiting. While many real world networks may use other mechanisms to generate networks with small-world properties, our empirical data strongly indicates that pre-emptive recruiting is a viable mechanism for self-organizing real world, large scale wireless sensor networks.

Given a WSN network with small-world properties, we found that the Bloom Gradient Routing (BGR) protocol enables the network to serve queries far more efficiently than conventional protocols that rely on flooding. BGR's superior performance was made possible by using the space efficient properties of Bloom filters, when optimized for transmission size in particular, to suppress useless queries and exploit existing short paths between sinks and motes.

The original motivation for Bloom filters was lack of memory for storage. In the WSN context, we found that Bloom filters are more useful for some tasks when op-

timized for storage, and other tasks when optimized for transmission. In fact, BGR relies primarily of Bloom filters optimized for transmission size.

While the benefits of Bloom filters have been well established in other domains, we conclude that they will be even more useful in wireless sensor networks.

8.2 Future Work

This thesis covered a wide range of material and converged on a few key ideas. Bloom filters were used primarily for query suppression, but have many other possibilities in the WSN domain because nodes must frequently exchange information in order to complete tasks that require cooperation and coordination.

First, more research is needed to develop a protocol for large scale sensor networks where the automatic deployment of more powerful nodes, as presented in Chapters 6 and 7, is not feasible. To provide scalability for homogeneous sensor networks, we propose a version of Stochastic Hierarchical Clustering (SHC) [48] that self-organizes numerous and widely deployed sensor nodes (motes) into a forest of *height limited data aggregation trees* where inter-cluster connectivity is provided by CHs. However, the target network is not entirely homogeneous – it assumes the availability of mobile data collection units ("data mules") with sufficient capacity for query delivery and data collection at the root CH of each data aggregation tree, and data delivery to the sink.

Figure 8.1 shows a hypothetical WSN that has been self-organized into a forest of height limited trees. We used pre-emptive clustering, as presented in Chapter 5, and a beacon range of 100 to generate the clusters; and a preliminary version of SHC to generate the trees. Sensor data flows from each mote its CH, and each CH transmits information, in the form of sensor data or Bloom filters, to its root via a path of child-parent CH links. Each CH maintains a set of Bloom filters and suppresses queries using a variation of the BGR protocol presented in Chapter 7. By testing filters, *first at the root CH* and then at child CHs, each CH can easily determine with a high degree of confidence whether or not an "element" is probably in its cluster or in one of its subtrees without transmitting a query. As a result, queries flow down each tree to clusters with

measurement data of interest, or even to a sensor with a specific identifier.

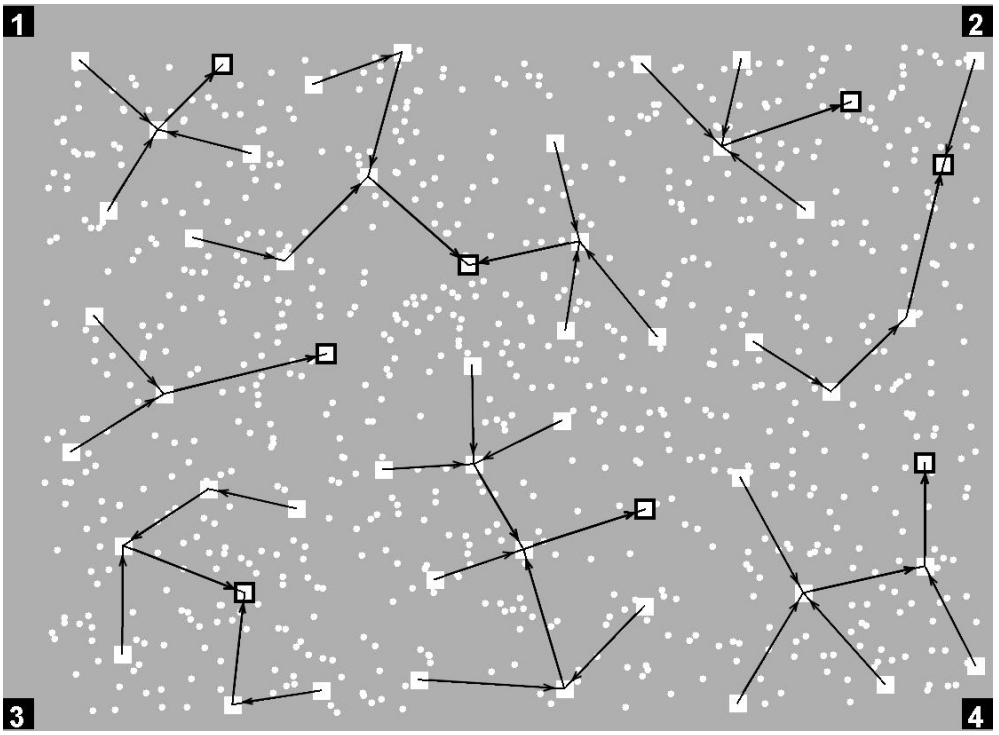


Figure 8.1: **WSN Simulator**: Field of sensors after hierarchical clustering. Motes are white circles, CHs are white squares, and the highest level CH for each tree, "the root", is a black outlined white square. Note: each CH is at most 3 hops from its root.

While SHC appears promising, many details have yet to be defined. For example, what is the optimal height of the data aggregation trees, and how should the data mules cooperate during query dissemination and data collection?

Second, WSNs are notoriously application specific. The relative performance of a protocol will depend multiple factors including the hardware used, the distribution of queries, and the distribution of sensor data. While BGR is expected to offer net performance gains for most applications, if most queries are relevant to most of the sensor nodes, then some form of limited flooding may be more efficient. More research is needed to determine exactly what types of applications are likely to benefit from BGR.

And finally, we noted that Bloom filters require a naming scheme for meta-data, and that application specific naming schemes are an active research area. While element size has no impact of Bloom filter size, the number of elements to be stored

does. Future work in this area might consider the trade-off between memory and communication costs versus the utility at the application level of naming schemes defined various granularities.

Appendix A

Micro-Electro-Mechanical Systems Technology

MEMS is the breakthrough technology that has made low cost pervasive sensor networks possible.

> Micro-Electro-Mechanical Systems (MEMS) is the integration of mechanical elements, sensors, actuators, and electronics on a common silicon substrate through microfabrication technology. While the electronics are fabricated using integrated circuit (IC) process sequences (e.g., CMOS, Bipolar, or BICMOS processes), the micromechanical components are fabricated using compatible ”micromachining” processes that selectively etch away parts of the silicon wafer or add new structural layers to form the mechanical and electromechanical devices.
>
> MEMS promises to revolutionize nearly every product category by bringing together silicon-based microelectronics with micromachining technology, making possible the realization of complete systems-on-a-chip. MEMS is an enabling technology allowing the development of smart products, augmenting the computational ability of microelectronics with the perception and control capabilities of microsensors and microactuators and expanding the space of possible designs and applications.
>
> Microelectronic integrated circuits can be thought of as the ”brains” of a system and MEMS augments this decision-making capability with ”eyes” and ”arms”, to allow microsystems to sense and control the environment. Sensors gather information from the environment through measuring mechanical, thermal, biological, chemical, optical, and magnetic phenomena. The electronics then process the information derived from the sensors and through some decision making capability direct the actuators to respond by moving, positioning, regulating, pumping, and filtering, thereby controlling the environment for some desired outcome or purpose. Because MEMS devices are manufactured using batch fabrication techniques similar to those used for integrated circuits, unprecedented levels of functionality, reliability, and sophistication can be placed on a small silicon chip at a relatively low cost. [http://www.memsnet.org/mems]

Appendix B

Radio Propagation Models

Radio propagation models are used to predict the received signal strength (RSS) at a receiver. In order for a packet to be received, the RSS must be $\geq$ to the device's threshold T. If the RSS less than T at the physical layer, the packet is dropped by the MAC layer.

The *free space* and *two-ray ground reflection* are two well known models.

B.0.1 Free Space Model:

The free space model assumes there is only one clear line of sight between transmitter and receiver, i.e. no interference.

Power at the receiver is given by the Friis free space equation [36]:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^n L} \tag{B.1}$$

for received power P_r, transmitted power P_t, antenna gain G, wavelength λ, separation distance d, path loss exponent of $n = 2$, and system loss $L \geq 1$.

The path loss exponent of 2 is based on the equation for the surface area of a sphere: $A = 4\pi r^2$. In simulations it is common to let $G_t = G_r = 1$ and $L = 1$ [115], and assume that required $P_t \propto d^2$.

The free space model represents an ideal case where a transmitter is at the center of a circle, its radio range is the radius, and the RSS is equal at all points on the circle. If a receiver is on or in the circle, it receives all packets, else it loses all packets.

However, in WSNs radio communication is **irregular** and difficult to model. Reflection, diffraction, and scattering impact propagation. The value of n depends heavily on the radio propagation environment and typically ranges from 2 to 5. P_r may vary greatly with small changes in location.

B.0.2 Two-Ray Ground Reflection Model

A single line-of-sight path between two mobile nodes is seldom the only means of propagation. The two-ray ground reflection model considers both the direct path and a ground reflection path. It is shown [123] that this model gives more accurate prediction at a long distance than the free space model. The received power at distance d is predicted by

$$P_r(d) = \frac{P_t G_t G_r {h_t}^2 {h_r}^2}{d^4 L} \tag{B.2}$$

where h_t and h_r are the heights of the transmit and receive antennas respectively. Note that the original equation in [123] assumes $L = 1$. To be consistent with the free space model, L is added here. The above equation shows a faster power loss than Eqn. B.1 as distance increases. However, the two-ray model does not give a good result for a short distance due to the oscillation caused by the constructive and destructive combination of the two rays. Instead, the free space model is still used when d is small. Therefore, a cross-over distance d_c is calculated in this model. When $d < d_c$, Equation B.1 is used. When $d > d_c$, Equation B.2 is used. At the cross-over distance, Equations B.1 and B.2 give the same result. So d_c can be calculated as

$$d_c = \frac{(4\pi h_t h_r)}{\lambda} \tag{B.3}$$

B.0.3 Limitations

The free space model and the two-ray model predict the mean received power as a deterministic function of distance given an assumed path loss exponent. However, "the surrounding environmental clutter may be vastly different at two different locations having the same T-R separation" [123]. Empirical data has shown that the path loss at a particular location is random and log-normally distributed (normal in dB) about the mean distance-dependent value. Communication is more probabilistic when nodes are near the limits their expected communication range.

Appendix C

Signal to Noise Ratio

The signal to noise ratio (SNR) is the power ratio between a signal (meaningful information) received from a target and the background noise.

$$SNR = \frac{P_{signal}}{P_{noise}}$$

SNRs are usually expressed in terms of the logarithmic decibel scale. In decibels, the SNR may be expressed as 10 times the logarithm of the power ratio.

$$SNR = 10 \log \frac{P_{signal}}{P_{noise}} \; dB$$

The strength of the signal depends on distance and low cost wireless sensors make it possible to get very close to the target. This is critical because each sensor has a limited range, and we need to increase the probability of detecting and accurately measuring a signal at its source, i.e. we want to improve the signal to noise ratio. For example, suppose we want to sense an acoustic signal in a 2D plane. If we assume inverse distance squared attenuation, power received at distance r is

$$P_{receive} \propto \frac{P_{source}}{r^2}.$$

The SNR is

$$SNR_r = 10 \log \frac{P_{receive}}{P_{noise}} = 10 \log P_{source} - 10 \log P_{noise} - 20 \log r.$$

By deploying large numbers of sensors we increase network density and increase the probability that a sensor is close to a target. Increasing density by a factor of x reduces average distance to a target by a factor of $\frac{1}{\sqrt{x}}$. The SNR improvement is

$$\eta_{snr} = SNR_{\frac{r}{\sqrt{x}}} - SNR_r = 20 \log \frac{r}{\frac{r}{\sqrt{x}}} = 10 \log x.$$

Increasing sensor density x times should improve SNR by $10 \log x \; dB$ [32].

Appendix D

Calculating the Optimal Number of Hash Functions

Suppose a bit vector of length m, a data set S of size n, and we want to optimize for the number of hash functions. After all n elements of S are hashed into the filter k times, the probability p that a specific bit is still 0 is

$$\left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$$

The probability that a bit is 1 ("saturation") s would be

$$s = (1 - e^{-\frac{kn}{m}})$$

The probability f of a false positive is

$$f = (1 - e^{-\frac{kn}{m}})^k$$

First use $\ln$ to make f easier to work with. Then the optimal number of hash functions that minimizes f as a function of k may be found by taking the derivative.

$$f = exp(k \ln(1 - e^{-\frac{kn}{m}}))$$

Let

$$g = k \ln(1 - e^{-\frac{kn}{m}})$$

Minimize f by minimizing g with respect to k

$$\frac{dg}{dk} = \ln(1 - e^{-\frac{kn}{m}}) + \frac{kn}{m} \frac{e^{-\frac{kn}{m}}}{1 - e^{-\frac{kn}{n}}}$$

The global minimum is where $\frac{dg}{dk} = 0$ and $k = \frac{m}{n} \ln 2$.
The false positive rate is minimized when $k = \frac{m}{n} \ln 2$.

Using $p = e^{-\frac{kn}{m}}$ we find

$$g = -\frac{m}{n} \ln(p) \ln(1 - p)$$

from which symmetry reveals that the minimum value for g occurs when $p = \frac{1}{2}$, or equivalently $k = \frac{m}{n} \ln 2$. The false positive rate is

$$f = \left(\frac{1}{2}\right)^k = (0.6185)^{\frac{m}{n}}.$$

Given m bits and n elements, the optimal value for k in terms of false positives occurs when hashing n elements k times causes $\frac{1}{2}$ of the bits to be set to 1. At such a k, $k + 1$ hash functions will cause the false positive rate to increase. Therefore, the optimum performance of this algorithm occurs when saturation p of the bit array is 50% [15].

Appendix E

The MD5 Algorithm and Hashing

One question that naturally arises is how exactly should one go about hashing keys to a Bloom filter? Fan et al. [33] needed to hash each key to the filter 4 times. They chose MD5, which is a cryptographic message digest algorithm that hashes an arbitrary length string (URL, in this case) to 128 bits - its MD5 signature. Those bits are partitioned into four 32 bit words, then the modulus of each 32 bit value (by filter size m) was used as an index to the filter and that bit was set to one.

The 128-bit MD5 hashes are typically represented as 32-digit hexadecimal numbers. The following example, from [157], demonstrates a 43-byte ASCII input and the corresponding MD5 hash,

> MD5("The quick brown fox jumps over the lazy dog") =
> 9e107d9d372bb6826bd81d3542a419d6

Even a small change in the message will almost certainly result in a completely different hash, e.g., changing d to c.

> MD5("The quick brown fox jumps over the lazy cog") =
> 1055d3e698d289f2af8663725127bd4b

MD5 was chosen because of its well known properties and relatively fast implementation. It is also more secure in that one cannot easily build a URL that hashes to a particular location, i.e., MD5 is not efficiently invertible. Even so, where security is important, cryptographers have recommended switching to a replacement, such as WHIRLPOOL, SHA-1 or RIPEMD-160 [157].

In the context of web proxies and Bloom filters, the security issue is not obvious. How might an invertible hash function be exploited? Suppose a valid URL and a publicly available hash function. The URL should hash to a set of on bits in the filter. It would test positive against the filter, and a complex task such as disk access would be triggered. A cracker could use the URL's hash codes to generate one or more fake URLs that hash to the same locations in the filter. By using the fake URLs to generate false positives, system performance would degrade - possibly to the extent that valid queries would be denied.

Faster hash functions are available. They can be based on polynomial arithmetic such as Rabin's fingerprinting method [122, 14]. Fingerprints are short tags for larger objects. If two fingerprints are different, then their objects are almost certainly different. In hashing we are interested in bounding the number of collisions, but in fingerprinting we want to avoid collisions altogether [14].

Another approach is to use a very simple hash function to generate a 32 bit value, then perform a linear transformation to generate an integer [41, p. 48]. Unfortunately they are efficiently invertible and, therefore, vulnerable to attack.

The following hash function written in C is derived from [41]. It takes a character string as an argument and returns an integer in the range from 0 to $m-1$. It uses 131 as its base - this will generate extremely large integers and overflow is likely, so if $i < 0$ then we add m to i to put i within the desired range.

```
int hashfunction(char *s)
{
  int i;
  for( i=0; *s; s++ )
     i = 131*i + *s;
  i = i % m;
  if (i < 0)
     i += m;

  return i;
}
```

Java's hash function is a member of the java.lang.String class and uses 31 for its base [70].

$$s[0] * 31^{n-1} + s[1] * 31^{n-2} + ... + s[n-1]$$

Using integer arithmetic, where $s[i]$ is the i-th character of the string, n is the length of the string, and the hash value of the empty string is zero.

The following hash function is written in Java 1.4 and was derived from Weiss [155]. It takes a Java String object and returns and an integer in the range from 0 to $m-1$. Again, we guard against overflow.

```
public int hashCode(String s)
{
  int code = 0;
  int n = s.length();

  for (int i = 0; i < n; ++i)
     code = 31 * code + s.charAt(i);
  code = code % m;
  if (code < 0)
     code += m;

  return code;
}
```

Usually we need to generate more than one hash value for each key. One very simple way to create multiple independent hash functions is to use one of the above

hash functions to generate a seed value for a random number generator. The we can use the next k random integers, modulo filter size m, to get k indices for the filter.

Java provides security with message digest algorithms implemented in its java.security package. Their API requires and uses standard names for algorithms, certificate and keystore types.

> The algorithm names in this section can be specified when generating an instance of MessageDigest.
>
> MD2: The MD2 message digest algorithm as defined in RFC 1319.
>
> MD5: The MD5 message digest algorithm as defined in RFC 1321.
>
> SHA-1: The Secure Hash Algorithm, as defined in Secure Hash Standard, NIST FIPS 180-1.
>
> SHA-256, SHA-384, and SHA-512: New hash algorithms for which the draft Federal Information Processing Standard 180-2, Secure Hash Standard (SHS) is now available. SHA-256 is a 256-bit hash function intended to provide 128 bits of security against collision attacks, while SHA-512 is a 512-bit hash function intended to provide 256 bits of security. A 384-bit hash may be obtained by truncating the SHA-512 output. [71]

Bibliography

[1] K. Akkaya and M. Younis, A Survey on Routing Protocols for Wireless Sensor Networks, *Journal of Ad Hoc Networks*, Vol. 3, No. 3, pp. 325-349, May 2005.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine*, August 2002.

[3] J.N. Al-Karaki, A.E. Kamal, Routing Techniques in Wireless Sensor Networks: A Survey, *IEEE Wireless Communications*, December 2004.

[4] J.N. Al-Karaki, A.E. Kamal, A Taxonomy of Routing Techniques in Wireless Sensor Networks, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, Edited by Mohammad Ilyas and Imad Mahgoub, CRC Press, 2005.

[5] G. Anastasi, E. Borgia, M. Conti, E. Gregori, and A. Passarella, Understanding the real behavior of Mote and 802.11 ad hoc networks: an experimental approach, *Pervasive and Mobile Computing*(PerCom), Vol. 1, No. 2., pp. 237-256, July 2005.

[6] A. Arora, et al., ExScal: Elements of an Extreme Scale Wireless Sensor Network, *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* (RTCSA), 2005.

[7] Atmel, Atmel ATmega128L Product Summary, November, 2004.

[8] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing, *U.C. Berkeley Technical Report UCB/CSD-01-1141*, April 2001.

[9] Mark Baard, Making Wines Finer With Wireless, *www.wired.com/news*, April 4, 2003.

[10] A-L. Barabási and R. Albert, Emergence of scaling in random networks, *Science*, vol. 286, pp. 509-512, Oct. 1999.

[11] B. Bloom. Space/Time Tradeoffs in Hash Coding with Allowable Errors, *CACM*, 13(7):422-426, July 1970.

[12] P. Bonnet, J. Gehrke and P. Seshadri. Querying the Physical World, *IEEE Personal Communication*, Vol. 7, No. 5, pp.10-15, 2000.

[13] D. Braginsky and D. Estrin, Rumor Routing Algorithm For Sensor Networks, *Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, 2002.

[14] A.Z. Broder, Some applications of Rabin's fingerprinting method, In *Sequences II: Methods in Communications, Security, and Computer Science*, R. Capocelli, A. De Santis, and U. Vaccaro, Eds. New York, NY: Springer Verlag, pp. 143-152, 1993.

[15] A. Broder and M. Mitzenmacher, Network Applications of Bloom Filters: A Survey, *Internet Mathematics*, 1(4):485-509, 2004.

[16] Nirupama Bulusu, Sanjay Jha. *Wireless Sensor Networks: A Systems Perspective*, Artech House, Inc., 2005.

[17] John W. Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost, Informed content delivery across adaptive overlay networks, *IEEE/ACM Transactions on Networking (TON)*, 12(5): 767-780, 2004.

[18] Cerpa, A. et al., Habitat monitoring: application driver for wireless communications technology, In *Proc. ACM SIGCOMM Workshop Data Communications*, Latin America, Caribbean, 2001.

[19] S. Chang, A. Kirsch, and M. Lyons, Energy and Storage Reduction in Data Intensive Wireless Sensor Network Applications, *Technical Report TR-15-07*, Harvard University, 2007.

[20] B. Chazelle, J. Kilian, R. Rubinfeld, A. Tal, The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Tables, Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 30-39, January 2004.

[21] C.F. Chiasserini, I. Chlamtac, P. Monti and A. Nucci, Energy Efficient Design of Wireless Ad Hoc Networks, In *Proceedings of European Wireless*, February 2002.

[22] S. Cohen, Y. Matias, Spectral Bloom Filters, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, California, pp. 241 - 252, 2003.

[23] S. Bandyopadhyay, E. Coyle, Minimizing communication costs in hierarchically-clustered networks of wireless sensors, *Computer Networks*, 44(1): 1-16 (2004).

[24] S. Dixit, E. Yanmax, and O. Tonguz, On the Design of Self-Organized Cellular Wireless Neworks, *IEEE Communications Magazine*, July 2005.

[25] L. Doherty, B. A. Warneke, B. E. Boser, and K. Pister. Energy and performance considerations for smart dust, *International Journal of Parallel Distributed Systems and Networks*, 4(3):121-133, 2001.

[26] G.L. Duckworth, D.C. Gilbert, and J.E. Barger. Acoustic counter-sniper system, In *SPIE International Symposium on Enabling Technologies for Law Enforcement and Security*, 1996.

[27] R.O. Duda, P.E. Hart, & D.G. Stork, *Pattern Classification*, Second Edition., New York, NY: John Wiley and Sons, 2001.

[28] P. Erdös and A. Rényi, On the evolution of random graphs, *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17-60, 1959.

[29] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks, *IEEE Pervasive Computing*, pages 59-69, January 2002.

[30] D. Estrin, R. Govindan, J. Heidemann and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks, *In Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99)*, Seattle, Washington, August 1999.

[31] F. Zhao, J. Liu, J. J. Liu, L. Guibas, and J. Reich. Collaborative signal processing: an information directed approach, *Proc. IEEE*, 91(8):1199-1209, 2003.

[32] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: an information processing approach*, Elsevier Inc., 2004

[33] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary Cache: a scalable wide-area Web cache sharing protocol, *IEEE/AACM Transactions on Networking*, 8(3):281-293, 2000.

[34] Feng, J. et al., Sensor networks: quantitative approach to architecture and synthesis, *UCLA, Technical Report*, 2002.

[35] J. Feng, F. Koushanfar, and M. Potkonjak. System-Architectures for Sensor Networks: Issues, Alternatives, and Directions, *ICCD, special session on Sensor Networks*, Freiburg, Germany, page 112 121, Sep 2002.

[36] H. T. Friis. A note on a simple transmission formula. *Proc. IRE*, p34, 1946.

[37] M. T. Gastner and M. E. J. Newman, The spatial structure of networks, *The European Physics Journal B*, 49, 247-252, 2006.

[38] J. Gehrke, S. Madden, Query Processing in Sensor Networks, *Pervasive Computing*, January - March, 2004.

[39] A. Ghose, J. Grossklags and J. Chuang, Resilient Data-Centric Storage in Wireless Ad-Hoc Sensor Networks, *Mobile Data Management*, M.-S.Chen, P.K.Chrysanthis, M.Sloman, A.Zaslavsky (Eds.) Springer LNCS series, No. 2574, pp. 45-62, 2003.

[40] O. Gnawali, B. Greenstein, K-Y Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, E. Kohler, The TENET Architecture for Tiered Sensor Networks, In Proceedings of the *ACM Conference on Embedded Networked Sensor Systems* (Sensys), November 2006.

[41] G. Gonnet and R. Baeza-Yates, *Handbook of Algorithms and Data Structures*. Reading, MA: Addison-Wesley, 1991.

[42] D.L. Guidoni, R.A.F. Mini, and A.A.F. Loureiro, On the Design of heterogeneous Sensor Networks Based on Small World Concepts, *The 11-th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Vancouver, BDC, Canada, October 27-31, 2008.

[43] P. Gupta and P. R. Kumar. The capacity of wireless networks, *IEEE Trans. on Information Theory*, 46(2):388-404, 2000.

[44] J.A. Gutierrez, E.H. Callaway, R. Barrett, *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4*, IEEE Press, New York, 2003.

[45] Martin Haenggi and Daniele Puccinelli, Routing in Ad Hoc Networks: A case for Long Hops, *IEEE Communications Magazine*, Series on Ad Hoc and Sensor Networks, vol. 43, pp. 93-101, Oct. 2005.

[46] F. Hao, M. Kodialam, and T.V. Lakshman, Building High Accuracy Bloom Filters using Patitioned Hashing, *Joint International Conference on Measurement and Modeling of Computer Systems* (SIGMETRICS), pp:277 - 288, San Diego, California, USA, 2007.

[47] P. Hebden and A.R. Pearce. Data-Centric Routing using Bloom Filters in Wireless Sensor Networks, *Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP)*, Bangalore, India, December 2006.

[48] P. Hebden and A.R. Pearce. Bloom filters for data aggregation and discovery: a hierarchical clustering approach, *Second International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, December 2005.

[49] P. Hebden and A.R. Pearce. Distributed Asynchronous Clustering for Self-Organisation of Wireless Sensor Networks, *Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP)*, Bangalore, India, December 2006.

[50] J. Heidemann, F. Silva and D. Estrin. Matching Data Dissemination Algorithms to Application Requirements, In *Proceedings of the ACM SenSys Conference*, Los Angeles, CA, 2003.

[51] W. Heinzelman, J. Kulik and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks, *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom '99), Seattle, Washington, pp. 174-185, August 15-20, 1999.

[52] W. Heinzelman. Application-Specific Protocol Architectures for Wireless Networks, Ph.D. Dissertation, Massachusetts Institute of Technology, June 2000.

[53] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks, *IEEE Trans. Wireless Communication*, 1(4):660-670, Oct. 2002.

[54] W. Heinzelman, A. Murphy and M. Perillo. Enabling Data- and Event-centric Communications, *Wireless Sensor Networks: A Systems Perspective*, Editors: Nirupama Bulusu, Sanjay Jha, Artech House, Inc., 2005.

[55] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks, *IEEE HICSS*, January 2000.

[56] A. Helmy. Architectural Framework for Large-Scale Multicast in Mobile Ad Hoc Networks, IEEE International Conference on Communications (ICC), Vol. 4, pp. 2036-2042, New York, April 2002.

[57] A. Helmy, S. Garg, P. Pamu, N. Nahata. Contact Based Architecture for Resource Discovery (CARD) in Large Scale MANets, IEEE/ACM IPDPS Int'l Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN), pp. 219-227, Apr 2003.

[58] A. Helmy, S. Garg, P. Pamu, N. Nahata. CARD: A Contact-based Architecture for Resource Discovery in Ad Hoc Networks, ACM Baltzer Mobile Networks and Applications (MONET) Journal, Kluwer publications, Special issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks, Volume 10, Issue 1, pp. 99-113, Feb 2005.

[59] A. Helmy. Small Worlds in Wireless Networks, *IEEE Communications Letters*, 7(10), Oct 2003.

[60] Jason Hill, JLH Labs: Hardware Profiles. http://www.jlhlabs.com/

[61] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors, In *Proceedings of ACM ASPLOS IX*, pages 93-104, November 2000.

[62] Jason Hill. System Architecture for Wireless Sensor Networks, PhD Thesis, 2003.

[63] C. Intanagonwiwat, R. Govindan, D. Estrin. Directed Diffusion: A Scalable and robust Communication Paradigm for Sensor Networks, *Proc. 6th Annual International Conference on Mobile Computing and Networks (MobiCom)*, Boston, MA, 2000.

[64] C. Intanagonwiwat, R. Govindan and D. Estrin. Directed Diffusion for Wireless Sensor Networking, *IEEE/ACM Transactions on Networking*, 11(1): 2-16, Feb 2003.

[65] Intel Corporation. Heterogeneous Sensor Networks, 2005. http://www.intel.com/research/exploratory/heterogeneous.htm

[66] Intel Corporation. New Computing Frontiers - The Wireless Vineyard, http://www.intel.com/technology/
techresearch/research/rs01031.htm

[67] Ivan Stojmenovic, Simulations in Wireless Sensor and Ad Hoc Networks: Matching Advancing Models, Metrics, and Solutions, *IEEE Communications Magazine*, December 2008.

[68] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks, *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November, 2003.

[69] A.K. Jain, M.N. Murty and P.J. Flynn. Data Clustering: A Review, *ACM Comp. Surv*, 1999.

[70] Java, http://java.sun.com/j2se/1.4.2/docs/api/

[71] JavaTM Cryptography Architecture: API Specification & Reference, 25 July 2004.
http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html

[72] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, pages 243-254, Aug. 2000.

[73] Chris Karlof, Naveen Sastry, David Wagner. TinySec: a link layer security architecture for wireless sensor networks, Conference On Embedded Networked Sensor Systems, Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, pp. 162-175, 2004.

[74] Jon M. Kleinberg, Navigation in a small world, *Nature*, 406:845, 2000.

[75] C. Newport, D. Kotz, R.S. Gray, J. Liu, Y. Yuan and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, April 2007.

[76] B. Krishnamachari, D. Estrin, S. Wicker. Modelling Data-Centric Routing in Wireless Sensor Networks. *IEEE Infocom*, 2002.

[77] Bhaskar Krishnamachari. Networking Wireless Sensors, Cambridge University Press, 2005.

[78] Lakshman Krishnamurthy, et al., Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea, *The 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[79] J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks, *Wireless Networks*, Volume 8 Issue 2/3, March 2002.

[80] A. Kumar, J. Xu, L. Li, and J. Wang, Space-code bloom filter for efficient traffic flow measurement, In the Proceedings of *ACM/USENIX Internet Measurement Conference* (IMC), Miami, October 2003.

[81] K. Langendoen, A. Baggio, and O. Visser, Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture, *14th Int. Workshop on Parallel and Distributed Real-Time Systems* (WPDRTS), Rhodes, Greece, April 2006.

[82] Jonathan Ledlie, Jacob M. Taylor, Laura Serban, and Margo Seltzer. Self-Organization in Peer-to-Peer Systems, In *Proceedings of Tenth ACM SIGOPS European Workshop*, September 2002.

[83] Z. Li and K. A. Ross. Perf join: An alternative to two-way semijoin and Bloomjoin. In *Proceedings of the 1995 International Conference on Information and Knowledge Management (CIKM '95)*, pp 137-144, November 1995.

[84] Jai-Jin Lim and K.G. Shin. Gradient-Ascending Routing via Footprints in Wireless Sensor Networks, *26th IEEE International Real-Time Systems Symposium (RTSS)*, 5-8 December 2005.

[85] Meng-Jang Lin, K. Marzullo, and S. Masini. Gossip Versus Deterministically Constrained Flooding on Small Networks. *14th International Conference on Distributed Computing (DISC)*, Oct. 2000.

[86] S. Lindsey, C. Raghavendra, and K. M. Sivalingam. Data Gathering Algorithms in Sensor Networks Using Energy Metrics, *IEEE Trans. Parallel and Distrib. Sys.*, vol. 13, no. 9, pp. 924 935, Sept. 2002.

[87] Yi Lu, Balaji Prabhakar, and Flavio Bonomi, Perfect Hashing for Network Applications, *IEEE International Symposium on Information Theory* (ISIT), Seattle, USA, July 9-14, 2006.

[88] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, MiniSec: A Secure Sensor Network Communication Architecture, *Information Processing in Sensor Networks (IPSN)*, Cambridge, Massachusetts, USA, 25-27 April 2007.

[89] Samuel. R. Madden, Michael. J. Franklin, Joseph Hellerstein, and Wei Hong, The Design of an Acquisitional Query Processor for Sensor Networks, *SIGMOD*, San Diego, CA, June 2003.

[90] Samuel. R. Madden, Michael. J. Franklin, Joseph Hellerstein, and Wei Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks, OSDI, Boston MA, 2002.

[91] Samuel. R. Madden, Michael. J. Franklin, Joseph Hellerstein, and Wei Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks, In *ACM Transactions on Database Systems*, 2005.

[92] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, and David Culler. Wireless sensor networks for habitat monitoring, In *First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[93] U. Manber and S. Wu. An algorithm for approximate membership checking with application to password security. Information Processing Letters, 50:191-197, 1994.

[94] A. Manjeshwar and D. P. Agrawal, APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks, *2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, April 15-19, 2002.

[95] A. Manjeshwar, and D. P. Agrawal, TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks, *1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, CA, April 23-27, 2001.

[96] J.M. McCune, E. Shi, A. Perrig, and M.K. Reiter, Detection of denial-of-message attacks on sensor network broadcasts *IEEE Symposium on Security and Privacy*, pp:64-78, 8-11 May 2005.

[97] M.D. McIlroy. Development of a spelling list, *IEEE Transactions on Communications*, 30(1):91-99, January 1982.

[98] Xiaoqiao Meng, Thyaga Nandagopal, Li Li, and Songwu Lu. Contour Maps: Monitoring and diagnosis in sensor networks, *Computer Networks*, 50:2820-2838, 2006.

[99] Stanley Milgram, The Small World Problem, *Psychology Today*, 1 (May):60-67, 1967.

[100] R. Min and A. Chandrakasan, Energy-Efficient Communication for High Density Networks, in *Ambient Intelligence: Impact on Embedded System Design*, editors T. Basten, M. Geilen, H. Groot, Kluwer Academic Publishers, 2003.

[101] R. Min and A. Chandrakasan, Top Five Myths about the Energy Consumption of Wireless Communication, *Mobile Computing and Communications Review*, 7(1): 65-67, January 2003.

[102] M. Mitzenmacher. Compressed Bloom Filters, *IEEE/ACM Transactions on Networks*, 10:5, pp. 613-620, October 2002.

[103] A. Moffat, R. Neal, and I.H. Witten. Arithmetic Coding Revisited, *ACM Trans. Inform. Syst.*, vol. 16, no 3, pp. 256-294, July 1998.

[104] Andreas F. Molisch. *Wireless Communications*, Wiley, 2005.

[105] Gordan Moore. Progress in Digital Integrated Electronics, *IEDM Tech. Digest*, pages 11-13, 1975.

[106] O. Moussaoui and M. Naimi. A Distributed Energy Aware Routing Protocol for Wireless Sensor Networks, *The 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, Montreal, Quebec, Oct. 2005

[107] J. K. Mullin. Optimal semijoins for distributed database systems, *IEEE Transactions on Software Engineering*, 16(5):558, May 1990.

[108] John Nagle, On Packet Switches with Infinite Storage, *IEEE Transactions on Communications*, 35(4):435 - 438, Apr 1987

[109] M.E.J. Newman, Mathematics of Networks, *The New Palgrave Encyclopedia of Economics*, 2nd edition, L. E. Blume and S. N. Durlauf (eds.), Palgrave Macmillan, Basingstoke (2008).

[110] M. E. J. Newman, Models of the small world: A review, *Journal of Statistical Physics*, vol. 101, pp. 819-841, 2000.

[111] M. E. J. Newman and D. J. Watts, Renormalization group analysis of the small-world network model, Phys. Lett. A, 263:341-346, 1999.

[112] M. E. J. Newman, The structure and function of complex networks, *SIAM Review*, 45(2): 167-256, 2003.

[113] D. Niculescu, Communication Paradigms for Sensor Networks, *IEEE Communications Magazine*, March 2005

[114] National Research Council. *Embedded, Everywhere: A Research Agendafor Networkded Systems of Embedded Computers*. National Academy Press, 2001.

[115] The *ns* Manual, Edited by Kevin Fall and Kannan Varadhan, 27 April 2007.

[116] A. Pagh, R. Pagh, and S.S. Rao, An Optimal Bloom Filter Replacement, *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (SODA), Vancouver, B.C., Canada, 2005.

[117] Joseph Polastre, Jason Hill, David Culler. Versatile Low Power Media Access for Wireless Sensor Networks, *Proc. Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, 2004.

[118] G. J. Pottie and W. J. Kaiser. Wireless Integrated Network Sensors, *Communications of the ACM*, vol. 43(5), May 2000, pp. 51-58.

[119] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network, In *Proceedings of the ACM SIGCOMM 2001 Conference*, 31(4):161-172, Aug. 2001.

[120] S. Ratnasamy, B. Karp, Y. Li, F. Yu, R. Govindan, S. Shenker and D. Estrin. GHT: A Geographic Hash Table for Data-Centric Storage, In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications* (WSNA 2002), Oct. 2002.

[121] J. M. Rabaey, J. Ammer, T. Karalar, S.Li, B. Otis, M Sheets, T. Tuan, PicoRadios for wireless sensor networksthe next challenge in ultra-low power design, ISSCC2002.

[122] M.O. Rabin, Fingerprinting by random polynomials, Center for Research in Computing Technology, *Harvard University, Tech. Rep. TR-15-81*, 1981.

[123] T. Rappaport, *Wireless Communications: Principles & Practice*, 2nd Edition, Prentice-Hall, 2002.

[124] Patrick Reynolds, Amin Vahdat. Efficient Peer-to-Peer Keyword Searching, Lecture Notes in Computer Science, Springer-Verlag Heidelberg, Vol. 2672/2003, pp. 21 - 40

[125] RFM, RF Monolithics TR1000 Radio, http://www.rfm.com/products/data/tr1000.pdf

[126] S.C. Rhea and J. Kubiatowicz. Probabilistic Location and Routing, *Proceedings of INFOCOM 2002*

[127] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-peer systems, Lecture Notes in Computer Science, 2218:329-340, 2001.

[128] K. Romer and F. Mattern. The Design of Wireless Sensor Networks, *IEEE Wireless Communications*, December 2004

[129] A. Rousskov and D. Wessels. Cache digests. Computer Networks and ISDN Systems, 30(22-23): 2155-2168, 1998.

[130] K. Sayood, Data Compression. Second Edition, Morgan Kaufmann Publishers, 2000.

[131] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks, *8th Annual International Conference on Mobile Computing and Networks (MobiCom)*, Atlanta, GA, Sept. 2002.

[132] C. Schurgers and M. B. Srivastava, Energy efficient routing in wireless sensor networks, *Military Communications Conference*, 2001.

[133] P. Sen, S. Dasgupta, A. Chatterjee, P. A. Sreeram, G. Mukherjee, and S. S. Manna, Small-world properties of the Indian railway network, *Phys. Rev. E*, vol 67, 036106, Issue 3, March 2003.

[134] R. Shah, J. Rabaey. Energy Aware Routing for Low Energy Ad Hoc Sensor-Networks, *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Orlando, FL, 2002.

[135] G. Sharma and R. Mazumdar, Hybrid Sensor Network: a Small World, *ACM International Symposium on Mobile Ad Hoc Networking and Computing* (MobiHoc), May 25-27, 2005.

[136] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-Centric Storage in Sensornets, In *Proceedings of the First ACM SIGCOMM Workshop on Hot Topics in Networks*, Oct. 2002.

[137] F. Silva, J. Heidemann, R. Govindan, and D. Estrin. Directed Diffusion, *Technical Report ISI-TR-2004-586, USC/Information Sciences Institute*, January, 2004.

[138] E. H. Spafford, Opus: Preventing weak password choices, Computer and Security, 11:273-278, May 1992.

[139] F. Stann and J. Heidemann, BARD: Bayesian-Assisted Resource Discovery In Sensor Networks, *In Proceedings of the IEEE Infocom*, pp. 866-877. Miami, Florida, USA, March, 2005.

[140] I. Stoica , R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications, In *Proceedings of the ACM SIGCOMM 01 Conference*, San Diego, California, August 2001.

[141] S. H. Strogatz, Exploring complex networks, *Nature*, vol. 410, pp. 268-276, March 2001.

[142] L. Subramanian and R.H. Katz, An Architecture for Building Self-Configurable Systems, *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, August 2000.

[143] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition, In *First European Workshop on Wireless Sensor Networks* (EWSN '04), January 2004.

[144] Andrew S. Tanenbaum, Computer Networks, Third Edition, 1996.

[145] S. Tilak, N. Abu-Ghazaleh, W. Heinzelman. A Taxonomy of Wireless Microsensor Network Models, *ACM SIGMOBILE Mobile Comp. Commun. Rev.*, vol. 6, no. 2, Apr. 2002, pp. 2836.

[146] J. Travers and S. Milgram, An Experimental Study of the Small World Problem *Sociometry*, Vol. 32, No. 4. (Dec., 1969), pp. 425-443.

[147] Smart buildings admit their faults. Lab Notes: Research from the College of Engineering, UC Berkeley, http://www.coe.berkeley.edu/labnotes/ 1101smartbuildings.html, November 2001.

[148] A. Vahdat and D. Becker, Epidemic Routing for Partially Connected Ad Hoc Networks, *Technical Report CS-200006*, Duke University, April 2000.

[149] Hal R. Varian. How to Build an Economic Model in your Spare Time, *Passion and Craft: Economists at Work*, Ed. M. Szenberg, University of Michigan Press, 1997.

[150] M.A.M. Vieira, C.N. Coelho, Jr., D.C. da Silva, Jr., J.M. da Mata, Survey on wireless sensor network devices, *IEEE Conf. on Emerging Technologies and Factory Automation* (ETFA), 1:537-544, 16-19 Sept. 2003

[151] Q. Wang, M. Hempstead, and W. Yang. A Realistic Power Consumption Model for Wireless Sensor Network Devices. In Proceedings of the Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks(SECON). Reston, VA, September 2006.

[152] D. J. Watts and S. H. Strogatz, Collective dynamics of small world networks, *Nature*, vol. 393, pp. 440-442, June 1998.

[153] Y. Wei, J. Heidemann, D. Estrin. An Energy Efficient MAC Protocol for Wireless Sensor Networks, *Proc. 21st International Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, New York, NY, 2002

[154] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94-100, September 1991.

[155] Mark Allen Weiss, *Data Structures & Problem Solving Using Java*, Addison Wesley Longman, 1998.

[156] Matt Welsh, Dan Myung, Mark Gaynor, and Steve Moulton. Resuscitation monitoring with a wireless sensor network. Supplement to Circulation: Journal of the American Heart Association, October 2003.

[157] wikipedia, MD5, http://en.wikipedia.org/wiki/MD5

[158] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks, *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[159] Crossbow Technology, Inc., MICA2DOT Datasheet, Document Part Number: 6020-0043-05 Rev A, *www.xbow.com*.

[160] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient Broadcast: A robust data delivery protocol for large scale sensor networks, *ACM Wireless Networks (WINET)*, vol. 11, no. 2, March 2005.

[161] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor neworks. In *Proc. 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2002)*, volume 3, pages 3-12, June 2002.

[162] Fan Ye, Haiyun Luo, Songwu Lu and Lixia Zhang. Statistical En-route Detection and Filtering of Injected False Data in Sensor Networks, *IEEE Journal on Selected Areas in Communications*, Volume: 23 , Issue: 4, Pages: 839 - 850, Publication Date: April 2005.

[163] O. Younis and S. Fahmy. HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks, *IEEE Transactions on Mobile Computing*, Vol. 3, No. 4, Oct - Dec 2004.

[164] Y. Yu, R. Govindan, D. Estrin. Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks, Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, May 2001.

[165] L.C. Zhong, J.M. Rabaey, and A. Wolisz, Does proper coding make single-hop wireless sensor networks reality: the power consumption perspective, WCNC2005