A FAST DATA AGGREGATION TECHNIQUE FOR WIRELESS SENSOR NETWORKS

by

Yan Long

B.Sc., Simon Fraser University, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2006

APPROVAL

Name:	Yan Long
Degree:	Master of Science
Title of thesis:	A Fast Data Aggregation Technique for Wireless Sensor Networks

Examining Committee: Dr. Greg Mori
Chair

____

Dr. A. Funda Ergun, Senior Supervisor

____

Dr. Arthur L. Liestman, Supervisor

____

Dr. Ramesh Krishnamurti, SFU Examiner

Date Approved: July 18, 2006

Abstract

A wireless sensor network is composed of a large number of sensor nodes embedded in the physical world to perform monitoring and surveillance tasks. The primary operation on sensor networks is extracting aggregated information from the networks, which can be time-consuming due to the environmental dynamics and the large number of sensor nodes involved. In this thesis, we present a fast data aggregation technique for wireless sensor networks which uses a randomized architecture. The architecture is composed of layers which are constructed in a distributed, localized fashion. The key property of our technique is that data are collected from one layer containing a subset of sensor nodes, resulting in fewer hops and thus lower delay in data aggregation. We provide theoretical guarantees for the delay incurred and the accuracy level of the results. In addition, we explore ways to further speed up the data aggregation process by using history information.

To Lei

Acknowledgments

I would like to thank Dr. Funda Ergun for offering her valuable time and support as my senior supervisor. I feel very fortunate to have been mentored by someone with such keen insight. Were it not for her advice and vision, this work would not have been possible.

I would like to thank my supervisor, Dr. Arthur Liestman, and my thesis examiner, Dr. Ramesh Krishnamurti, for taking the effort to read this thesis carefully and make thoughtful suggestions.

I would like to thank my fellow graduate students in Network Modeling Group for enabling me to spend such a lovely time during my graduate study at SFU.

Mostly I am indebted to my collaborator and friend Dan Wang, whose insightful ideas and dedicated spirit have been essential to the success of this work.

I would like to thank my parents for their unconditional love and support.

Finally, thank you Lei for sharing with me the happy moments and the more difficult times, and for always being on my side encouraging and believing in me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Recent advances in hardware technology and engineering design have allowed the integration of sensing, data processing, and wireless communication capabilities into a small, inexpensive, self-contained, battery-powered device called a *sensor node.* Depending on their sensing components, sensor nodes can be used to monitor temperature, light, humidity, pressure, sound, radiation, or vibration.

In most settings, sensor nodes are spatially distributed throughout a region of interest, self-organize into a network through wireless communication, and collaborate with each other to accomplish a common task. Such a network is called a *wireless sensor network.* As sensor nodes become smaller, cheaper and more powerful, the deployment of large scale wireless sensor networks is changing from a technological vision to reality.

The strength of wireless sensor networks lies in their flexibility and scalability. The characteristic of being self-contained, and the capacity of wireless communication allow sensor nodes to be deployed in an ad-hoc fashion, in remote and/or hazardous locations without any existing infrastructure. This will enable scientists to observe the physical world in much more detail and reveal many previously unobservable phenomena. In experimental sensor networks, sensor nodes are being deployed on the ground, under the water [41], on bridges or buildings [45], where the deployment of traditional wired or wireless networks are not possible or too costly. Wireless sensor nodes usually only communicate directly with other nodes in their vicinity and reach far-away nodes through *multi-hop* communication. This allows new sensor nodes to be easily added to a wireless sensor network to expand the area covered by the network. In theory, the coverage range of a wireless sensor network can be extended indefinitely.

The popular vision for wireless sensor networks is to create "smart environments" through ad-hoc deployment of a large number of sensor nodes, which will not only be able to acquire information, but also be able to act on the information to improve the environments. Such a vision opens up numerous opportunities. For example, tiny wireless sensor nodes can be scattered over an uninhabited island to observe how sea birds respond to environment changes without human disturbance [31]. In disaster response, sensor networking allows real-time, wireless monitoring and tracking of multiple patients and emergency response teams, and directs resources to patients who require immediate attention [27]. Smart kindergartens will allow parents and teachers to investigate children's learning processes, and develop individualized education processes for each child [40]. These are just a few motivating usage scenarios. We will look into sensor network applications in more detail later.

1.1.1 Sensor Network Challenges

While wireless sensor networks promise a wide range of applications, they also present a new collection of challenges. Realization of the vision of wireless sensor networks demands new protocols, algorithms and network architectures that address these challenges:

- **Physical resource constraints:** The most severe constraint imposed on sensor networks is the limited power supply of sensor nodes. The effective lifetime of a sensor node will be determined by its power supply. Sensor nodes are battery-powered, and batteries of sensor nodes are not rechargeable in most cases. Hence energy conservation is one of the main system design issues. Limited computation power and memory size of sensor nodes also restrict the types of processing algorithms that can be used and the amount of information that can be stored in individual sensor nodes. Communication delay in sensor networks can be rather high due to the limited communication capacity shared by sensor nodes within each other's transmission range, and multi-hop networking necessitated by short transmission ranges of sensor nodes.

- **Ad hoc deployment:** In many applications, sensor nodes are deployed in regions without any infrastructure. For example, for forest fire detection, sensor nodes typically would be dropped into a forest from a plane. For deep-sea phenomenon observation, sensor nodes would be thrown from submarines. It is often difficult to control the exact distribution of sensor nodes. In such situations, it is up to the sensor nodes to communicate with their neighbors to identify their connectivity and distribution.

- **Unattended operation in dynamic environments:** Given the large number of sensor nodes and their generally inaccessible deployment locations, in most cases, sensor networks have to operate for long periods of time without human intervention. Sensor nodes may expire due to exhausted batteries or malfunction due to accidents. Communication links may break down frequently due to environmental interference and noise. Thus, such networks have to be able to adapt to any changes in the connectivity and environment stimuli. The failure of individual sensor nodes should not affect the functionalities of the whole network.

1.1.2 Sensor Networks vs. Traditional Wireless Networks

While many existing techniques and concepts from traditional wireless networks, such as cellular networks, wireless local area networks and Bluetooth, are applicable to sensor networks, there are also a number of fundamental differences. Some of the most important defining characteristics of sensor networks are outlined as follows:

- Sensor networks have much larger number of nodes than traditional wireless networks. Potentially, the number of sensor nodes of a sensor network has to scale to thousands, even hundreds of thousands. Moreover a sensor network might need to be extended from time to time, which makes it difficult to predict the number of nodes at its initial deployment. This calls for highly scalable solutions to ensure a sensor network is operational regardless of the number of nodes.

- *Data centricity* is the feature of sensor networks which distinguishes them from other wireless networks. Individual nodes in sensor networks possess limited processing speed, storage capacity, and communication bandwidth. Thus applications require collaborative effort from the sensor nodes. As a result, operations in sensor networks are centered around data instead of individual sensor nodes. This calls for data-centric network architectures and communication protocols.

- Sensor networks are environment-driven. While traffic in most traditional wireless networks is generated by humans, most data of sensor networks are generated automatically in response to changes in the environment. As a result, sensor networks are expected to exhibit entirely new traffic patterns, which can vary dramatically from time to time. This calls for adaptive topology schemes to extend the useful lifetime of networks.

- Sensor networks require a new set of performance metrics. Traditional wireless networks are general traffic transport networks; they aim to deliver payload bits from sources to destinations efficiently. By contrast, sensor networks are data collection networks. Here the focus is not on original bits; rather, it is on the amount and quality of the information that can be obtained about the objects and areas being monitored. In many application scenarios, the large number of sensor nodes involved, coupled with the uncertainty of sensor readings rules out exact and deterministic approaches; approximate and randomized computations have to be used. These features call for more complicated metrics such as probability of correct detection and approximation accuracy.

- Another issue unique to sensor networks is the correlated data problem. Data collected by neighboring sensor nodes are often quite similar, which makes possible the development of techniques that combine routing structure and in-network data processing to reduce data redundancy and improve energy efficiency. Meanwhile, many environment quantities change only slowly over time. Thus, consecutive sensor readings are temporally correlated to a certain extent. This advantageous feature can be exploited to develop efficient data collection techniques.

1.1.3 Sensor Network Applications

Application areas for sensor networks include military, environment, civil, health, and industry. Today, more and more pilot sensor network applications are being deployed in academia and industry.

In the following paragraphs, we discuss some of the emerging applications that have attracted many research efforts. We also discuss how these applications can vary significantly in latency requirements.

- **Military Applications:**

  Wireless sensor networks can be rapidly deployed in an ad hoc fashion and self-organize to operate in highly dynamic environments, making them ideal candidates for many military applications. For instance, sensor networks can be deployed in open terrains for detecting intruders. Another potential application is battlefield situation awareness: soldiers, vehicles and weapons equipped with wireless sensor nodes can be monitored in real time; sensor networking techniques will lead to better collaboration among soldiers in the battlefield. In military applications, it is essential that the networks are robust and fault-tolerant. Most importantly, military applications usually have strict latency requirements.

- **Security and Surveillance:**

  Critical infrastructures such as airports, subways, water supply systems and energy systems can be subjected to terrorist attacks. Networked sensor nodes can be deployed throughout these facilities to provide ubiquitous protection. For example, sensor nodes can be used as air quality monitors in airports and subways, and as water monitors in water supply systems to provide early detection of chemical and biological substances in the air and water. In security applications, the tasks of sensor nodes are collecting and immediately reporting anomalies. Here low latency is the most important consideration. Stringent delay requirements are usually met at the cost of higher energy consumption.

- **Environmental Monitoring:**

  Many scientific researches in biology and life science require taking observations in unattended environments over a long period of time. Sensor nodes can be unobtrusively embedded in uninhabited areas to perform such monitoring tasks where traditional human-centric methods are difficult, impractical or too disturbing. For instance, researchers have deployed a sensor network on Great Duck Island [31] to study how micro-climatic factors influence the habitat selection of the sea birds. A prototype sensor network for meteorological and hydrological observations has been deployed in Yosemite National Park, California [28]. Prolonging network lifetime is of paramount importance for this kind of applications, while delay usually can be tolerated, since real-time data analysis is usually not required.

- **Disaster Response:**

  Networked sensor nodes are envisioned to play a variety of roles at disaster scenes: embedded into emergency personnel's uniforms, sensor nodes can not only track locations and status of emergency personnel, but also expand emergency personnel's awareness, helping to locate victims and warning of dangerous situations; sensor nodes can be used to capture realtime vital signs from a large number of patients, directing emergency personnel to the patients who need immediate attention. In general, sensor network technology will ensure reliable and prioritized information flow at disaster scenes, and substantially improve disaster response efficiency. In disaster response applications, latency is a critical performance measure.

- **Traffic Control:**

  Wireless sensor networks have the potential to revolutionize transportation systems in many ways. Recently researchers have proposed "sensors on wheels" [13], a solution that turns vehicles themselves into sensor nodes. As vehicles travel around, they collect information on traffic conditions, air quality or any other events from their immediate environment. They either relay this information to nearby vehicles or report it upon receiving queries. Informed of realtime traffic information, drivers will be able to avoid traffic jams and optimize their routes. In such systems, the latency requirement is of high priority due to the realtime nature of the information.

1.2 Background and Motivation

In order to take advantage of the aggregated power of the large number of sensor nodes and overcome the limited capacities of individual sensor nodes, sensor networks are usually considered as a distributed data collection and storage scheme. The primary function of sensor networks is then to collect data from the physical environment and to answer queries from users [4].

In many sensor network applications, individual sensor readings are inherently unreliable due to environmental dynamics and severe resource constraints placed on individual sensor nodes. Focusing on the overall picture rather than individual sensor readings provides a more accurate description of the physical environment. In a canonical usage scenario of sensor networks, each sensor node would collect a small piece of information such as temperature, density of gases, or radiation of nuclear signals from its immediate environment. An overall picture can be obtained by aggregating many small pieces of information collected by individual sensor nodes. Some of the most common queries over sensor networks are aggregation queries such as MAX, MIN, AVERAGE, COUNT, SUM, QUANTILE, the most frequent data values, and range queries.

In-network aggregation is a technique that aggregates data *en route* from sources to destinations to reduce data redundancy. Since communication consumes more energy than computation [38], significant energy gain is obtained by carrying out local computation to reduce the amount of data that need to be transported. In-network aggregation is the key to improving scalability and prolonging the lifetime of sensor networks as it significantly increases the energy efficiency of the networks.

While in-network aggregation improves energy efficiency, it also introduces significant aggregation delay. On the path from the sources to the destination, data from nearer sources have to wait for data from farther sources at intermediate nodes before they can be aggregated. The aggregation delay is a function of the number of hops between the destination and the farthest sources [22]. In addition, to maximize the degree of aggregation within the network, data tend to be routed through the paths that promote aggregation rather than the shortest path, which contributes additional delay.

The multi-hop nature of sensor networks also contributes to the overall delay of sensor networks. Multi-hop communication is the norm in sensor networks due to the short transmission ranges of sensor nodes. Moreover short links are preferred over longer links in sensor networks for the sake of energy efficiency, resulting in non-optimal routes in terms of delay.

In many applications, sensor nodes are redundantly deployed to compensate for their low reliability and harsh environments. Therefore, sensor networks usually have high node density. Sensor nodes will interfere with their neighbors by congesting the channels. Hence the channel access delay of sensor networks is more severe than that of traditional wireless networks.

As we elaborated in the last section, sensor networks promise a wide range of applications which have very different delay requirements. It is important that sensor networks provide a delay differentiation facility that permits applications to exploit desired tradeoffs between delay and other performance metrics. While there has been increasing interest in the delay problem in sensor networks [1, 5, 50], such delay differentiation schemes are missing.

Collecting exact data from sensor networks can be challenging if not impossible due to the large number of sensor nodes involved, environmental noise, interference between sensor nodes, and the like. In many situations an approximate answer obtained immediately is more useful than an exact answer that comes after a long delay. As an example, consider a sensor network monitoring toxic chemical gases in an airport: an approximate density of some toxic gases reported within seconds is desired instead of an exact density reported later. While various techniques for answering approximate queries have been developed, most of them lack provable guarantees for the accuracy of the answers.

Many applications require sensor nodes to periodically collect data from the environment. Consider the query "*For the next 10 days, report the average temperature on the summit of the Blackcomb Mountain every two hours*". Environmental quantities usually change slowly over time; thus consecutive sensor readings are temporally correlated to a certain extent. How to exploit this feature of sensor readings to develop efficient communication scheme is an important research topic for sensor networks.

1.3 Contributions of this Thesis

In this thesis, we study fast data aggregation techniques for large wireless sensor networks. Data aggregation in sensor networks has been extensively studied in the past few years. While most of the existing work on data aggregation focuses on energy efficiency, this thesis concentrates on time-efficiency: the main goal is to provide techniques to speed up aggregation queries over large wireless sensor networks.

To this end, we develop a simple distributed architecture for a sensor network with densely and uniformly deployed sensor nodes. We organize the sensor network into a number of layers, where each layer consists of a random subset of the sensor nodes, in decreasing order of its size as one goes from a lower layer to a higher layer. In this model, any aggregation query will be addressed to a particular layer, which produces an estimate of the aggregate of the whole network. Different layers of the network represent different accuracy levels and different delays. Since only a subset of sensor nodes participate in answering each query, the data aggregation process involves fewer hops, and the aggregation delay at the intermediate nodes is decreased. Thus the data aggregation process is sped up.

The data aggregation scheme proposed in this thesis is fully distributed. The layers of the network are constructed in a distributed, localized fashion. Each sensor node decides locally on which layer(s) it will exist without any communication with the outside world. Sensor nodes do not need to keep information about other sensor nodes.

The layered architecture proposed in this thesis differs from existing hierarchical architectures in the following way. Most existing hierarchical architectures, such as those proposed in [17, 26, 33, 49], aim at clustering the sensor nodes so that cluster heads can aggregate data from cluster members to reduce data redundancy for saving energy. By contrast, there are no clusters nor cluster heads in our layered architecture. Each sensor node in our network only represents itself and submits its own data if it is involved in a particular query.

In our layered architecture, the reduction in delay comes with a price tag: since only a subset of the sensor nodes submit their data, the accuracy of the aggregates of the network is compromised. We study the tradeoff between the delay and the accuracy in the context of five aggregates of the network: MAX, MIN, QUANTILE, AVERAGE and SUM. Given user-specified accuracy requirements, we analyze which layer of the network should be queried to obtain answers with theoretically guaranteed accuracy. This provides applications with the ability to effect desired tradeoffs between accuracy and delay. We show that different queries exhibit distinct characteristics which affect the delay-accuracy tradeoff.

We then improve our scheme to further speed up the data aggregation process by exploiting the statistical properties of the data. Given statistical information obtained from the history of the environment, our improved scheme further reduces the number of sensor nodes involved in answering certain types of queries such as AVERAGE and SUM. We then investigate the new tradeoffs between delay and accuracy given the additional information.

We also consider how to balance the power consumption at each sensor node to extend the overall lifetime of the network. The layered structure of the network needs to be reconstructed periodically for the sensor nodes to serve evenly on different layers. This leads to relatively uniform energy consumption across all sensor nodes in the network, thus resulting in an increase in the life expectancy of the whole network.

1.4 Organization of this Thesis

The material covered in Chapters 3, 4, 5 and 6 is adapted from the conference paper [43] by Wang, Ergun and myself. The rest of this thesis is organized as follows.

Chapter 2 is a review of related work on sensor networks. We focus on three areas pertaining to this thesis: in-network aggregation, delay in sensor networks, and periodic queries for temporally correlated data.

In Chapter 3, we present a randomized layered architecture for wireless sensor networks. We first discuss the network model with which we will be working. We then describe the construction process and the properties of the layered network. Finally we discuss how data is collected and aggregated in the layered network.

In Chapter 4, we analyze the tradeoffs between the accuracy of the query answers and the latency of the query response in the layered network.

In Chapter 5, we use numerical simulations to evaluate the theoretical expressions for the tradeoffs between the accuracy and the latency, and to observe the effects of various parameters on the performance of the layered network.

In Chapter 6, we address the energy consumption issues of the layered network.

In Chapter 7, we summarize this thesis and discuss some open problems and future work.

Chapter 2

Related Work

Early visionary works on wireless sensor networks appeared around the year 2000 [21, 12, 38]; these works envisioned an unusual range of applications promised by large scale wireless sensor networks, and motivated the challenges presented by such networks. Since then, there has been a growing body of research work that addresses many aspects involved in realizing the vision of such networks. A comprehensive overview of wireless sensor networks can be found in the survey paper [2] by Akyildiz *et al.* In this chapter, we review important research efforts on three areas related to this thesis: in-network aggregation, delay in sensor networks and periodic queries for temporally correlated data.

2.1 In-Network Aggregation

Almost all techniques for in-network aggregation require the construction of routing trees from multiple sources to a single destination. There is a body of work that investigates various schemes for building aggregation trees for spatially correlated data [37, 11, 9, 42]. For a wide range of spatial correlations, Pattem *et al.* [37] evaluate the energy consumption of three routing schemes: distributed source coding, routing driven compression, and compression driven routing, and propose a cluster-based tree building scheme that achieves near-optimal energy saving. Building on the work of Pattem *et al.* [37], Enachescu *et al.* [11] present a randomized opportunistic aggregation scheme for the grid topology, which is a constant factor approximation to the optimal aggregation tree.

Data-centric routing is one of the key techniques that support in-network aggregation.

The intuition behind data-centric routing is that applications usually need to access aggregated data rather than individual nodes. Based on data rather than the identities of data sources and destinations, data-centric routing aims to find paths from multiple sources to a single destination that promote data aggregation. One of the most cited techniques for data-centric routing is directed diffusion [20]: data instead of the nodes generating the data are named; if nodes express their interest for named data, data matching the interest are sent to them; data from multiple sources are aggregated whenever appropriate at intermediate nodes *en route* to the destination. A more general look at data-centric routing is provided in the work of Krishnamachari *et al.* [23] where data-centric routing is compared with address-centric routing in terms of energy savings, delay and robustness. Here address-centric routing refers to traditional end-to-end routing where multiple sources send data independently to the destination along the shortest path. Several variations with similar concepts can be found in the literature [16, 15, 24].

In-network aggregation has been studied in the database context. Viewing sensor networks as a distributed database, researchers at UC Berkeley [30, 29] and Cornell University [47, 48] have developed query processing architectures for sensor networks. To reduce energy consumption, they explore in-network aggregation techniques for a variety of database aggregate operators including MIN, MAX, COUNT, SUM, and AVERAGE. These four papers, however, do not provide in-network aggregation techniques for optimizing complex queries such as MEDIAN. For simple aggregates MIN, MAX, COUNT, SUM, and AVERAGE, intermediate nodes can aggregate all the data they receive to a single message of constant size before sending it to their parents. On the other hand, to compute exact answers to aggregation queries such as MEDIAN, all distinct sensor values have to be forwarded to the destination. Thus for such complex queries, the required message size increases with the size of the network.

To address the issue of large message size for complex queries such as MEDIAN, Shrivastava *et al.* [39] propose an aggregation scheme that provides approximate answers with bounded errors using fixed message size. Greenwald and Khanna [14] develop a distributed algorithm to compute approximate *quantiles* within a given error ϵ using message size of $O(\log^2 n/\epsilon)$ per sensor node, where n is the number of sensor nodes in the network.

A recent result [7] classifies aggregation queries into fully-aggregated query, unaggregated query and partially aggregated query based on the level of aggregation that can be employed at the intermediate nodes. It considers power-aware routing and aggregation query processing together, building energy-efficient routing trees explicitly for each type of aggregation queries.

Another approach to in-network aggregation is the use of hierarchies, where sensor nodes are usually organized into clusters. Within each cluster, cluster members send their data to a node designated as "cluster head", which will perform data aggregation to reduce the amount of data being transmitted to the destination or an upper level cluster head. In LEACH [17], the role of cluster head is rotated randomly among various sensor nodes to avoid draining the batteries of a few sensor nodes; each cluster head periodically collects data from its members and aggregates the data before sending them to the destination. In HEED [49], cluster heads are selected based on the residual energy of the sensor nodes; a secondary parameter such as node proximity to neighbors or node degree is used to "break ties". In TEEN [32], each sensor node collects data from the environment continuously, but only sends the data to its cluster head when the data value is greater than a hard threshold and the data value differs from the previous-sensed data value by more than a soft threshold. Here the hard threshold specifies the range of interest where data will be reported, and the soft threshold controls the tradeoff between energy efficiency and data accuracy.

2.2 Delay in Sensor Networks

As sensor networks are being deployed for more and more missions of critical nature, delay issues have drawn increasing research attention. Researchers have studied delay problems in sensor networks from different perspectives. Krishnamachari *et al.* [22] empirically show that greater delay is a tradeoff of energy gains achieved by in-network aggregation. Kunniyur and Narasimhan [25] derive a model to study the effect of channel access probability, transmission radius, network load and network density on delay in wireless sensor networks.

Intanagonwiwat *et al.* [19] propose a variant of directed diffusion, in which intermediate nodes send data to their parents after waiting a period of time or receiving a sufficient amount of data for aggregation. The accuracy of the aggregation will depend on the delay allowed at the intermediate nodes, which is specified by the application.

Yu *et al.* [50] explore the energy-latency tradeoffs in a scenario where the data gathering must be performed within a latency constraint. They present algorithms to minimize the overall energy consumption in the aggregation tree subject to the latency constraint.

In the work [36] by Narasimhan and Kunniyur, each data packet is assigned a power budget, which is the total excess energy that can be used by any intermediate nodes to send this packet through any low latency route instead of the energy efficient route. Hence the delay of a packet can be controlled by adjusting its power budget.

Boukerche *et al.* [5] propose a fault tolerant and low latency protocol for critical condition monitoring applications, where response time assumes primary importance.

2.3 Periodic Queries for Temporally Correlated Data

Consecutive sensor observations of many physical phenomena are usually temporally correlated. Recently there have been several research efforts that exploit temporal correlation between sensor data to develop efficient communication protocols for sensor networks.

Akyildiz *et al.* [3] consider an event signal, which is a Gaussian random process in time. Sensor nodes periodically report their data to a destination that wants to find the expectation of the signal over a decision interval. Under the assumption that all sensor nodes in the event area will report the same data during the decision interval, the data reporting frequency is determined so that the estimate expectation of the signal achieves a desired accuracy.

In another interesting study [10], Cristescu and Vetterli show that there is an optimal node density for gathering spatio-temporally correlated data in sensor networks. They consider a scenario where data will be distorted by delay. Given the data measured at the sensor nodes, and the knowledge of the spatio-temporal correlation of the data, the destination needs to reconstruct the data at all points at the measured area. A high node density gives a good spatial approximation, while a low node density gives a good temporal approximation. They aim to find an optimal node density such that the quality of the overall approximation is optimal.

Chapter 3

A Layered Architecture for Data Aggregation

In this chapter, we present a randomized layered architecture for retrieving aggregated information efficiently from a large wireless sensor network. The following is the network scenario considered in this thesis. The task of each sensor node is to measure some quantities such as temperature, humidity and particle concentrations from its immediate environment. The sensor network will provide users with aggregated information which summarizes the data collected by individual sensor nodes over the entire sensor field. Aggregation queries are the primary operation on sensor networks, as they are robust to individual node failures and allow in-network processing to reduce communication cost. Due to the large number of sensor nodes involved, extracting exact aggregated information from sensor networks can be very challenging in terms of response time and energy consumption. In most sensor network applications, aggregated information with 100% accuracy is not necessary; a good estimate of it is sufficient. The randomized layered architecture developed in this chapter provides users with the ability to quickly obtain approximate aggregated information with desired accuracy.

We start this chapter with the network model and the assumptions under which this research is conducted. We then describe how the layered architecture of our network is constructed and maintained. Following a discussion of the properties of the layered architecture, we expound on the data collection and aggregation process in the layered network.

3.1 Network Model

We consider a large wireless sensor network where sensor nodes are densely deployed. We describe the network model with which we will be working as follows.

1) The sensor network has N sensor nodes denoted by $s_1, s_2, \ldots, s_N$.

2) The sensor nodes are deployed uniformly at random in a square area with side length D. More specifically, let (X, Y) be the coordinates of a sensor node in the two-dimensional plane, then X, Y are independent random variables uniformly distributed in the integer interval $[0, D)$.

3) There exists a base station which serves as the gateway for extracting data from the sensor network.

4) The number of queries received by the sensor network is a Poisson process with rate λ.

5) Each sensor node can adjust its transmission range, not to exceed maximum value R.

6) The energy consumed by a sensor node for a transmission, denoted by e, is proportional to r^α, where r is the transmission range used for the transmission, and α is the path-loss exponent experienced by radio transmission. The value of α varies in the interval [2, 4] depending on the communication medium [44]. Without loss of generality, we normalize the energy coefficient to 1, hence $e = r^\alpha$.

7) With the exception of the base station, all the sensor nodes are assumed to be homogeneous and begin with the same initial energy, denoted by B.

3.2 Layered Network Construction

We will embed a layered structure on the network. There are L layers on the network which are numbered $0, 1, 2, \ldots, L-1$, with layer 0 as the base layer and layer $L-1$ as the top layer. Each layer will be assigned a transmission range. The method of determining the value of L will be discussed in later sections. We use $r(l)$ to denote the transmission range used on layer l: during a query taking place on layer l, the sensor nodes on layer l communicate using transmission range $r(l)$, and can reach each other in one or more hops.

```
Procedure SelfPromoting
Input: $L$: the number of layers of the network, $p$: the promotion probability
Output: the highest layer where sensor node $s_i$ exists
begin
    1.  $l := 0$;
    2.  if $l = L - 1$ then
    3.      return $l$;
    4.  else
    5.      $s_i$ promotes itself to layer $l + 1$ with probability $p$;
    6.  if $s_i$ fails to promote itself then
    7.      return $l$;
    8.  else
    9.      $l := l + 1$;
    10.     Goto step 2;
end
```

Figure 3.1: The decision process for a generic sensor node s_i

Given that data communication is the dominant reason for energy consumption in sensor networks and communication links in sensor networks are unreliable, it is desirable that the layered structure be constructed and maintained locally to provide network scalability and robustness. We now expound on how the layered structure is constructed locally. The base station initiates the network construction by sending out a broadcast message, which specifies a value L, the number of layers of the network, and a value $0 < p < 1$, the *promotion probability.* The layers in the network are constructed as follows. Once a sensor node s_i receives the broadcast message, it decides, without any communication with any other sensor nodes or the base station, to which layer(s) it will belong. By default, all sensor nodes, including s_i, exist on the base layer, which is layer 0. Inductively, if s_i exists on some layer l, it will, with probability p, promote itself to layer $l + 1$, which means that s_i will exist on layer $l + 1$ in addition to all the lower layers $l, l - 1, \ldots, 0$. If on some layer l', s_i makes the decision not to promote itself to layer $l' + 1$, s_i stops the randomized procedure and will not exist on any higher layers. If s_i promotes itself to the highest layer $L - 1$, it stops the promotion procedure since no sensor node is allowed to exist beyond layer $L - 1$.

Figure 3.1 gives a formal description of the above decision process for a generic sensor node s_i. For a sensor network with N sensor nodes, there will be N copies of the procedure running in parallel independently during the layered network construction.

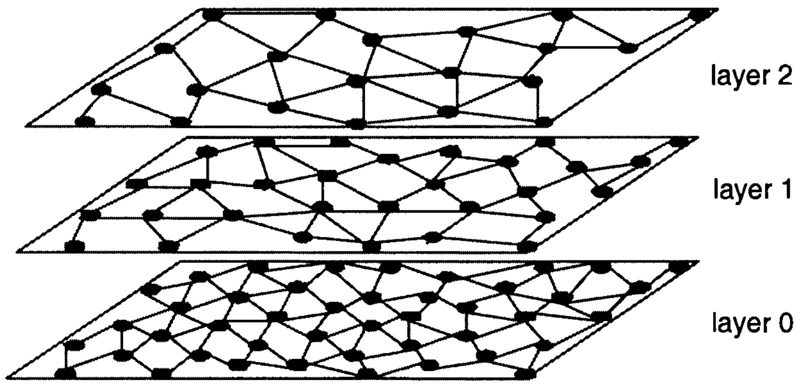

Figure 3.2: A layered sensor network with three layers

The network construction does not assume the existence of any synchronization mechanism. Each sensor node starts its own promotion procedure upon receiving the construction request from the base station. Those sensor nodes in the lower part of the broadcast tree will be late in starting their promotion procedures. Since the construction scheme works in a distributed fashion, this is not a problem – the late sensor nodes can simply promote themselves using probability p and join their related layers in their own time. Nevertheless, the base station needs to make sure the network construction is finished before it starts accepting queries from the users.

Figure 3.2 shows the layered structure of a sensor network with three layers. On each layer, there is a link connecting two nodes if they are within each other's transmission range on that layer.

3.3 Analysis of the Layered Architecture

In this section, we first analyze the fundamental properties of the randomized layered architecture. The key parameters of the layered network are the number of layers of the network, the promotion probability and the transmission range used by the sensor nodes on a specific layer. We discuss how to determine these key parameters of the layered network, and how the choices of these parameters affect the layered network.

3.3.1 Properties of the Layered Architecture

A natural question that arises at this point would be how many sensor nodes one will see on a particular layer. Since the layered network construction is a random process, the number of sensor nodes on each layer is a random variable. To be able to exist on a high layer, a sensor node needs to succeed continuously in promoting itself. One would expect to see fewer sensor nodes going from a lower layer to a higher layer. The following lemma shows how many sensor nodes are expected to exist on a specific layer.

Lemma 3.1. *For a sensor network with N sensor nodes and promotion probability p, the* expected number of sensor nodes *on layer l is $N \cdot p^l$.*

Proof: For $i = 1, \dots, N$, define indicator random variable Y_i as follows.

$$Y_i = \begin{cases} 1 & \text{if sensor node } s_i \text{ is promoted to layer } l; \\ 0 & \text{otherwise.} \end{cases}$$

Let Y be the random variable describing the number of sensor nodes on layer l. Then

$$Y = \sum_{i=1}^{N} Y_i.$$

During the network construction, the decision process for each sensor node consists of a certain number of Bernoulli trials, each of which determines whether or not a sensor node will promote itself one layer up. Let a sensor node's decision for self-promoting from layer $j-1$ to layer j be the jth trial. Since a sensor node stops its promotion procedure once it fails to promote itself to a higher layer, for a sensor node to exist on layer l, it needs to succeed in all of the first l trials. Clearly, these l trials are mutually independent, and each has the same probability p for success. Hence $Pr[Y_i = 1] = p^l$, and $Pr[Y_i = 0] = 1 - p^l$.

The expected number of sensor nodes on layer l can be computed as follows:

$$E[Y] = E\left[\sum_{i=1}^{N} Y_i\right] = \sum_{i=1}^{N} E[Y_i] = \sum_{i=1}^{N} (1 \cdot p^l + 0 \cdot (1 - p^l)) = N \cdot p^l \qquad \square$$

The properties of the randomized layered architecture can be summarized as follows:

1) The *base layer* contains all sensor nodes $s_1, \dots, s_N$.

2) Any sensor node will exist on layers $0, 1, \dots, k$ for some $0 \leq k \leq L - 1$.

3) The sensor nodes on layer l form a subset of those on layer $l - 1$ for $1 \leq l \leq L - 1$.

4) The expected number of sensor nodes on layer l drops *exponentially* with $\frac{1}{p}$ as l increases.

3.3.2 Specifying the Structure of the Layers

We now discuss how to determine *the number of layers* of the network, *the promotion probability* and *the transmission range* used by the sensor nodes on a specific layer. These three parameters need to be specified before the layered network construction.

We first introduce a definition for the network density due to Bulusu *et al.* [6]. In this definition, *network density* is expressed in terms of the number of sensor nodes within the transmission radius around each sensor node.

Definition 3.2. (Network Density) *Consider a sensor network with N sensor nodes scattered in a region of area A, let r be the transmission range of each sensor node, the* network density $\mu(r)$ *is defined as*

$$\mu(r) = \frac{N\pi r^2}{A}$$

Since there is a small probability that a sensor node might promote itself indefinitely, *the number of layers* in the network must be upper-bounded for the construction process to terminate. Another reason for "cutting off" the top of the layered structure is that, if a layer is populated with too few sensor nodes, the inter-sensor distance on that layer will exceed the maximum transmission range R. To keep the sensor nodes on the high layers connected, we do not allow more than L layers in the network, where $L = \Theta\left(\log_{\frac{1}{p}} \frac{N}{(\frac{D}{R})^2}\right)$, which is determined so that with high probability the inter-sensor distance on the top layer does not exceed the maximum transmission range R. The maximum number of layers in the network is thus picked as a function of the promotion probability and the network density which is calculated with all sensor nodes operating at the maximum transmission range.

We now determine *the transmission range* used on each layer. The transmission range used during a communication taking place on layer l, where $l = 0, \dots, L - 1$, is determined by the expected distance between two neighboring sensor nodes on layer l, i.e. $r(l) = \frac{D}{\sqrt{N \cdot p^l}}$, and can be enlarged a little further to ensure a higher chance of connectivity.

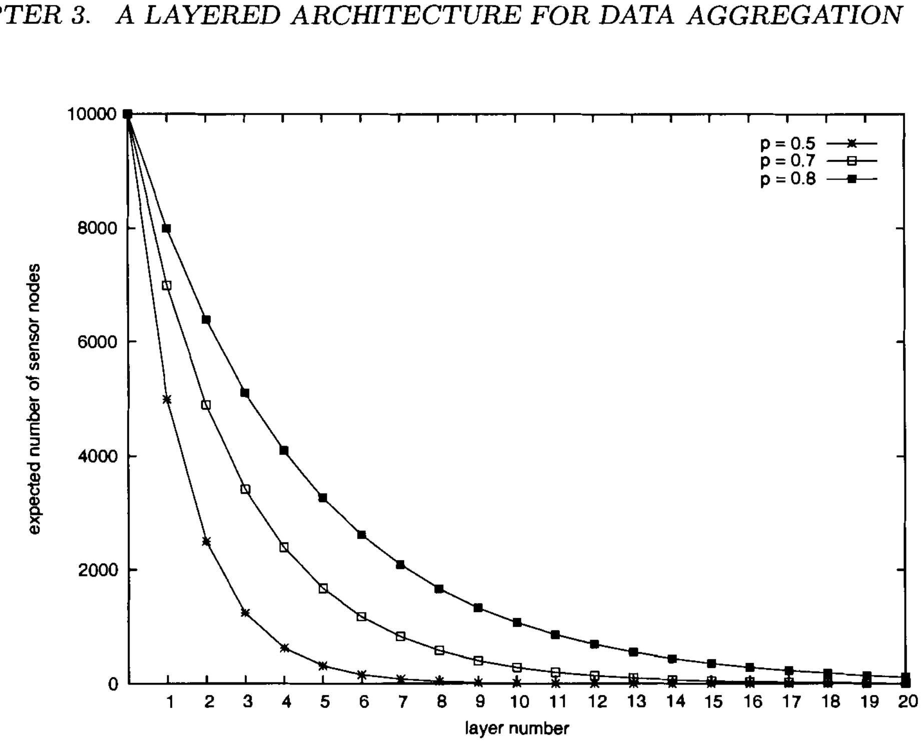

Figure 3.3: Expected number of sensor nodes vs. layer number for various values of promotion probability p for a sensor network with $N = 10000$ sensor nodes

The promotion probability determines how quickly the number of sensor nodes decreases as the layer number increases. Figure 3.3 shows the impact of the promotion probability on the expected number of sensor nodes on different layers. Since there must be enough sensor nodes on the top layer to maintain the network connectivity, the promotion probability also has a direct impact on the maximum number of layers that the sensor network can have.

3.4 Data Collection and Aggregation in a Layered Network

Given a layered sensor network, we now focus on how a query is disseminated across the network and how data are collected and aggregated in the network. We simplify the situation by assuming the same as [39] that the base station is a special node where a query will be initiated and the aggregated data will be converged. Thus the base station acts as an interface between the sensor network and the external users.

When the base station receives a query from the users, it first determines which layer is to be used for this query (the method for which will be discussed in detail in the next chapter).

Let this layer be l. The base station then initiates the query by sending out a message whose recipients are all the sensor nodes on layer l. This can be achieved by reserving a small field (of $\log\log N$ bits) in the transmission packet for the layer number. In this message, the base station also specifies the query type (such as MAX, MIN, QUANTILE, AVERAGE or SUM). Any sensor node on layer l that hears this message will relay it using communication range $r(l)$; those sensor nodes not on layer l will simply ignore this message.

After the query message is received by all of the sensor nodes on layer l, a routing tree rooted at the base station is formed. Each leaf node then collects its data and sends the data to its parent, which then aggregates its own data with the data from its children, relaying the aggregated data up to its own parent. The data collected by the sensor nodes on layer l will be propagated in this fashion all the way up the tree and will converge at the root. Once the root has the aggregated information, it can compute the answer to the query.

The schemes proposed in this thesis are independent of the algorithms for building routing/aggregation trees. Once the layer to be used for a particular query has been identified, the particular algorithm used for building the routing/aggregation tree is transparent to our schemes. Any existing techniques, such as those proposed in the literature [20, 19, 37, 7], can be applied to the distribution of the query and the collection of the data from the network.

Chapter 4

Querying the Layered Network: Accuracy vs. Latency

In this chapter, we show how the layered architecture presented in Chapter 3 can be used to support fast data aggregation in large wireless sensor networks.

The core idea behind the layered architecture is to trade off the accuracy of query answers for low query latency. As we describe in Chapter 3, any query on the layered network will be addressed to a specific layer. When a query utilizes data from all the sensor nodes, the answer is accurate; however, when data from only a subset of the sensor nodes are used, errors are introduced. The more sensor nodes participate in a query, the more accurate is the answer. In the mean time, the delay incurred in obtaining the query answer is also longer.

In sensor network applications such as critical condition monitoring and real-time target tracking in battle fields, the latency of query response has a critical impact on application performance. It is desirable that sensor networks be able to respond quickly to queries from their users and the accuracy of the answers be guaranteed to be above a certain threshold. The latency of a query response is a function of the number of sensor nodes whose data are being utilized for a particular query. Thus, in the layered network the latency of a query response is reflected by the layer to which the query is sent. In this chapter, we investigate, given a query with an accuracy requirement, which layer of the network should be queried to obtain an answer with a theoretically guaranteed level of accuracy.

As any queries on the layered network will be addressed to a specific layer which contains a subset of the sensor nodes, we expect approximate rather than exact answers from the layered network. We start our discussion by determining how to relax the accuracy requirement of a query.

4.1 Relaxing the Accuracy Requirement

Consider some aggregation function f over the set S of all sensor values in the network. Example aggregation functions are MAX, MIN, QUANTILE, AVERAGE and SUM. Let $\hat{f}$ be the output of our approximate aggregation scheme. We would like, with high probability, the deviation of $\hat{f}(S)$ from $f(S)$ not to exceed a certain threshold, for any input S. We define the accuracy requirement formally as follows.

Definition 4.1. (Accuracy requirement) *Let S be the set of all sensor values in the network. Let f be an aggregation function over $\mathcal{S}$. Let $\hat{f}(\mathcal{S})$ be an approximation of $f(\mathcal{S})$ produced by some approximate aggregation scheme. The* accuracy requirement *of an aggregation query for $f(\mathcal{S})$ is, for $\epsilon > 0$ and $0 < \delta < 1$,*

$$Pr\left[|f(\mathcal{S}) - \hat{f}(\mathcal{S})| \geq \epsilon\right] \leq \delta$$

In the rest of this thesis, we will refer to ϵ as the *error bound* and δ as the *error probability.* These two parameters will be specified by query users.

4.2 Analysis of the Accuracy and the Latency

In this section, we investigate the accuracy and the latency tradeoff in the context of certain aggregation functions. Given an aggregation query with an accuracy requirement, we perform formal analysis to determine which layer should be queried to obtain an approximation with desired accuracy.

The accuracy of a query answer directly relates to the number of sensor nodes whose data are utilized for the query. Due to the random nature of the construction process of the layered network, the number of sensor nodes on each layer is a random variable. In the following lemma, we investigate which layer must be queried if one would like to have input from at least k sensor nodes with a high probability.

Lemma 4.2. *Consider a sensor network with N sensor nodes and promotion probability p, let μ be the expected number of sensor nodes on layer l. For $k \leq \mu$, if $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(k + ln\frac{1}{\delta} + \sqrt{ln\frac{1}{\delta}(2k + ln\frac{1}{\delta})}\right)$, then the probability that there are fewer than k sensor nodes on layer l is smaller than δ.*

Proof: For $i = 1, \ldots, N$, let Y_i be the indicator random variable for whether sensor node s_i is promoted to layer l. Clearly, $Y_1, \ldots, Y_N$ are independent. On layer l there are $Y = \sum_{i=1}^{N} Y_i$ sensor nodes. By Lemma 3.1, $\mu = E[Y] = N \cdot p^l$.

$$\begin{aligned} Pr[Y < k] &= Pr\left[Y < \frac{k}{\mu}\mu\right] \\ &< e^{-\frac{\mu}{2}(1-\frac{k}{\mu})^2} \end{aligned}$$

by Chernoff's inequality (see Lemma A.1 in Appendix A).

Solving $e^{-\frac{N \cdot p^l}{2}(1-\frac{k}{N \cdot p^l})^2} < \delta$ for l, we have

$$l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(k + ln\frac{1}{\delta} + \sqrt{ln\frac{1}{\delta}(2k + ln\frac{1}{\delta})}\right). \qquad \square$$

In general, exact answers to MAX or MIN queries cannot be obtained unless all sensor nodes in the network contribute to the answers, since any missed sensor node might have an arbitrarily high or low data value. The following theorem is immediate.

Theorem 4.3. *The queries for* MAX *and* MIN *must be sent to the base layer to avoid arbitrarily high error.*

Since we assume the number of sensor nodes in the network is given, once the AVERAGE is obtained, the SUM is just the AVERAGE times the number of sensor nodes. We thus will not explicitly explain the SUM queries.

In what follows, we will focus on two aggregation functions: QUANTILE and AVERAGE. We consider two types of aggregations: *snapshot aggregation* and *periodic aggregation*. *Snapshot aggregation* refers to an aggregation of certain quantity such as temperature, pressure, or humidity of the network at a point in time. In many application scenarios, users need to periodically query the network for some aggregated quantity. This kind of aggregation is usually called *periodic aggregation*. We first analyze QUANTILE queries and AVERAGE queries in the context of *snapshot aggregation*. We then refine our scheme to further speed up AVERAGE queries for *periodic aggregation*, exploiting statistical information about the sensor readings over time. Finally we investigate the new tradeoffs between delay and accuracy under this improved scheme.

4.2.1 QUANTILE Queries

Consider a sensor network with N sensor nodes deployed for detecting some chemical agent in a physical environment. Let $a_1, \ldots, a_N$ be the sensor readings of the N sensor nodes. Consider the following query "*Which sensor reading is greater than* 80% *of all the sensor readings*?". Queries of this type are called QUANTILE queries. Since *quantiles* are less sensitive to outliers than *average* and *sum*, QUANTILE queries are useful in rough environments where sensor readings can be very unreliable. Formally, *quantile* is defined as follows:

Definition 4.4. (Quantile) *Let $\mathcal{S}$ be a data set. Let $\hat{\mathcal{S}}$ denote the ordered sequence obtained by sorting the elements in $\mathcal{S}$ in non-decreasing order. For $a_i \in \mathcal{S}$, denote by $r(a_i)$ the rank of a_i in $\hat{\mathcal{S}}$. For $0 < \phi \leq 1$, if $r(a_i) = \phi|\mathcal{S}|$, then a_i is the ϕ-*quantile *of $\mathcal{S}$.*

As we can see, the minimum element is the $\frac{1}{|\mathcal{S}|}$-*quantile*, and the maximum element is the 1-*quantile*. MAX and MIN queries are special cases of QUANTILE queries. Due to similar reasons as the MAX and MIN queries, we cannot obtain exact *quantiles* by querying a proper subset of the sensor nodes in the network. We relax our requirement to obtain a sensor value whose rank is close to the exact *quantile*:

Definition 4.5. (ϵ-approximation Quantile) *Let $\mathcal{S}$ be a data set. Let $\hat{\mathcal{S}}$ denote the ordered sequence obtained by sorting the elements in $\mathcal{S}$ in non-decreasing order. For $a_i \in \mathcal{S}$, denote by $r(a_i)$ the rank of a_i in $\hat{\mathcal{S}}$. For $0 \leq \epsilon < \phi \leq 1$, and $\phi + \epsilon \leq 1$, if $a_i \in \{x \in \mathcal{S} \mid (\phi - \epsilon)|\mathcal{S}| \leq r(x) \leq (\phi + \epsilon)|\mathcal{S}|\}$, then a_i is an ϵ-*approximation ϕ-quantile *of $\mathcal{S}$.*

From now on, for ease of exposition, whenever we mention a *rank* in a set, we mean this *rank* is over a sequence obtained by sorting the elements of the set.

We can observe, the difference in rank between the ϕ-*quantile* and ϵ-*approximation ϕ-quantile* is at most ϵ fraction of the size of the input set. To satisfy a query for ϕ-*quantile* with error bound ϵ and error probability δ, our scheme should return an ϵ-*approximation ϕ-quantile* with probability at least $1 - \delta$.

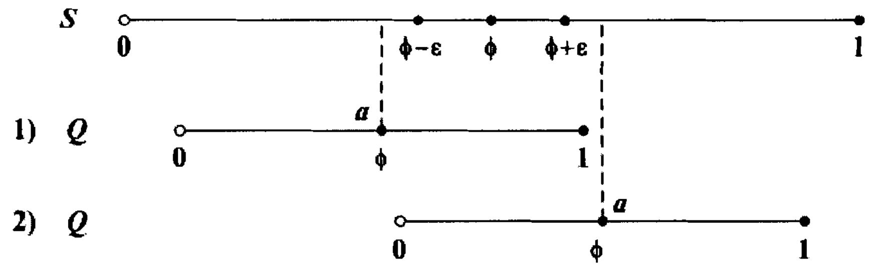

Figure 4.1: The ϕ*-quantile* of $\mathcal{Q}$ falls outside the rank between $(\phi-\epsilon)|\mathcal{S}|$ and $(\phi+\epsilon)|\mathcal{S}|$ in $\mathcal{S}$.

To meet an accuracy requirement, we need to identify the sources contributing to the error of a query. First, since the layers are constructed in a randomized fashion, the number of sensor nodes on a given layer cannot be known exactly. In Lemma 4.2, we solve this problem by showing which layer should be queried if we would like to have input from at least a certain number of sensor nodes with at least a given probability. The second source of error is the fact that a query is always addressed to a subset of the sensor nodes. How big should the size of this subset be to achieve the desired accuracy? The following lemma provides an answer to this question.

Lemma 4.6. *Let $\mathcal{S}$ be a data set. Let $\mathcal{Z} = \{\mathcal{S}_k \mid \mathcal{S}_k \subseteq \mathcal{S}, |\mathcal{S}_k| = k\}$. Let $\mathcal{Q}$ be picked at random from $\mathcal{Z}$. If $k \geq \frac{ln\frac{2}{\delta}}{2\epsilon^2}$, then, with probability greater than $1-\delta$, the ϕ-quantile of $\mathcal{Q}$ is an ϵ-approximation ϕ-quantile of $\mathcal{S}$.*

Proof: We will bound the probability that the ϕ*-quantile* of $\mathcal{Q}$ falls outside the rank between $(\phi-\epsilon)|\mathcal{S}|$ and $(\phi+\epsilon)|\mathcal{S}|$ in $\mathcal{S}$.

Claim: the element with rank $\phi|\mathcal{Q}|$ in $\mathcal{Q}$ does not have rank between $(\phi-\epsilon)|\mathcal{S}|$ and $(\phi+\epsilon)|\mathcal{S}|$ in $\mathcal{S}$ if and only if one of the following holds: 1) More than $\phi|\mathcal{Q}|$ elements in $\mathcal{Q}$ have rank less than $(\phi-\epsilon)|\mathcal{S}|$ in $\mathcal{S}$; or 2) more than $(1-\phi)|\mathcal{Q}|$ elements in $\mathcal{Q}$ have rank greater than $(\phi+\epsilon)|\mathcal{S}|$ in $\mathcal{S}$.

The above claim is obvious from Figure 4.1. Since $|\mathcal{Q}| = k$, the distribution of $\mathcal{Q}$ is identical to the distribution where k elements are picked uniformly at random without replacement from $\mathcal{S}$. This is due to the fact that any element of $\mathcal{S}$ is as likely to be included in $\mathcal{Q}$ as any other element in either scheme, and both schemes include k elements in $\mathcal{Q}$. Note that the distributions of the elements in $\mathcal{Q}$ are not independent.

Since the two distributions mentioned above are identical, we can think of the construction of $\mathcal{Q}$ as k random draws without replacement from a *0-1 box* that contains $|\mathcal{S}|$ items, of which those with rank less than $(\phi - \epsilon)|\mathcal{S}|$ are labeled "1" and the rest are labeled "0".

For $i = 1, \ldots, k$, let X_i be the random variable for the label of the ith element in $\mathcal{Q}$, then $Pr[X_i = 1] = \phi - \epsilon$, and $X = \sum_{i=1}^{k} X_i$ is the number of elements in $\mathcal{Q}$ that have rank less than $(\phi - \epsilon)|\mathcal{S}|$ in $\mathcal{S}$. By the linearity of expectation,

$$E[X] = E\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} E[X_i] = \sum_{i=1}^{k}(\phi - \epsilon) = (\phi - \epsilon)k.$$

Therefore, by Hoeffding's inequality [1] (see Lemma A.3 in Appendix A),

$$\begin{aligned} Pr[X > \phi k] &= Pr\left[X - E[X] > \phi k - (\phi - \epsilon)k\right] \\ &= Pr[X - E[X] > \epsilon k] \\ &= Pr\left[\frac{X}{k} - E\left[\frac{X}{k}\right] > \epsilon\right] \\ &< e^{-2\epsilon^2 k}. \end{aligned}$$

Now the above notations remain unchanged except that the elements with rank greater than $(\phi+\epsilon)|\mathcal{S}|$ are labeled "1" and the rest are labeled "0". Then $X = \sum_{i=1}^{k} X_i$ is the number of elements in $\mathcal{Q}$ that have rank greater than $(\phi + \epsilon)|\mathcal{S}|$ in $\mathcal{S}$. Clearly, $E[X] = (1 - \phi - \epsilon)k$. Applying Hoeffding's inequality again,

$$\begin{aligned} Pr[X > (1 - \phi)k] &= Pr[X - E[X] > (1 - \phi)k - (1 - \phi - \epsilon)k] \\ &= Pr[X - E[X] > \epsilon k] \\ &< e^{-2\epsilon^2 k}. \end{aligned}$$

Hence the probability that the *ϕ-quantile* of $\mathcal{Q}$ is an *ϵ-approximation ϕ-quantile* of $\mathcal{S}$ is greater than $1 - 2e^{-2\epsilon^2 k}$. Setting $1 - 2e^{-2\epsilon^2 k} \geq 1 - \delta$, we have $k \geq \frac{ln\frac{2}{\delta}}{2\epsilon^2}$. $\square$

Lemma 4.6 indicates that, if a k-subset of set $\mathcal{S}$ is chosen at random from all the k-subsets of $\mathcal{S}$, and k is large enough, then this k-subset has similar *quantiles* to $\mathcal{S}$. We observe that

[1]Note that Hoeffding's inequality applies to random samples chosen without replacement from a finite population, as shown in Section 6 of Hoeffding's original paper [18], without the need for independence of the samples.

k is determined by the error bound ϵ and the error probability δ, and is independent of the size of $\mathcal{S}$.

Combining Lemmas 4.2 and 4.6, we now show which layer should be queried for a given error bound and error probability.

Theorem 4.7. *Let p be the promotion probability of a sensor network with N sensor nodes. If a ϕ-*quantile *query is sent to layer $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}}\left(\frac{ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta} + \sqrt{ln\frac{2}{\delta}(2\frac{ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta})}\right)$, then the answer will be an ϵ-*approximation *ϕ-*quantile *of the whole network with probability greater than $1-\delta$.*

Proof: Denote by B the event that the answer returned by layer l is not an ϵ-*approximation ϕ-quantile* of the whole network. Denote by A the event that layer l has fewer than k sensor nodes. Denote by $\overline{A}$ the complement of event A, the event that layer l has at least k sensor nodes.

Now consider the case $k = \frac{ln\frac{4}{\delta}}{2\epsilon^2}$:

1) By Lemma 4.2, if $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}}\left(\frac{ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta} + \sqrt{ln\frac{2}{\delta}(2\frac{ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta})}\right)$, then $Pr[\,A\,] < \frac{\delta}{2}$.

2) Since the sensor nodes on layer l forms a random subset of the sensor nodes of the sensor network, by Lemma 4.6, if the number of sensor nodes on layer l is at least $\frac{ln\frac{4}{\delta}}{2\epsilon^2}$, the probability that the *ϕ-quantile* of layer l is not an *ϵ-approximation ϕ-quantile* of the sensor network is less than $\frac{\delta}{2}$. Namely $Pr[\,B \mid \overline{A}\,] < \frac{\delta}{2}$.

By theorem of total probability [34],

$$\begin{aligned} Pr[\,B\,] &= Pr[\,B \mid A\,] \cdot Pr[\,A\,] + Pr[\,B \mid \overline{A}\,] \cdot Pr[\,\overline{A}\,] \\ &\leq Pr[\,A\,] + Pr[\,B \mid \overline{A}\,] \\ &< \frac{\delta}{2} + \frac{\delta}{2} = \delta. \end{aligned}$$

Hence the answer returned by layer $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}}\left(\frac{ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta} + \sqrt{ln\frac{2}{\delta}(2\frac{ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta})}\right)$ is an *ϵ-approximation ϕ-quantile* of the whole network with probability greater than $1-\delta$. □

4.2.2 AVERAGE Queries

In this section, we analyze the accuracy and the latency tradeoff for AVERAGE queries.

We now consider approximating the average data value over the whole sensor network by querying a particular layer. Here the goal is to obtain a value that is close to the true average almost all the time. How can we know a value is close to the true average without knowing what the true average is? We know from sampling theory [46] that the expectation of the average of random samples without replacement is equal to the average of the population. We can think of the sensor nodes on each layer as random samples without replacement from the sensor nodes of the whole network. Hence the following lemma is immediate.

Lemma 4.8. *For a sensor network with N sensor nodes, let $a_1, a_2, \ldots, a_N$ be the data values collected by the N sensor nodes. Let k be the number of sensor nodes on layer l. Let $X_1, X_2, \ldots, X_k$ be the random variables describing the data values collected by the k sensor nodes on layer l. Let $\overline{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$. Then $E[\overline{X}] = \frac{1}{N}\sum_{i=1}^{N} a_i$.*

We thus propose that the average returned by the queried layer be output as the average of the whole network. It remains to investigate the relationship between the accuracy requirement and the layer to be queried. As with QUANTILE queries, the error is due to the uncertainty of the number of sensor nodes on a specific layer and the utilization of data from only one layer instead of the whole network. The following lemma indicates at least how many sensor nodes we must query, given error bound ϵ and error probability δ.

Lemma 4.9. *Let $\mathcal{S}$ be a data set whose elements are bounded within the interval $[a, b]$. Let $\mathcal{Z} = \{\mathcal{S}_k \mid \mathcal{S}_k \subseteq \mathcal{S}, |\mathcal{S}_k| = k\}$. Let $\mathcal{Q}$ be picked at random from $\mathcal{Z}$. If $k \geq \frac{(b-a)^2 ln\frac{2}{\delta}}{2\epsilon^2}$, then, with probability less than δ, the average value of $\mathcal{Q}$ deviates from the average value of $\mathcal{S}$ by more than ϵ.*

Proof: The k elements in $\mathcal{Q}$ can be considered to be random samples without replacement from $\mathcal{S}$. Let $X_1, X_2, \ldots, X_k$ be the random variables describing the k data values in $\mathcal{Q}$. Then $a \leq X_i \leq b$ for $i = 1, 2, \ldots, k$. Let $\overline{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$. By Lemma 4.8, $E[\overline{X}]$ is the average value of $\mathcal{S}$. By Hoeffding's inequality, for any $\epsilon > 0$,

$$Pr[\,|\,\overline{X} - E[\overline{X}]\,| > \epsilon\,] < 2e^{\frac{-2k\epsilon^2}{(b-a)^2}}$$

Setting $2e^{\frac{-2k\epsilon^2}{(b-a)^2}} \leq \delta$, we have $k \geq \frac{(b-a)^2 ln\frac{2}{\delta}}{2\epsilon^2}$. $\square$

Combining Lemmas 4.2 and 4.9, we now show that the average of an appropriate layer constitutes an *ϵ-approximation* to the true average of the whole network with probability greater than $1 - \delta$. The proof of the following theorem is similar to that of Theorem 4.7.

Theorem 4.10. *Consider a sensor network with N sensor nodes whose data values are in the range $[a, b]$. Let p be the promotion probability, and let l be such that $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta} + \sqrt{ln\frac{2}{\delta}(2\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta})} \right)$. Then the probability that the average of layer l deviates from the true average of the whole network by more than ϵ is less than δ.*

Proof: Denote by B the event that the average of layer l deviates the true average of the whole network by more than ϵ. Denote by A the event that layer l has fewer than k sensor nodes. Denote by $\overline{A}$ the complement of event A, the event that layer l has at least k sensor nodes.

Consider the case $k = \frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2}$:

1) By Lemma 4.2, if $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta} + \sqrt{ln\frac{2}{\delta}(2\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta})} \right)$, then $Pr[\,A\,] < \frac{\delta}{2}$.

2) By Lemma 4.9, if the number of sensor nodes on layer l is at least $\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2}$, the probability that the average of layer l deviates the true average of the whole network by more than ϵ is less than $\frac{\delta}{2}$, that is, $Pr[\,B \mid \overline{A}\,] < \frac{\delta}{2}$.

By the theorem of total probability [34],

$$\begin{aligned} Pr[\,B\,] &= Pr[\,B \mid A\,] \cdot Pr[\,A\,] + Pr[\,B \mid \overline{A}\,] \cdot Pr[\,\overline{A}\,] \\ &\leq Pr[\,A\,] + Pr[\,B \mid \overline{A}\,] \\ &< \frac{\delta}{2} + \frac{\delta}{2} = \delta. \end{aligned}$$

Thus, if $l < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta} + \sqrt{ln\frac{2}{\delta}(2\frac{(b-a)^2 ln\frac{4}{\delta}}{2\epsilon^2} + ln\frac{2}{\delta})} \right)$, the probability that the average of layer l deviates from the true average of the whole network by more than ϵ is less than δ. □

4.2.3 Utilizing Statistical Information for Periodic Queries

Up to this point, the queries we consider have been *snapshot queries*. Sensor networks have been thought of as dynamic data repositories over which continuous queries can be run. Many sensor network applications need to periodically obtain some aggregate quantity of the environment such as temperature, pressure and particle concentrations. In most practical settings, these values observed by sensor nodes change slowly over time. The correlation between consecutive observations of a sensor node is usually referred to as *temporal correlation.* The characteristics of the correlation vary depending on the environmental quantity being monitored.

The correlation between consecutive sensor observations provide potential opportunities for developing efficient data gathering schemes tailored for sensor networks. If data values observed by sensor nodes are related to their previous observations, it is possible to use history information to help estimate current data values. In this section, we explore ways to further speed up the data gathering process, by leveraging the statistical information regarding the environmental quantity the sensor network is monitoring. For the case in which the change in data values over time follows the normal distribution, we are able to improve our scheme to reduce the latency even further for AVERAGE queries.

We now refine our scheme under the assumption that the change of each sensor value in one unit of time has the normal distribution. Since the sum of independent normal distributions is a normal distribution with mean equal to the sum of the individual means and variance equal to the sum of individual variances, the change in the average value also follows a normal distribution. To illustrate our assumption, consider the scenarios shown in Figures 4.2 and 4.3. The statistics of the history information about an area indicate that, the average temperature of the area is likely to rise around 10 degrees from 2 am to 12 pm, and fall around 6 degrees from 12 pm to 8 pm; small variations might happen but substantial changes are less likely.

To convey the intuition of our proposed strategy, consider the case in which the distribution of the change of the environment is known to be a normal distribution with mean μ. First we use the scheme proposed in Chapter 3 to obtain an initial estimate avg of the average data value in the network. By our analysis in the last section, avg is likely to be close to the true average. The change of the average in one unit of time, by our assumption, is expected to be μ. So the true average after one unit of time is likely to change by some value close to μ. Thus, $avg + \mu$ is likely to be a good estimate for the average for that point in time. However, new error has been added into our estimate. Since the scheme we use to obtain the initial avg queries only a subset of the sensor nodes, there is an error in avg. The new contribution to the error is our inability to know the exact change in the average value. Since the quantity of the error, as well as its likelihood increases, we need to make sure that the error of our estimate and the error probability remain at acceptable levels.

We now explain the main idea of our improved scheme for AVERAGE queries. We adopt

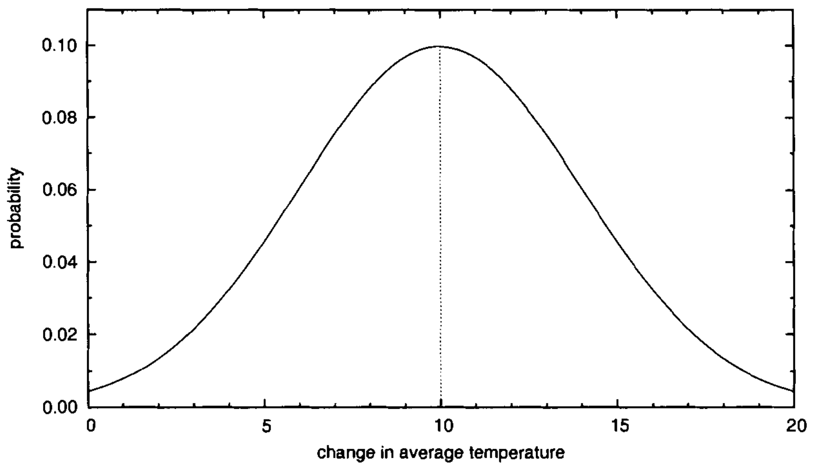

Figure 4.2: Probability distribution of change in average temperature from 2am to 12pm

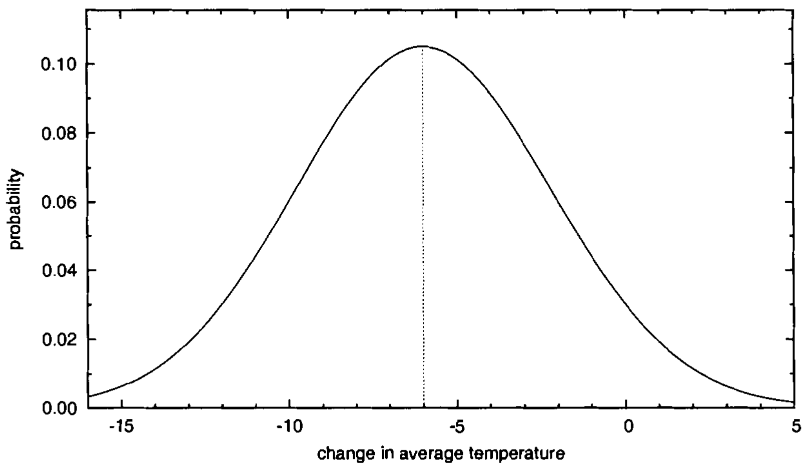

Figure 4.3: Probability distribution of change in average temperature from 12pm to 8pm

a multi-stage approach to our estimation of the average. In the first stage, which we call *QueryAverage*, we obtain an average value which is more accurate than the user-specified requirement, by querying a relatively large subset of the sensor nodes – more precisely, we query a layer which is lower than required to obtain an answer with error $\epsilon_1 < \epsilon$ and error probability $\delta_1 < \delta$. The purpose of doing this is to leave some room for extra error to be incurred in later stages.

In the following stage (after one time unit has elapsed), which we call *TestAverage*, and subsequent ones, we will query higher layers, thus involving a smaller number of sensor nodes, to see whether the expected change pattern is followed. The result of doing this is that either (a) we will boost the confidence to an acceptable level or (b) we will observe an "anomaly", that is, a deviation from expected behavior, which we will attempt to resolve by querying a lower layer with a larger number of sensor nodes. In case of (a), we will have obtained an answer with acceptable accuracy quickly by querying only a very small number of sensor nodes. Case (b) on the other hand is, by definition of the normal distribution, an anomaly that will not happen often. In the unlikely event of an "accident" near one of the sensor nodes, in the form of an atypical value, we first get informed quickly. Then our network will experience a longer query time for the sake of accuracy. In the long run, we will see more "expected" cases and will observe a lower average query time.

Before we formally present the algorithm, we use the example shown in Figure 4.4 to explain the major steps of the algorithm. In this scenario, the change of the average temperature has the normal distribution with mean $\mu = 10$ and standard deviation $\sigma = 3.2$. Now suppose the network receives a periodic query "report the average temperature every unit of time with accuracy requirement $\epsilon = 6$ and $\delta = 0.2$". In the first stage, we obtain the average data value $avg_1 = 60°\text{F}$ by querying the layer corresponding to $\epsilon_1 = 2$, and $\delta_1 = 0.1$. After one unit of time, we expect that the average temperature changes to $avg_1 + \mu = 70°\text{F}$. However, to ensure that the error of this estimate is bounded by the user-specified bound $\epsilon = 6$, the error contributed by the normal distribution must be bounded by 4 (i.e. the one-sided confidence interval $\epsilon_n = 4$). As shown in Figure 4.4, with probability 78.86%, the expected change $\mu = 10$ does not deviate from the true change by more than 4. In summary, if the error of the initial estimate avg_1 for the average is bounded by $\epsilon_1 = 2$, and the error of the estimate μ for the change is bounded by $\epsilon_n = 4$, then the error of the estimate $avg_1 + \mu$ for the average after one unit of time will be bounded by 6. Therefore, after one unit of time, with probability $0.7886 \times 0.90 = 0.7097$, i.e. confidence level 70.97%, the error of the

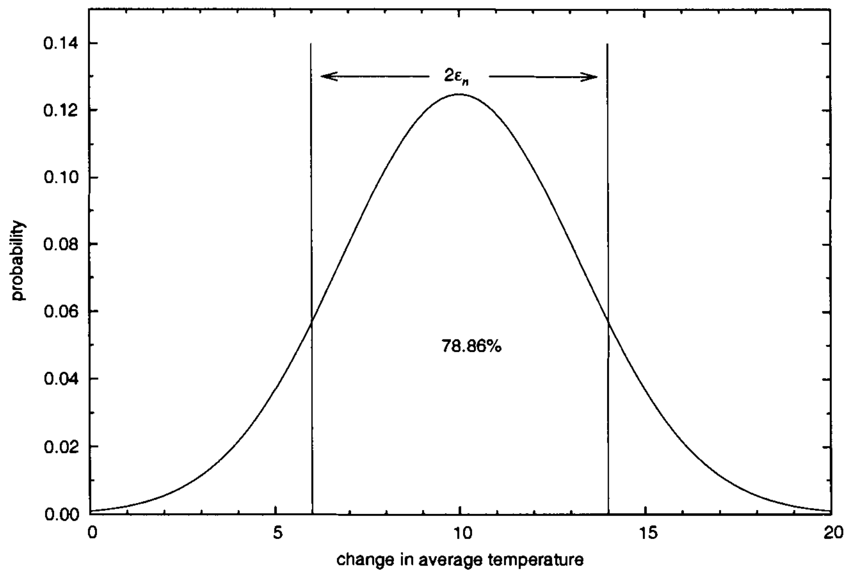

Figure 4.4: The change of average temperature in one unit of time follows a normal distribution with $\mu = 10$ and $\sigma = 3.2$. To ensure the ultimate error bound of $\epsilon = 6$, the error bound (one-sided confidence interval) $\epsilon_n = 4$.

estimate is bounded by 6. The error probability is $1 - 0.7097 = 0.2903$, which is larger than the user-specified error probability $\delta = 0.2$. To boost the confidence level to 80%, we query a few more sensor nodes with an error bound of $\epsilon_2 = 6$ and a much looser error probability of $\delta_2 = \frac{0.2}{1-0.7097} = 0.689$. If the returned value of *TestAverage* is 70°F, we return this value. Otherwise, if the returned value falls outside of 70 ± 6°F, an anomaly might have occurred [2]. In this case, we perform *QueryAverage* to determine the new average value. If the returned value falls within 70 ± 6°F, to ensure the error bound, we perform *TestAverage* with a more stringent error bound $\hat{\epsilon}$ until an anomaly is found or the average value obtained by using the initial estimate and the normal distribution is confirmed.

We are now ready to present our improved scheme for AVERAGE queries, which is composed of algorithms *QueryAverage* and T*estAverage*. First the complete list of the parameters and notations used in the algorithms are given in Table 4.1. Figure 4.5 shows Algorithm *QueryAverage*. It takes as input the *error bound* ϵ and the *error probability* δ. Here we denote by $Query(l)$ the average data value obtained by querying the sensor nodes on

[2]Here we use the word *anomaly* to indicate a situation whose likelihood is small according to the given normal distribution.

N	number of sensor nodes in the network
p	promotion probability
ϵ	user-specified error bound
δ	user-specified error probability
μ	mean of the normal distribution
σ	standard deviation of the normal distribution
ϵ_1	error bound used for *QueryAverage*
δ_1	error probability used for *QueryAverage*
avg_1	average value output by *QueryAverage*
ϵ_n	one-sided confidence interval of the normal distribution
X	random variable for the change in the average value
q_i	confidence level for the interval $[\mu - \epsilon_n,\ \mu + \epsilon_n]$ of the normal distribution
s	factor to make the confidence interval stringent
avg_i	average value output by *TestAverage*

Table 4.1: Parameter table for *QueryAverage* and *TestAverage*

```
Algorithm QueryAverage ($(\epsilon, \delta)$)

1.  Select $\epsilon_1 < \epsilon$ and $\delta_1 < \delta$.
2.  $l_1 = \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left( \frac{(b-a)^2 ln \frac{4}{\delta_1}}{2\epsilon_1^2} + ln \frac{2}{\delta_1} + \sqrt{ln \frac{2}{\delta_1} (2 \frac{(b-a)^2 ln \frac{4}{\delta_1}}{2\epsilon_1^2} + ln \frac{2}{\delta_1})} \right)$.
3.  $avg_1 = Query(l_1)$.
4.  return $avg_1$
```

Figure 4.5: Algorithm *QueryAverage*

layer l. Figure 4.6 shows the algorithm *TestAverage*. It takes as input the error bound ϵ and the error probability δ, as well as the mean μ and standard deviation σ of the distribution of the change in the average data value. The other input parameters of *TestAverage* are the round number i, and avg_1, the average obtained from *QueryAverage*.

In Line 2 of algorithm *TestAverage*, the probability that the change will fall within the interval $[\mu - \epsilon_n,\ \mu + \epsilon_n]$ is obtained by using the probability distribution of the change. From Line 1 to Line 5, *TestAverage* tests if the average obtained by combining *QueryAverage* and the normal distribution satisfies the user-specified accuracy requirement. If this is the case, *TestAverage* will not perform any further operations. This might occur when the number of sensor nodes queried in *QueryAverage* is large enough. In Lines 17 and 18, an anomaly

```
Algorithm TestAverage ($i, \epsilon, \delta, \mu, \sigma, avg_1$)
1.  $\epsilon_n = \epsilon - \epsilon_1$.
2.  $q_i = Pr[\mu - \epsilon_n < X < \mu + \epsilon_n]$.
3.  if $1 - q_i \times (1 - \delta_1) < \delta$
4.      $avg_i = avg_1 + \mu$.
5.      return $avg_i$.
6.  else
7.      $\delta_i = \frac{\delta}{1-q_i \times (1-\delta_1)}$.
8.      $s = 0$.
9.      repeat
10.         $\epsilon_i = \epsilon - s$.
11.         $l_i = \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left( \frac{(b-a)^2 ln\frac{4}{\delta_i}}{2\epsilon_i^2} + ln\frac{2}{\delta_i} + \sqrt{ln\frac{2}{\delta_i}(2\frac{(b-a)^2 ln\frac{4}{\delta_i}}{2\epsilon_i^2} + ln\frac{2}{\delta_i})} \right)$.
12.         if ($Query(l_i) \leq avg_1 + \mu + s$ and $Query(l_i) \geq avg_1 + \mu - s$)
13.             $avg_i = avg_1 + \mu$.
14.             return $avg_i$.
15.         else
16.             increase $s$.
17.             if $s \geq \epsilon$
18.                 Goto QueryAverage.
```

Figure 4.6: Algorithm *TestAverage*

is detected and thus *QueryAverage* will be performed again to obtain an initial estimate.

In what follows, we show that the above improved scheme does return an average value which meets the user-specified accuracy requirement.

Theorem 4.11. *Consider a sensor network with N sensor nodes and promotion probability p. Assume the data value collected by each sensor node is in the range $[a, b]$ and the change in the average value of the sensor nodes follows a normal distribution with mean μ and standard deviation σ. The probability that the average value returned by algorithm* TestAverage *deviates from the true average by more than ϵ is less than δ.*

Proof: First consider *QueryAverage*. Choose any ϵ_1, δ_1 such that $\epsilon_1 < \epsilon$ and $\delta_1 < \delta$. By Theorem 4.10, if we send an AVERAGE query to layer

$$l_1 < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(\frac{(b-a)^2 ln\frac{4}{\delta_1}}{2\epsilon_1^2} + ln\frac{2}{\delta_1} + \sqrt{ln\frac{2}{\delta_1}(2\frac{(b-a)^2 ln\frac{4}{\delta_1}}{2\epsilon_1^2} + ln\frac{2}{\delta_1})} \right),$$

we can obtain an average value avg_1 which deviates from the true average by less than ϵ_1 with probability greater than $1 - \delta_1$.

For *TestAverage*, let $\overline{Y}_i$ denote the average value over all the sensor nodes at round i. Define $\Delta_i = \overline{Y}_i - \overline{Y}_1$. We know *a priori* that Δ_i has the normal distribution with mean μ and standard deviation σ.

In round i, $\epsilon_n = \epsilon - \epsilon_1$ is the maximum error we allow to be contributed by the normal distribution, and q_i is the probability that the change in the average value will fall within the interval $[\mu - \epsilon_n, \mu + \epsilon_n]$.

Therefore with probability $q_i \times (1 - \delta_1)$, we can guarantee an error bound of $\epsilon_1 + \epsilon_n < \epsilon$. If $1 - q_i \times (1 - \delta_1) < \delta$, then the average value obtained using the initial estimate and the expected change satisfies both the *error bound* ϵ and the *error probability* δ. We will return this value. Otherwise, we choose $\delta_i = \frac{\delta}{1 - q_i \times (1-\delta_1)}$ which ensures that the error probability will be bounded by δ in the ith round. For error bound ϵ_i in the ith round, since we do not know the returned value, we use all ϵ_i from ϵ to 0 as long as the returned value $avg_i \pm (\epsilon_i)$ is bounded within the interval $(avg_1 + \mu) \pm \epsilon$. If that happens, the change is confirmed. Otherwise, an anomaly might have occurred and thus *QueryAverage* must be performed again. In our algorithm, we reduce ϵ_i iteratively from ϵ to 0, and use ϵ_i and δ_i to query layer $l_i < \log_{\frac{1}{p}} N - \log_{\frac{1}{p}} \left(\frac{(b-a)^2 ln\frac{4}{\delta_i}}{2\epsilon_i^2} + ln\frac{2}{\delta_i} + \sqrt{ln\frac{2}{\delta_i}(2\frac{(b-a)^2 ln\frac{4}{\delta_i}}{2\epsilon_i^2} + ln\frac{2}{\delta_i})} \right)$ so that the number of sensor nodes involved in *TestAverage* will increase gradually and stop as early as possible. $\square$

Chapter 5

Numerical Simulations

In this chapter, we use numerical simulations to evaluate the theoretical expressions derived in the previous chapter for the tradeoffs between the accuracy and the latency. In particular, we show the effects of the accuracy requirement of a query on the layer to be queried, and also observe the effects of various parameters on the performance of the layered network.

5.1 QUANTILE Queries

We now show the relationship between the latency and the accuracy for QUANTILE queries in a snapshot query setting. In our simulations, we assume the sensor network has $N = 10000$ sensor nodes. We first set the promotion probability $p = 0.7$ and explore the effects of varying p later. Figure 5.1 is a general picture of the relationship among error bound ϵ, error probability δ and the layer to be queried. Clearly, as ϵ and δ increase, which means the accuracy requirement is relaxed, the layer to be queried also increases, as confirmed by our formal analysis.

Figure 5.2 shows the relationship between δ and the layer to be queried. Given a QUANTILE query with $\epsilon = 0.13$ and $\delta = 0.05$, by Figure 5.2, we should query layer 11. For the same value of ϵ, if δ increases to 0.25, we only need to query layer 12, which contains fewer sensor nodes, resulting in lower query delay. The relationship between ϵ and the layer to be queried is demonstrated in Figure 5.3. For the same value $\delta = 0.15$, we need to query layer 8 if $\epsilon = 0.06$, while we only need to query layer 14 if $\epsilon = 0.18$. We can observe that the layer number monotonically increases with both ϵ and δ. However, the layer number increases faster with ϵ. Hence ϵ has greater impact on the layer to be queried than δ. This is

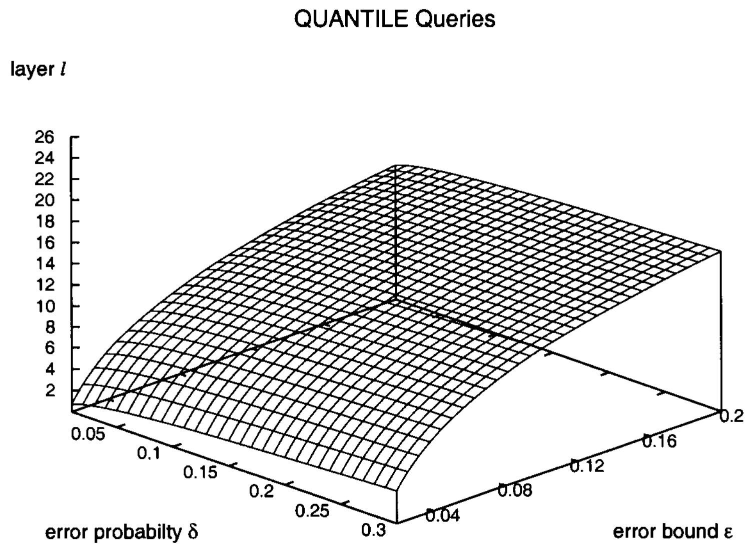

Figure 5.1: QUANTILE queries: the impact of ϵ, δ on the layer to be queried for the number of sensor nodes $N = 10000$ and promotion probability $p = 0.7$

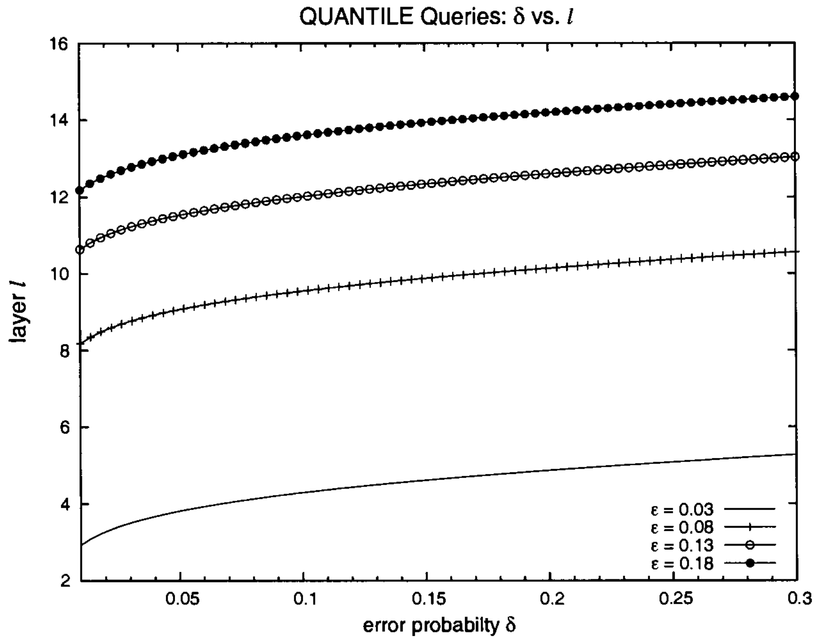

Figure 5.2: QUANTILE queries: error probability δ vs. layer l for $N = 10000$ and $p = 0.7$

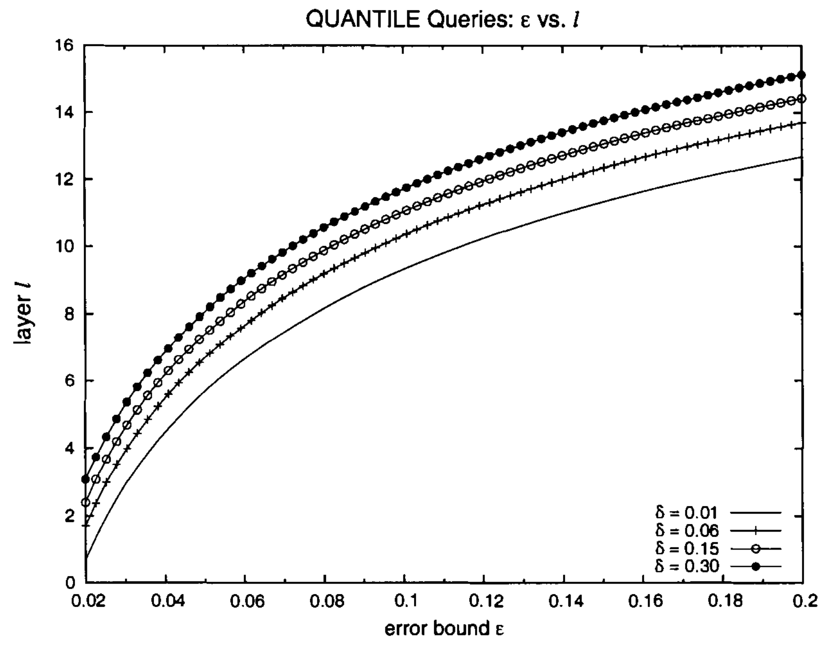

Figure 5.3: QUANTILE queries: error bound ϵ vs. layer l for $N = 10000$ and $p = 0.7$

due to the fact that the error probability δ can easily be reduced using standard techniques from probability theory and randomized algorithms. Running the algorithm $O(\log k)$ times and returning the median answer will reduce δ to δ/k, as can be shown using the Chernoff bound. On the other hand, to reduce ϵ, we need to query a lower layer, which contains more sensor nodes.

In Figure 5.4, we set the error probability $\delta = 0.2$, and observe how the layer number increases with the error bound ϵ for various values of promotion probability p. We can see, as ϵ increases from 0.02 to 0.14, the layer number increases from 1 to 6 for $p = 0.5$, while the layer number increases from 3 to 16 for $p = 0.75$. In summary, as p increases, the variation in the layer number is more obvious as we vary ϵ. This is because there are fewer layers in a network with smaller p and the choice of layers in a network with smaller p is more coarse-grained than that in a network with larger p. For a query with the same δ and ϵ, a higher layer will be queried in a network with larger p, since a layer l in a network with larger p contains more sensor nodes than a layer l in a network with smaller p.

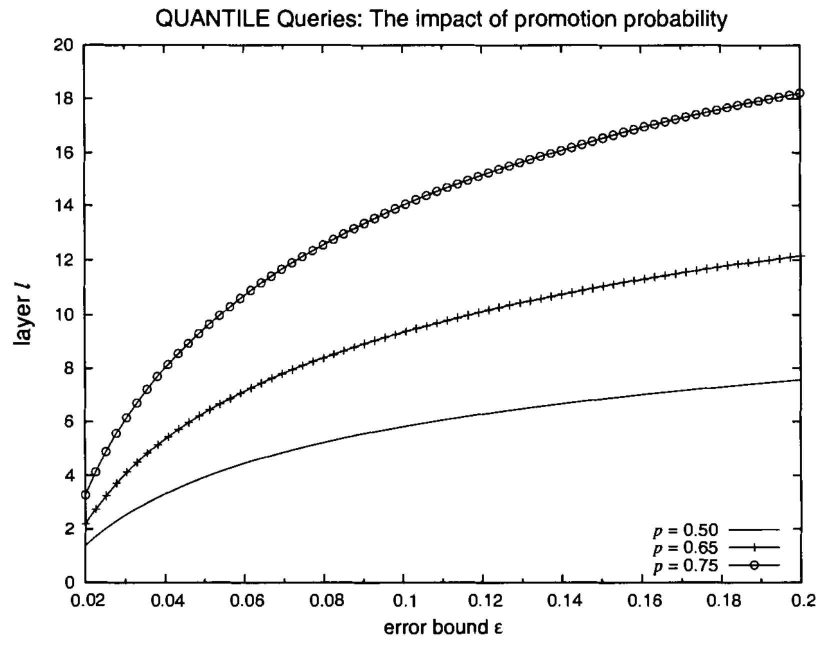

Figure 5.4: QUANTILE queries, the effect of promotion probability p for $N = 10000$ and $\delta = 0.2$

5.2 AVERAGE Queries

For AVERAGE queries, we first show the relationship between the latency and the accuracy in a snapshot query setting. We set the number of sensor nodes $N = 10000$, and the promotion probability $p = 0.7$, which is the same as those for the QUANTILE queries reported above.

The relationship between error probability δ and the layer to be queried is shown in Figure 5.5. From Figure 5.6, we can observe how the layer number increases with the error bound ϵ. As in QUANTILE queries, and for the same reason as in that case, ϵ has greater impact on the layer number than δ. This gives us a hint for improving our scheme for AVERAGE queries as we have additional statistical information.

5.3 Utilizing Statistical Information for Periodic Queries

We now investigate how to choose appropriate values for the parameters introduced in our improved scheme, which utilizes statistical information for periodic queries. Our goal is to

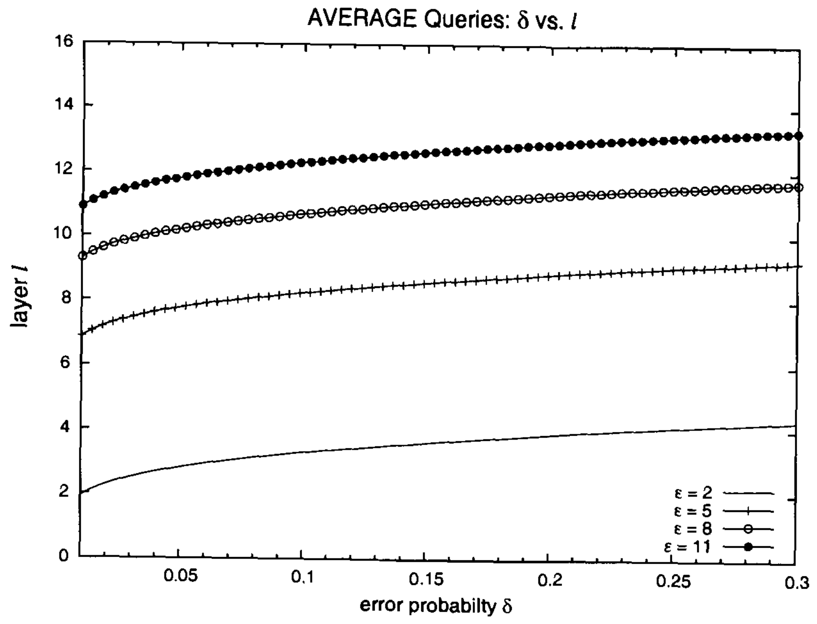

Figure 5.5: AVERAGE queries, error probability δ vs. layer l for $N = 10000$, $p = 0.7$, $a = 20$, and $b = 100$

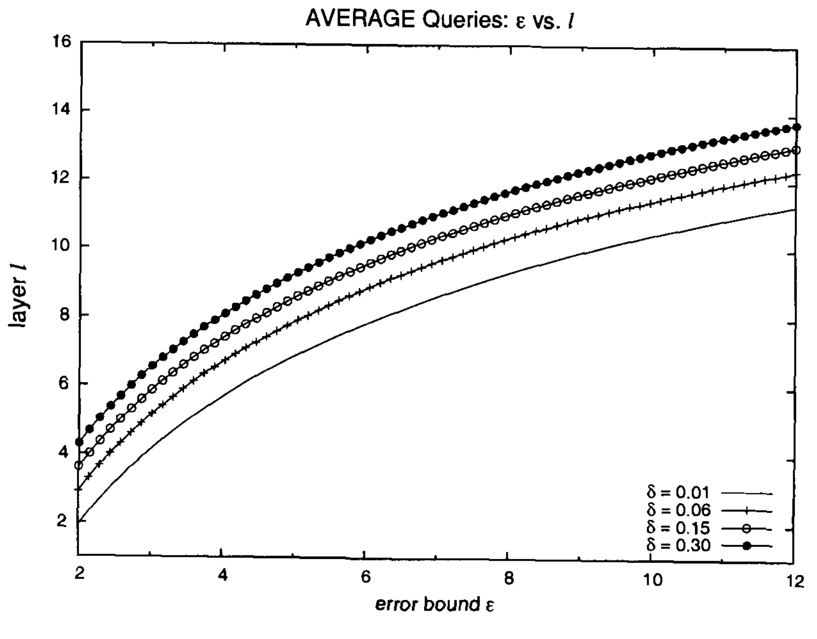

Figure 5.6: AVERAGE queries, error bound ϵ vs. layer l for $N = 10000$, $p = 0.7$, $a = 20$, and $b = 100$

find out how to split the error between *QueryAverage* and *TestAverage*, and which parameter has greater impact on the layer to be queried. Table 5.1 lists the values of the parameters we fix for the simulation. We will vary other parameters and observe the effects of these variables. A complete list of the parameters are given in Table 4.1.

N	10000	number of sensor nodes in the network
p	0.7	promotion probability
ϵ	8.1	user-specified error bound
δ	0.25	user-specified error probability
a	20	minimum of individual sensor values
b	100	maximum of individual sensor values
μ	10	mean of the normal distribution
σ	2	standard deviation of the normal distribution

Table 5.1: The values for the fixed parameters

Figure 5.7 shows the effect of ϵ_1 and δ_1 on the layer to be queried for *QueryAverage*. Figure 5.8 shows the effect of ϵ_1 and δ_1 on the layer to be queried for *TestAverage*. In our simulations, we observe that regardless of the choice of the values for ϵ_1 and δ_1, *QueryAverage* queries a larger number of sensor nodes than *TestAverage*. However, in *QueryAverage*, the layer number increases moderately as ϵ_1 increases whereas in *TestAverage* the layer number changes drastically as ϵ_1 varies. This implies a smaller value for ϵ_1 in *QueryAverage* will lead to fewer sensor nodes queried in *QueryAverage* and *TestAverage*.

We now study the effect of σ, the standard deviation of the distribution of the change in the average value. σ measures the average distance of the changes from the expected change μ. We vary the value of σ, and fix the rest of the parameters in Table 5.1. We observe from Figure 5.9 that σ has a big influence on efficiency. The improved scheme performs well when σ is small, which means in general the changes are close to the expected change. The discontinuity of the lines when $\sigma = 2$ and $\sigma = 3$ indicates that *QueryAverage* has tested more sensor nodes than required, making some of the following rounds of *TestAverage* unnecessary.

Note that, when $p = 0.5$, the expected number of sensor nodes doubles as we go down each layer. To reduce energy consumption, every query performed in *QueryAverage* on some layer $i - 1$ rather than layer i must be compensated by two or more runs of *TestAverage* performed on layer $i + 1$ rather than layer i, or four or more on layer $i + 2$ rather than layer $i + 1$, etc. Thus, the combination of *QueryAverage* and *TestAverage* is more profitable when

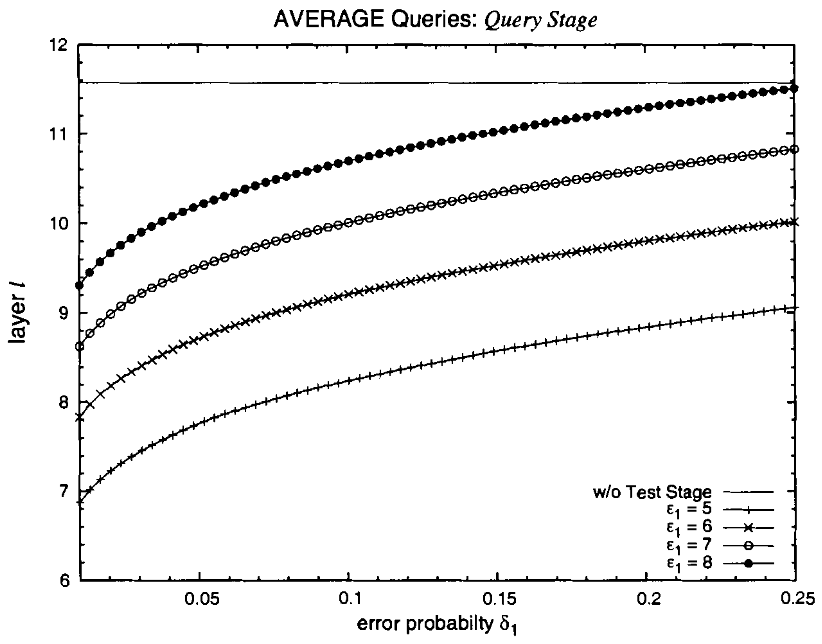

Figure 5.7: AVERAGE queries, Algorithm *QueryAverage*

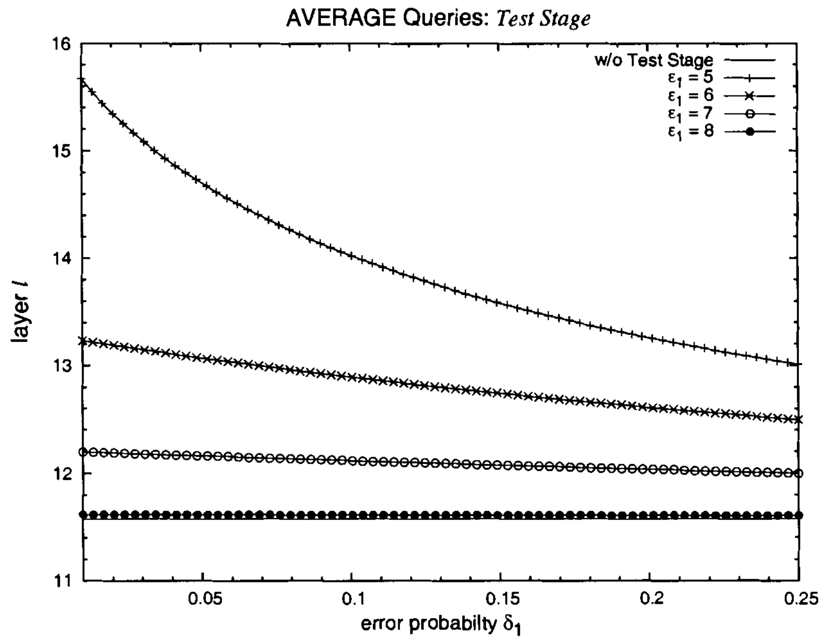

Figure 5.8: AVERAGE queries, Algorithm *TestAverage*

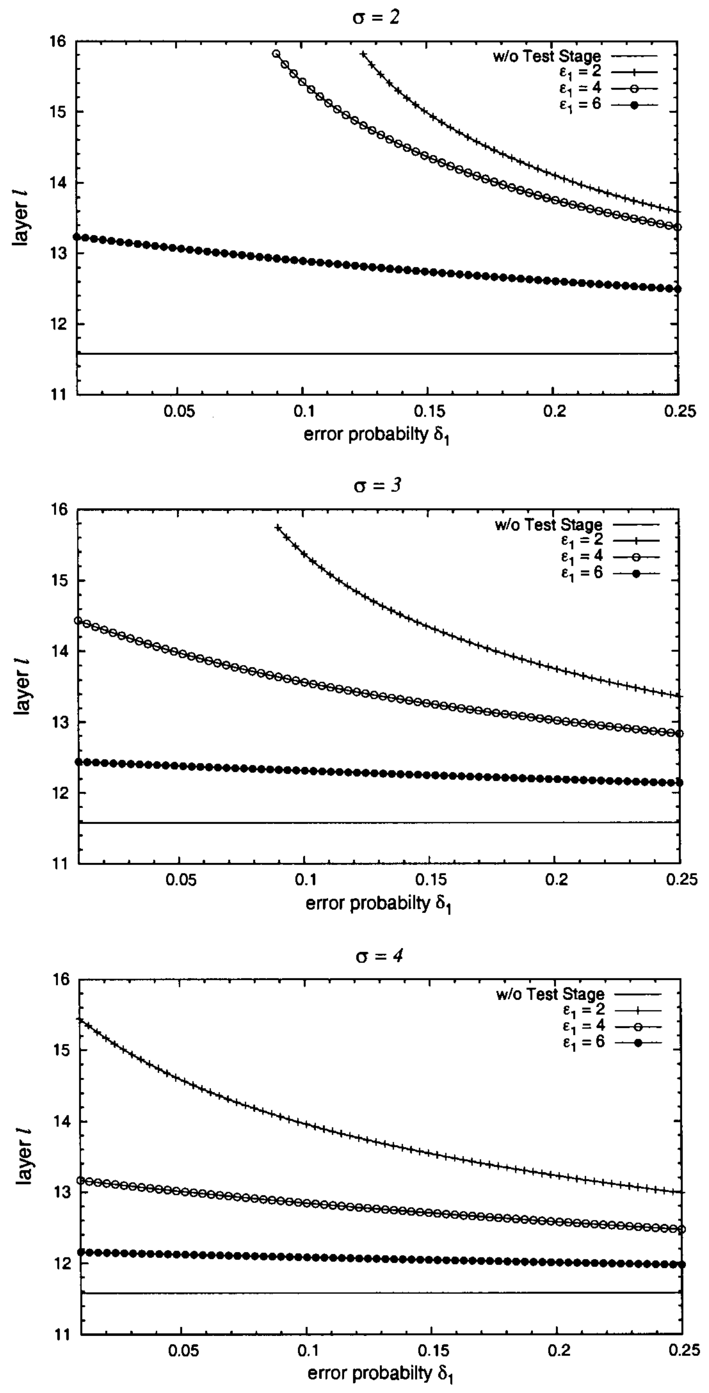

Figure 5.9: AVERAGE query: *TestAverage*, the effect of σ

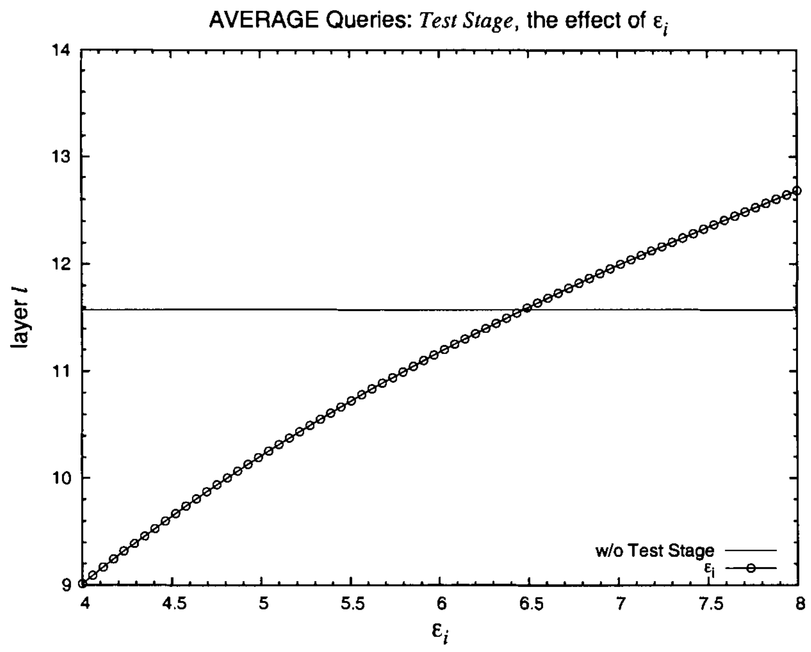

Figure 5.10: AVERAGE queries, the effect of the readings from *TestAverage*

the change in the data is highly predictable. Thus, *QueryAverage* and *TestAverage* can be used for critical monitoring applications in stable environments whereas *QueryAverage* alone can be used in applications of data acquisition in dynamic environments.

Finally, we investigate the effect of ϵ_i in *TestAverage* on the query latency of our improved scheme. Let $Query(l_i)$ denote the average value returned by layer l_i which is determined by ϵ_i and δ_i. In algorithm *TestAverage*, if $Query(l_i) \pm \epsilon_i$ is out of the range $avg_1 + \mu \pm \epsilon$, then we will decrease ϵ_i to query a lower layer. This will lead to involving a larger number of sensor nodes for this query. Algorithm *TestAverage* stops at $\epsilon_i = 0$. However, in our simulation, we observe that when ϵ_i decreases to some value, stopping *TestAverage* and moving to *QueryAverage* will lead to lower query latency. We observe this effect in Figure 5.10 where the same data values as those in Table 5.1 are used, setting $\epsilon_1 = 6$ and $\delta_1 = 15\%$, which are reasonable choices. In Figure 5.10 we see that when $\epsilon_i > 6.5$, using *TestAverage* will involve fewer sensor nodes. However, when $\epsilon_i < 6.5$, moving to *QueryAverage* will lead to fewer sensor nodes being queried.

Chapter 6

Energy Consumption Issues

In this chapter, we discuss the energy consumption issues of the randomized layered network and estimate the overall lifetime of the network.

6.1 Energy Consumption

In our randomized layered network, higher layer sensor nodes transmit over longer ranges than their lower layer counterparts. The energy consumed for a transmission is proportional to r^{α}, where r is the transmission range, and α varies in the interval [2, 4] depending on the communication medium. Given that any sensor node on a layer is also present on all the layers below this layer, if the sensor nodes are fixed on certain layers after the layered network is constructed, the sensor nodes on higher layers will run out of energy much faster than those on lower layers. To balance out the energy consumption among all the sensor nodes in the network, we need to reconstruct the layered network periodically by reassigning each node's layer, so that the sensor nodes on the high layers change over time.

An appropriate timing scheme for the reconstructions will lead to relatively uniform energy consumption across the sensor nodes in the network. Note that reconstruction has no expected effect on accuracy, since we are as likely to be stuck with a "good" sample of sensor nodes (in which case reconstruction is likely to give us a worse sample) as with a "bad" one. This argument is demonstrated in Figure 6.1. Here, we denote the event that a construction obtains a good sample by "1", and denote the event that a construction obtains a bad sample by "0". In Figure 6.1, the network is constructed twice. The second construction is beneficial only for the case "01" in which we switch from a "bad" sample

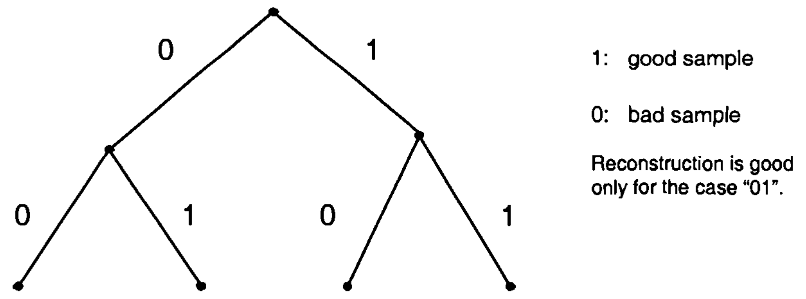

Figure 6.1: The effect of reconstruction on the quality of the sample

to a "good" sample. Given the above and the overhead of building a new aggregation tree for each new construction, it is not energy-efficient to reconstruct the network too frequently. However, the reconstruction time that maximizes the network lifetime depends on the energy consumption for building the specific new aggregation tree. The goal of our randomized scheme is to reduce the number of sensor nodes, hence the query latency. Once the layer to be used is determined, the particular algorithm for building the aggregation tree is transparent to our scheme. Hence the optimal reconstruction time is beyond the scope of this thesis. Nevertheless, we will use simulations to provide a rough picture of the effect of reconstructions on the lifetime of the network in Section 6.3, without taking into account the overhead of building new aggregation trees.

6.2 Overall Lifetime of the Layered Network

In this section, we analyze the lifetime of the layered network assuming that each sensor node has sufficient energy to let it undergo many reconstructions, and that we run sufficiently many reconstructions. Ideally, we have a totally symmetric scenario where the service that each sensor node has performed on each layer is identical across sensor nodes. Since the queried layers are chosen independently by users in a random fashion, given a large enough number of reconstructions, what one expects to see is that most sensor nodes will have served on most layers.

We define the lifetime of the network to be the expected lifetime of an arbitrary sensor node. The energy consumed by a sensor node for a query directly depends on the distance between the sensor node and its neighbors on the chosen layer. Since the expected number of sensor nodes on layer l is $N \cdot p^l$, the transmission range on layer l is set to be $r(l) = \frac{D}{\sqrt{N \cdot p^l}}$.

The energy spent by a sensor node for a query on layer l is $e(l) = \left(\frac{D}{\sqrt{N \cdot p^l}}\right)^\alpha$, which is what we will use below to estimate the overall lifetime of the layered network.

Under the assumption that the queries are uniformly distributed across all of the layers due to the error bounds and the error probabilities coming independently from the users, the following theorem estimates the expected lifetime of the layered network.

Theorem 6.1. *Consider a sensor network with N sensor nodes, where the initial battery power of each sensor node is B, and the number of incoming queries is a Poisson process with rate λ. In a setting where each layer is equally likely to be queried, the* expected lifetime *of the layered network is* $E[t] = \frac{BL(\sqrt{N})^\alpha(1-p^{1-\alpha/2})}{\lambda D^\alpha(1-p^{L(1-\alpha/2)})}$

Proof: First consider the expected energy consumption of a sensor node for each query. For $l = 0, \ldots, L-1$, when a query occurs, a sensor node consumes energy only if the query is addressed to layer l and the sensor node exists on layer l. Since each layer has the same probability $\frac{1}{L}$ of being queried, and the probability that a sensor node exists on layer l is p^l, the energy consumption of a sensor node for each query is $e(l)$ with probability $\frac{p^l}{L}$. Therefore the expected energy consumption of a sensor node for each query is $\sum_{l=0}^{L-1} \frac{p^l}{L} e(l)$. Let the life expectancy of a sensor node be t. Then $\sum_{l=0}^{L-1} \frac{p^l}{L} e(l) \lambda E[t] = B$. Since $e(l) = \left(\frac{D}{\sqrt{N \cdot p^l}}\right)^\alpha$, the expected lifetime of the layered network is

$$E[t] = \frac{BL(\sqrt{N})^\alpha(1-p^{1-\alpha/2})}{\lambda D^\alpha(1-p^{L(1-\alpha/2)})} \qquad \square$$

6.3 The Effect of Reconstruction on the Lifetime of the Layered Network

In this section we use simulations to show the effect of reconstruction on the lifetime of the network. We assume that the sensor network occupies an area of 100 $\times$ 100, where the sensor nodes are uniformly distributed, and each sensor node has 5000 units of energy. We also assume that the queries are generated according to a Poisson distribution with mean value $\lambda = 20$. The type of aggregation queries are generated uniformly from the set {MAX, MIN, QUANTILE, AVERAGE}. For QUANTILE queries, the *error bound* and the *error probability* are assumed to be uniformly generated at random within the intervals $0 < \epsilon < 0.5$ and $0 < \delta < 0.5$ respectively. For AVERAGE queries, the range $[a, b]$ of the sensor values is set to be $[20, 100]$, and ϵ is set to be proportional to a and b. In the energy

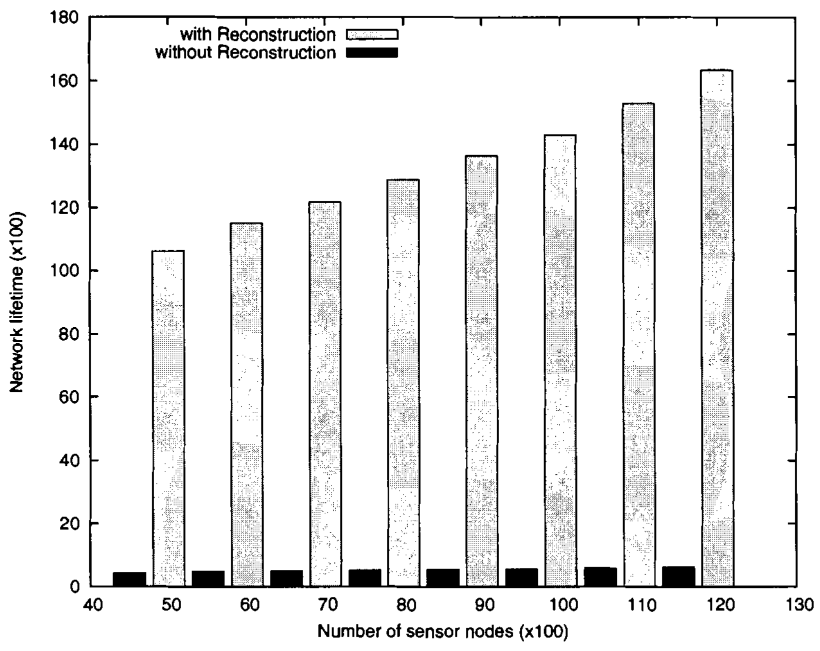

Figure 6.2: The effect of reconstruction on the lifetime of the network

consumption calculations, we use $\alpha = 2$, where α is the path-loss exponent experienced by radio transmission. The *promotion probability* is set to be $p = 0.5$. We do not consider the overhead of building a new aggregation tree for each network construction.

In our simulations, we use the simplest routing tree building technique, *naive* tree building [24]. Once the layer to be queried is determined, the base station and all the intermediate nodes on that layer broadcast the query using the transmission range of that layer and collect data in a reverse fashion. Every data point presented in the figure is the average of 100 random experiments. Figure 6.2 shows the effect of reconstruction on the lifetime of the network. Clearly, without reconstruction, the sensor network depletes much faster. As the number of sensor nodes increases, the lifetime of the network also increases, since the distances between the sensor nodes are shorter, leading to less energy consumption.

Chapter 7

Conclusion and Future Work

We have presented a fast data collection technique for approximate aggregation queries over large wireless sensor networks, which utilizes a randomized layered architecture. The layered architecture of our network is constructed and maintained locally and thus provides scalability and robustness for sensor networks where communication dominates energy consumption and communication links are unreliable.

In our layered network, each layer contains a random subset of the sensor nodes. A query is always addressed to a specific layer, and only the sensor nodes on this layer respond to the query. The data collected by this layer will be used to approximate the data collected by the whole network. This results in fewer sensor nodes being queried and thus lower query latency.

In our approximate aggregation scheme, the accuracy requirement of a query determines to which layer this query is addressed, hence the query latency. We perform formal analysis on the relationship between the accuracy of an answer and the latency of obtaining this answer. Given a query with an accuracy requirement, our scheme selects the appropriate layer which, with the lowest delay, returns an answer with the desired accuracy.

We present a scheme that permits applications to choose desired tradeoffs between accuracy and latency. Our scheme is useful in the context where users are interested in obtaining aggregated information with low latency, and are satisfied with a good estimation of it.

Many environmental quantities measured by sensor nodes vary only slowly over time. We exploit this advantageous feature to efficiently collect data from sensor networks. Given statistical information obtained from the history of the environment, we improve our scheme to further reduce the latency incurred in answering AVERAGE queries in a periodic query setting.

We have made our first attempt to attack the delay problem in sensor networks. There are still various open challenges and directions for future work:

- We would like to explore data aggregation techniques for other types of approximate queries over sensor networks such as majority, the most frequent item, and range queries.
- We want to investigate the optimal reconstruction time of the layered network. To prolong the lifetime of the network, we need to reconstruct the layered network periodically. Since the optimal reconstruction time is aggregation-tree specific, we have not addressed this issue. It would be interesting to study the optimal reconstruction time under various schemes for building aggregation trees.
- We want to develop a reconstruction scheme based on the residual energy of the sensor nodes. In terms of balancing the energy consumption among all the sensor nodes, a better way to reconstruct the layered network would be relating the promotion probability of a sensor node to its residual energy, so that the sensor nodes with larger residual energy will have better chances to exist on high layers.
- When we refine our scheme to utilize statistical information for periodic queries, we make the assumption that the change in the environmental quantity has the normal distribution. In general, the statistics of the environmental quantity being monitored might not be known, or its distribution might not follow the normal distribution. It would be interesting to investigate the problem in a more general setting.

The enormous amounts of data generated by large sensor networks necessitate the development of efficient data collection techniques to extract general characteristics of the data from sensor networks. Due to the constantly changing environments of sensor networks and the large number of sensor nodes involved, approximate and randomized computation over sensor networks is a promising approach. We envision that approximate and randomized computation will become very common in many sensor network applications.

Appendix A

The Chernoff and Hoeffding Bounds

In the design and analysis of randomized algorithms, it is essential to show that the algorithms behave well with high probability. The Chernoff and Hoeffding bounds are fundamental tools for analyses of this type. The probability that a random variable deviates from its expectation by a given amount is often called the *tail probability* of the random variable. The Chernoff and Hoeffding bounds are used for bounding the tail probabilities of the sums of independent random variables. The Hoeffding bound can be extended to the case of certain sums of dependent random variables.

A.1 The Chernoff Bound

In our analysis in Chapter 4, we require the following inequality [35], a simplified version of the Chernoff bound. This inequality bounds the lower tail probability of the sums of independent indicator random variables. A general version of the Chernoff bound can be found in [8].

Lemma A.1. (Chernoff's Inequality) *Let $X_1, X_2, \ldots, X_n$ be independent random variables such that, for $1 \le i \le n$, $X_i \in \{0,1\}$, $Pr[X_i = 1] = p_i$ and $Pr[X_i = 0] = 1 - p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = E[X] = \sum_{i=1}^{n} p_i$ and $0 < \epsilon \le 1$,*

$$Pr\left[X < (1-\epsilon)\mu\right] < e^{\frac{-\epsilon^2\mu}{2}}.$$

Random variables such as $X_1, \dots, X_n$ in Lemma A.1, whose values are in the range $\{0, 1\}$ are often referred to as *indicator random variables.* The sum of independent indicator random variables is said to have the *binomial distribution.*

A.2 The Hoeffding Bound

The following inequality, due to Hoeffding [18], applies to a more general class of random variables. It bounds the tail probabilities of the sums of independent random variables whose ranges are bounded.

Lemma A.2. (Hoeffding's Inequality) *Let $X_1, X_2, \dots, X_n$ be independent random variables such that, for $1 \le i \le n$, $a \le X_i \le b$. Then, for $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\epsilon > 0$,*

$$Pr\left[\overline{X} - E[\overline{X}] \ge \epsilon\right] \le e^{\frac{-2n\epsilon^2}{(b-a)^2}}.$$

Random variables $X_1, X_2, \dots, X_n$ in Lemma A.2 can be considered as n random samples chosen with replacement from the sample space $[a, b]$.

In section 6 of the paper [18], Hoeffding showed that the same bound as described in lemma A.2 applies to the tail probability of the sum of random samples without replacement from a finite population. This result is summarized in the following lemma.

Lemma A.3. *Let $C = \{c_1, c_2, \dots, c_N\}$, where $a \le c_i \le b$ for $1 \le i \le N$. Let $X_1, X_2, \dots, X_n$ be random samples chosen from C without replacement. Then, for $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\epsilon > 0$,*

$$Pr\left[\overline{X} - E[\overline{X}] \ge \epsilon\right] \le e^{\frac{-2n\epsilon^2}{(b-a)^2}}.$$

Bibliography

[1] K. Akkaya, M. Younis, and M. Youssef. Efficient aggregation of delay-constrained data in wireless sensor networks. In *Proceedings of the Workshop on Internet Compatible QoS in Ad Hoc Wireless Networks 2005*, 2005.

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

[3] I. F. Akyildiz, M. C. Vuran, and O. B. Akan. On exploiting spatial and temporal correlation in wireless sensor networks. In *Proceedings of Second Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2004)*, pages 71–80, 2004.

[4] P. Bonnet, J. E. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, 7(5):10–15, 2000.

[5] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2004)*, pages 157–164, 2004.

[6] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann. Scalable coordination for wireless sensor networks: self-configuring localization systems. In *Proceedings of the 6th International Symposium on Communication Theory and Applications (ISCTA 2001)*, Ambleside, UK, July 2001.

[7] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *Proceedings of INFOCOM 2005*, pages 1747–1757, 2005.

[8] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[9] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *Proceedings of IEEE INFOCOM 2004*, pages 2571 – 2582, 2004.

[10] R. Cristescu and M. Vetterli. On the optimal density for real-time data gathering of spatio-temporal processes in sensor networks. In *Proceedings of the 4th International*

Symposium on Information Processing in Sensor Networks (IPSN 2005), pages 159–164, 2005.

[11] M. Enachescu, A. Goel, R. Govindan, and R. Motwani. Scale free aggregation in sensor networks. In *Proceedings of 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004)*, pages 71–84, 2004.

[12] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1999)*, pages 263–270, 1999.

[13] S. Goel, T. Imielinski, K. Ozbay, and B. Nath. Poster abstract: sensors on wheels – towards a zero-infrastructure solution for intelligent transportation systems. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pages 338–339, 2003.

[14] M. B. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2004)*, pages 275–285, 2004.

[15] J. Heidemann, F. Silva, and D. Estrin. Matching data dissemination algorithms to application requirements. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pages 218–229, 2003.

[16] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 2001)*, pages 146–159, 2001.

[17] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS 2000)*, page 8020, 2000.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[19] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002)*, pages 457–458, 2002.

[20] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 56–67, 2000.

[21] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for "smart dust". In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1999)*, pages 271–278, 1999.

[22] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW 2002)*, pages 575–578, 2002.

[23] B. Krishnamachari, D. Estrin, and S. B. Wicker. Modelling data centric routing in wireless sensor networks. Technical Report CENG 02-14, Computer Engineering, University of Southern California, 2002.

[24] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran. Dfuse: a framework for distributed data fusion. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pages 114–125, 2003.

[25] S. S. Kunniyur and S. Narasimhan. Modelling the effect of network parameters on delay in wireless ad-hoc networks. In *Proceedings of the 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)*, pages 340–349, 2005.

[26] S. Lindsey and C. S. Raghavendra. Pegasis: power-efficient gathering in sensor information systems. In *Proceedings of 2002 IEEE Aerospace Conference*, pages 1125–1130, 2002.

[27] K. Lorincz, D. J. Malan, T. R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004.

[28] J. D. Lundquist, D. R. Cayan, and M. D. Dettinger. Meteorology and hydrology in yosemite national park: a sensor network application. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN 2003)*, pages 518–528, 2003.

[29] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI 2002)*, pages 131–146, 2002.

[30] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, pages 49–58, 2002.

[31] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, pages 88–97, 2002.

[32] A. Manjeshwar and D. P. Agarwal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS 2001)*, pages 2009–2015, 2001.

[33] A. Manjeshwar and D. P. Agarwal. Apteen: a hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, pages 195–202, 2002.

[34] T. M. Mitchell. *Machine Learning.* McGraw Hill, New York, 1997.

[35] R. Motwani and P. Raghawan. *Randomized Algorithms.* Cambridge University Press, Cambrige, United Kingdom, 1995.

[36] S. Narasimhan and S. S. Kunniyur. Delay differentiation in sensor networks using power control. In *Proceedings of the 39th Annual Conference on Information Sciences and Systems (CISS 2004)*, pages 1390–1395, 2004.

[37] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004)*, pages 28–35, 2004.

[38] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[39] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004)*, pages 239–249, 2004.

[40] M. Srivastava, R. Muntz, and M. Potkonjak. Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving enviroments. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001)*, pages 132–138, 2001.

[41] VENUS, Victoria experimental network under the sea. Available from `http://www.venus.uvic.ca`.

[42] P. von Rickenbach and R. Wattenhofer. Gathering correlated data in sensor networks. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing (DIALM-POMC 2004)*, pages 60–66, 2004.

[43] D. Wang, Y. Long, and F. Ergun. A layered architecture for delay sensitive sensor networks. In *Proceedings of 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)*, pages 24–34, 2005.

[44] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of IEEE INFOCOM 2000*, pages 585–594, 2000.

[45] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004)*, pages 13–24, 2004.

[46] T. Yamane. *Elementary Sampling Theory.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967.

[47] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record*, 31(3):9–18, 2002.

[48] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, 2003.

[49] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach. In *Proceedings of IEEE INFOCOM 2004*, pages 629–640, 2004.

[50] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *Proceedings of IEEE INFOCOM 2004*, pages 244–255, 2004.