2007

Data delivery in fragmented wireless sensor networks using mobile agents

Hisham M. Almasaeid
Iowa State University

Data delivery in fragmented wireless sensor networks using mobile agents

by

Hisham M. Almasaeid

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Ahmed E. Kamal, Major Professor
Daji Qiao
Ying Cai

Iowa State University

Ames, Iowa

2007

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to those who helped me throughout this research. First and foremost, Dr. Ahmed E. Kamal, my supervisor, for his support, guidance, and inspiration throughout the course of this study. His course in performance evaluation was so valuable and fundamental to conduct this research.
I would also like to thank my committee members, Dr. Daji Qiao and Dr. Cai Ying, for their valuable time and advice. My very special thanks go to Dr. Daji Qiao for his inspiring course in wireless sensor networks.
Finally, I would like to thank all my family members and friends for their love and support.

ABSTRACT

In the past few years, research in Wireless Sensor Networks (WSN) has grown at an unprecented rate. This is due to the large number of potential applications and environments WSNs can be used in. Nodes in WSNs communicate in multihop fashion to deliver the sensory information to a central processing unit, such as a base station or a sink node. This form of communication requires a degree of network connectivity which might not be always achievable, either due to the sensor deployment strategy, or due to sensor node failure, which can be malicious, or otherwise. In this thesis, we study the problem of data delivery in disconnected WSNs. A special class of disconnected sensor networks called "Fragmented wireless sensor networks (FWSN)" is considered. A FWSN consists of several groups of connected sensor nodes that we call "fragments". We propose a mobility based approach that exploits resource rich, in terms of power and buffer size, mobile agents that move in the network and operate as data relays between fragments to eventually deliver data to the base station. The movement of the mobile nodes and their role as relay stations is modeled using a closed queueing network approach, which is used to obtain steady state results. Building on these results, we derive the distributions of the fragment-to-fragment and fragment-to-sink delays. The results show that this model accurately captures the system behavior. Using the same model, the effect of the movement policy, the number and speed of mobile relays, and the service time at each fragment on the end-to-end delay has also been studied. The proposed queueing model can also be used to model other roles of the mobile nodes, including their roles as either data collectors or data sinks. We also study some practical

issues, including mobility control in large networks and engineering the service time, i.e., the time that an MR spend in relaying data between fragments.

CHAPTER 1 INTRODUCTION

1.1 Sensors and Sensor Networks

Recent advances in wireless communication technologies have enabled the development of small, low-cost and low-power multi-functional sensor nodes that are able to sense the environment, process the data and communicate with each other in short range. A sensor node consists, as shown in Figure 1.1, of five basic units: processing, memory, sensing, power, and communication units. A processing unit is responsible for executing the set of routines that form a sensor's task. A memory unit consists of three parts. First, the *program flash memory* that is used by the processing unit as a temporary storage area to execute routines. Second, *measurement flash memory* which is used to store sensory measurements obtained by the sensing unit. Third, *configuration EEPROM* where all configuration data for a sensor node is kept.
A sensing unit includes different kinds of sensors, like temperature, light, and humidity, depending on the application. The most important component is the power unit which supplies all other units with the required power to operate. In addition to regular batteries, a power unit might be sustained by solar cells [1]. A communication unit connects a sensor node with its neighbors via radio communication.
In addition to all the basic units described earlier, a sensor node might include some application specific units, like localization and mobility units. A localization unit helps the sensor node estimate its geographical position with certain level of accuracy. Some applications might require sensor nodes to be mobile in order to accomplish the assigned

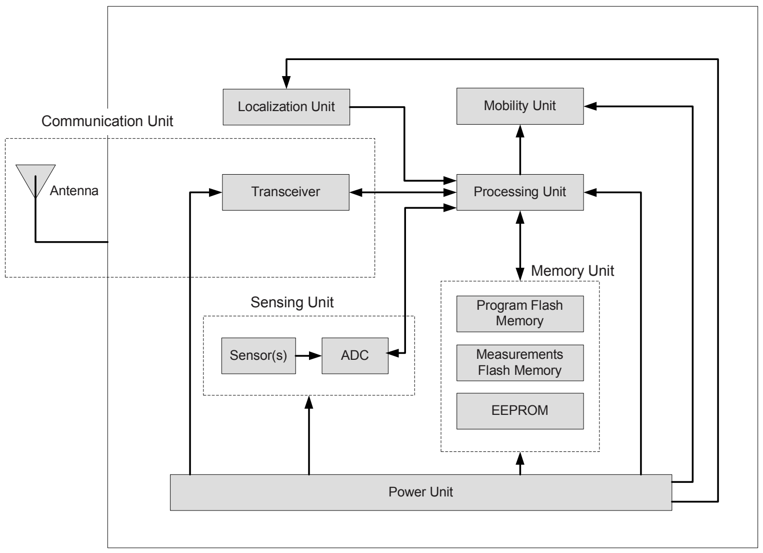


Figure 1.1: The general architecture of a sensor node.

mission. In this case, sensor nodes will be equipped with a mobility unit (mobilizer). MICA2, MICAz, Imote2, and TelosB are samples of commercial sensor node platforms developed by Crossbow [2].

A sensor network consists of a large number of those tiny sensors deployed randomly in an area of interest. Nodes in a sensor network communicate in multihop fashion to deliver the collected data to a central processing unit called the base station or the sink node. Networking of sensor nodes has been a challenging area of research due to its special characteristics. The limited sensor's power, for instance, requires all protocols designed for sensor networks to be energy efficient in order to prolong the network lifetime. The random deployment, however, poses another constraint on sensor networks to be self-organized. Moreover, the small memory size and limited power restrict the computational capabilities of sensor nodes as well as the storage capabilities. All the above mentioned limitations along with the fact that sensors are susceptible to failures led to a vast extent of research to operate the sensor network efficiently.

On the other hand, sensor networks have some unique characteristics that make it so attractive to use in many applications. In addition to the cooperative operation of sensor nodes [3, 4], those characteristics also include the dynamic availability of data sources as well as the dense random deployment of sensor nodes. Sensor network applications include, but not limited to, habitat monitoring [5], health monitoring [6], surveillance and target tracking [7, 8, 9], and intelligent transportation systems [10]. Many research problems associated with those applications have received a lot of attention. Security and privacy [11, 12], coverage and connectivity [13], and data routing [14] are samples of those problems.

1.2 Mobility in Sensor Networks

Several frameworks for sensor node mobility have recently been implemented. The basic building block of those frameworks is the mobilizer unit. A sensor node equipped with a mobilizer unit is called a mobile sensor node. However, a mobile node might not contain a sensor unit and in that case the term *"mobile sensor node"* is replaced with the more general term *"mobile agent"*. CotsBots is a mobile sensor node platform that operates for ~ 1 hour on four AAA batteries at a maximum speed of $1.2m/s$ [15]. MICAbot is another mobile sensor node platform which operates for 3.5 hours on two 1.2V AA batteries at a maximum speed of $0.3m/s$ [16]. On the other hand, some commercial and application specific mobile agents might operate at much higher speeds. iRobot PackBot Scout is an example of this category. It is designed for military operations and it operates at a maximum speed of about $3.89m/s$. Figure 1.2 shows pictures of CotBots, MICAbot, and iRobot PackBot Scout.

Mobile sensor nodes are useful for applications that rely on cooperative sensing, like target tracking, as well as applications in which data sources change dynamically. Mobile agents are used when mobility is not utilized for sensing operations, but rather for communication-based operations, like data relay and collection, as well as physical operations, like replacement of defected sensor nodes. Therefore, *"Mobile agents"* is a very wide term that might include any mobile entity like vehicles, humans, animals, and mobile robots. Exploiting mobility to improve the network performance has recently become an important area of research in sensor networks [17, 18, 19]. Next, we review the literature for some of the applications in which mobility was utilized.

1.2.1 Monitoring and Target Detection

In surveillance and target tracking applications, sensor nodes operate collaboratively to monitor (cover) specific area and detect intruders. Such applications impose strict constraints on the detection time. In stationary sensor networks, an intruder that is currently undetected will never be detected if it just keeps its position. Sensor node mobility has been introduced as a solution to this problem in [20] because when sensor nodes move, the probability of detecting the intruder increases. Both cases, when the target is stationary and mobile, have been considered. Moreover, the authors proved through mathematical analysis that the optimal sensor movement strategy is that each sensor chooses its movement uniformly in all directions. They also showed that the optimal policy for an intruder in order to maximize its detection time is to remain stationary. This thesis also investigated how node mobility can compensate for the lack of sensors in the network. That is possible when continuous coverage is not required. Instead, mobile sensor nodes keep moving in the network and consequently cover the

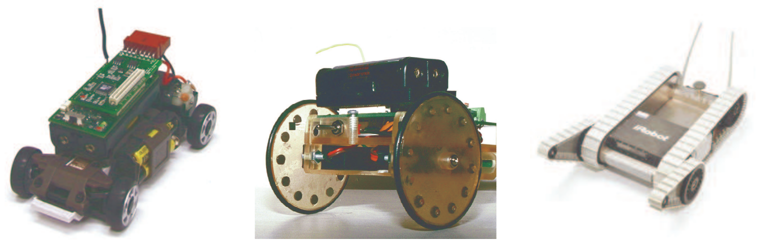

(a) CotsBots platform (b) MICAbot platform (c) iRobot PackBot Scout

Figure 1.2: Sample mobile node platforms.

area over a period of time. A Markov chain model was presented in [21] to evaluate the target detection delay based on a collaborative sensing approach with uncoordinated sensor mobility scheme. The results show that the worst-case detection latency in a mobile network is much shorter than that of a static network.

The problem of event capturing using mobile sensor nodes was studied in [22]. Events, which belong to a certain phenomenon of interest, arrive to the network field and vanish according to arrival and departure distributions. The authors studied the effect of controlled mobility as well as the dynamics of the phenomenon being covered on the fraction of events captured. The analytical results provide a guideline for choosing the number and speed of mobile sensor nodes to guarantee capturing a certain fraction of events.

1.2.2 Data Collectors

Mobility has been exploited by many researchers for data gathering in sensor networks. Some schemes rely on existing mobility in the environment, like vehicles or animals present in the network field [23, 24], whereas some others suggest supporting the system with mobile elements that have better buffer and energy capabilities than ordinary static sensor nodes and are able to communicate over longer distances [25, 26].

The concept of mobile data collectors was first introduced in [23] to connect sparse sensor networks. In that proposal, mobile data collectors, referred to as *data MULEs*, move randomly and collect the data from reachable sensors. Then, data MULEs unload the carried data as they get close enough to a base station. A three-tier architecture was presented to describe the system; a top-tier of base stations (or access points are referred to by the authors), a middle-tier of mobile agents (i.e. data MULES), and a bottom-tier of stationary sensor nodes. The problem was modeled as a Markov Chain with the assumption that data MULEs follow a two-dimensional random walk model. The model was used to predict the inter-arrival time at a sensor node, the return time to an access point, and data success rate (the ratio between the generated and delivered data in time t). In [27], multiple data MULEs with controlled mobility have been used for large scale networks. A load balancing algorithm was proposed to balance the number of sensor nodes that each data MULE serves.

The idea of sink mobility [26, 28] has been proposed as a method for data collection that prolongs the network lifetime by reducing the energy spent by static sensor nodes to relay traffic. Mobility of the base station poses several challenges regarding how the data should be routed [29] and what the optimal movement strategy is [30, 31]. In [29], a routing protocol (called MobiRoute) was proposed to route data to a mobile sink with a discrete mobility pattern. Contrary to continuous movement, discrete mobility implies a move-and-sojourn pattern. The sink's movement trajectory is composed of several *anchor points* where the mobile sink sojourns. The authors assume that the sojourn time at an *anchor point* is much longer than the movement time between *anchors*. Results showed that using the *MobiRoute* protocol with a discrete sink mobility can improve the network lifetime with a modest degradation on the packet delivery ratio.

When and where to relocate the sink node are issues that have been addressed in [31]. A heuristic search that tracks the density of the traffic going through the sink's neighboring nodes (i.e. sensor nodes that are in direct communication with the sink) was proposed to answer those questions. If the distance between the sink and any set of its neighboring nodes is larger than a threshold value, it computes the transmission power times the traffic density of that set to measure the impact of the sink's decision not to change its location. If the impact is large enough, the sink node considers relocating to a new position. The potential position with the maximum gain, in terms of saved energy, is selected as the next location. Performance results show that the mobile sink approach outperforms the stationary sink approach in terms of average energy and delay per packet. Moreover, the average lifetime of a sensor node is increased by about 33%.

1.2.3 Data Relays

The use of resource rich mobile nodes, referred to as mobile relays or routers, that keep moving in a network, in both sparse and dense deployments, to relay data between stationary sensor nodes has been recently utilized to prolong the network lifetime [32, 33] and enhance the data delivery process [34].

In [32], the performance of a large dense network with only one mobile relay has been evaluated by simulation. Results show that the improvement in the network lifetime with a mobile relay over an all-static network is upper bounded by a factor of four. Moreover, it has been shown through analysis that the mobile relay does not need to go beyond a two hop radius of the sink. This result is intuitive in the sense that the sensor nodes that are close to the sink relay more traffic and consequently consume much power.

Mobile relays have been referred to as *Message Ferries (MF)* by other researchers. In [34], a single message ferry with deterministic movement has been exploited to efficiently deliver data in sparse mobile ad hoc networks. Two variations of the MF mobility were presented based on who initiates the movement, an ad hoc node or a message ferry. In the *node-initiated* movement scheme MFs move around the network field according to predetermined routes. Those routes are known to the ad hoc nodes who move close to them to communicate with the ferry. In the *ferry-initiated* scheme however, ferries move proactively to meet ad hoc nodes.

1.3 Fragmented Wireless Sensor Network

In this thesis we are interested in the data delivery problem in disconnected networks. Most of the current research that rely on mobility to overcome network disconnection attributes this problem to the sparse nature of the network. Sparseness has been assumed to be either due to the lack of sensors at the deployment phase or due to node failures. In networks that do not tolerate failures, there are two possible ways to cope with node failures: first, to redeploy sensors wherein there are dead ones until the whole network is connected again [35], and second, to have a guaranteed level of fault tolerance at the initial deployment [36]. In this thesis, we consider another form of network disconnection in which the network is fragmented into several subnetworks (fragments), where the *fragment* is a connected group of sensor nodes. Figure 1.3 shows an example of a network of four fragments. Network fragmentation might be due to a number of reasons, including:

(1) *Node Failures*: which might cause network fragmentation in two different scenarios:

   (i) In harsh and hostile environments nodes might fail at a mass scale in certain regions causing the network to be fragmented. For example, a bomb or land mine detonation in a battle field might break the deployed network into several fragments. In another scenario, natural phenomena like heavy rain or a mud slide can wash or move the sensor nodes away, or even bury them where they become useless.

   (ii) Power depletion due to the unbalanced load distribution at individual sensors as a result of random deployments, i.e., hot spots.

(2) *Fragmented area of phenomenon*: in some applications, it might not be required to cover the whole field. Instead, specific regions in the field must be covered.

(3) *Environmental conditions*: the deployment of a connected wireless sensor network might not be feasible due to physical obstacles and restrictions.

Under the assumption that a FWSN tolerates faults, adding more static sensors between fragments to connect them might not be possible especially in hazardous scenarios, like battle fields, disaster areas, or areas contaminated with chemical materials. We propose a data-relay based scheme to deliver data in FWSNs that uses mobile agents to act as relay nodes between fragments. Agents move continuously in the network according to

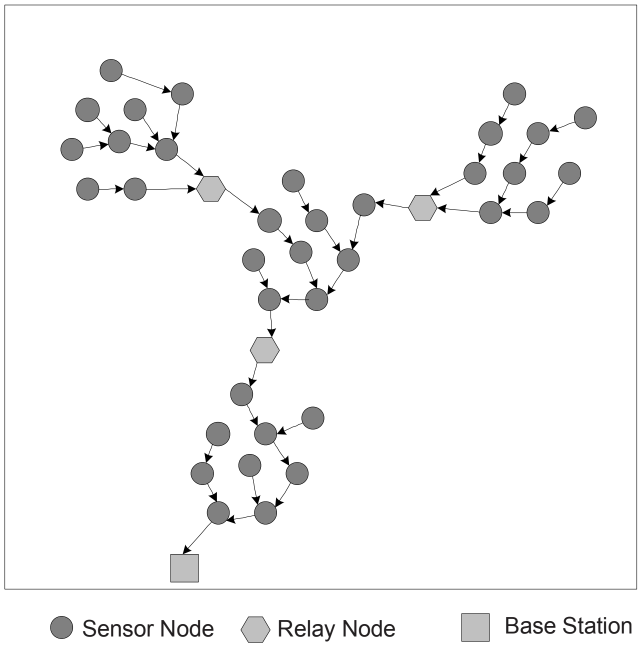


Figure 1.3: This example shows how relay nodes are used to connect a fragmented network. Arrows represent data flow between nodes.

a certain policy in order to act as relay nodes between fragments. This way, the network will be connected over a time period instead of continuous connectivity.

1.4 Thesis Contribution

Our first contribution is the introduction of the *fragmented sensor network* problem as well as the issue of data delivery in such networks. We propose a scheme that relies on mobile agents to relay the data between fragments and eventually deliver it to the base station. We have modeled the *"data delivery in fragmented wireless sensor networks"* problem as a *Closed Queueing Network* and used the model to accurately compute the average as will as the distribution of the fragment-to-fragment and fragment-to-sink data latencies. The effect of the number of mobile relays and the underlying movement policy on the data latency has been also studied. The results suggest that improving the movement policy might result in a better performance than just adding more relay nodes.

Utilizing mobile relays in large sensor networks (in terms of the area of the field) has also been studied. Our results imply that dividing the field into subfields and distributing the relay nodes over them results in better performance than just having the relay nodes serve the whole field as one entity. We also studied the problem of engineering the time that a mobile relay should spend in relaying data at each fragment. An algorithm was proposed to estimate the amount of data that will buffered in a fragment in a period of time t. Based on this amount of buffered data, we have been able to estimate the required relay time.

1.5 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we introduce a formal definition of the *"Data Delivery in Fragmented Wireless Sensor Networks"* problem and then present a Closed Queuing Network model to evaluate the performance of the system. Chapter 3 addresses the issue of fragment-to-fragment data latency as well as the effect of the movement policy on the performance. The fragment-to-sink data latency is studied in Chapter 4. Some other related issues, like dealing with large networks and calculating the sojourn times at service centers, and their effect on the performance are studied in Chapter 5. Chapter 6 concludes the thesis and suggests some future research directions. Appendix A explains how to solve a *Gordon-Newell* network for equilibrium probabilities. A table of definitions is presented in Appendix B.

CHAPTER 2 PROBLEM DEFINITION AND MODELING

In this chapter, we first give a formal definition of the *"Data Delivery in Fragmented Wireless Sensor Networks (FWSN)"* problem in Section 2.1. An introductory overview of Queueing Theory, its applications, and why we have chosen it to model the system is presented in Section 2.2. We then describe in detail the mathematical model we developed to model FWSNs, and show how to apply it to two different examples in Section 2.3. We summarize the chapter in Section 2.4.

2.1 Problem Definition

The problem addressed in this thesis is called *"Data Delivery in Fragmented Wireless Sensor Network"*, and it is defined as follows:
Given an FWSN that consists of n fragments, let K^* be the minimum number of static relay nodes required to connect the whole network[1]. We make the following assumptions:

- There are K mobile relays, $1 \leq K < K^*$.
- All mobile relays move at the same speed of L m/s.
- No more than one mobile relay is used to connect a pair of adjacent fragments. An MR relays data between a pair of adjacent fragment through direct communication as they are assumed to be close enough. [2]

[1]Note that there is always a way to connect the network over a sufficiently long time period, t, with K mobile relays, where $1 \leq K < K^*$. However, the fewer the number of mobile relays, the higher the delay.

[2]If the two fragments are far apart, then the MR should carry the data from one fragment, move close enough to the second fragment, and relay the data. This case is not studied in this thesis and it

- The connection between any pair of fragments i and j must persist for at least t^s_{ij} time units. During this period, data will be relayed from fragment i (source-fragment) to fragment j (destination-fragment) through the MR in between[3]. t^s_{ij} will be referred to as *service time* or *sojourn time* interchangeably. For the time being, t^s_{ij} is assumed to be long enough to relay all generated data, and we will revisit this issue of in Chapter 5.

Our objective in this study is the following:

- Find the distribution of the waiting time for a fragment before its data is relayed to the next fragment. We call this waiting time the *"idle time"*.
- Find the distribution of the end-to-end delay, i.e., the time to deliver data to the base station.

We make the simplifying assumption that the network can be connected with only one mobile relay (MR) between any pair of fragments. This assumption is valid because mobile relays are supposed to have a long transmission ranges.

Note that t^s_{ij} is necessary to give the mobile relay enough time to relay data between fragments. The value of t^s_{ij} depends on the amount of buffered data in fragment i which depends on three factors:

(1) Number of sensors in fragment i.

(2) The average data generation rate of a sensor in fragment i.

(3) The amount of data buffered in all fragments that pass through fragment i.

We will revisit the issue of engineering t^s_{ij} in Chapter 5. Before that, we assume that t^s_{ij} is known in advance.

will be part of our future work.

[3]In the case of distant fragments, this time period will include both the direct communication time and the movement time between the pair of fragments.

2.2 Queueing Theory Overview

Queueing Theory [37, 38] is the mathematical study of queues or waiting lines. The theory has its roots early in the last century with the studies of A. K. Erlang and A. A. Markov on stochastic processes. It is widely used to model a broad variety of systems including computers, telephone systems, military operations, water reservoirs, machine repairs and many others [39]. Queueing theory enables the evaluation of several performance measures like the probability of having the system in certain state, the average service time, and the expected queue length.

The queueing process is described as follows: *Customers* requiring service arrive to, according to a certain *arrival distribution*, to the queueing system and join a *queue* where they are buffered until being selected for service according to certain criterion known as the *service discipline*. Selected customers are then served in accordance to a *service discipline* and a *service-time distribution*. Figure 2.1 describes the queueing process. The number of *customers* that might go through the queueing system is usually referred to as the *population size*, which can be limited or unlimited. The *service discipline* is the criterion used to select members of the queue to be served. A very common criterion to buffer new arriving member is the first-come-first-serve and usually this is the default criterion assumed by queueing models unless stated otherwise. Criteria based on priority might also be used. The service mechanism includes one or more service channels called *servers* as well as the time of service that follows a *service time distribution*. The

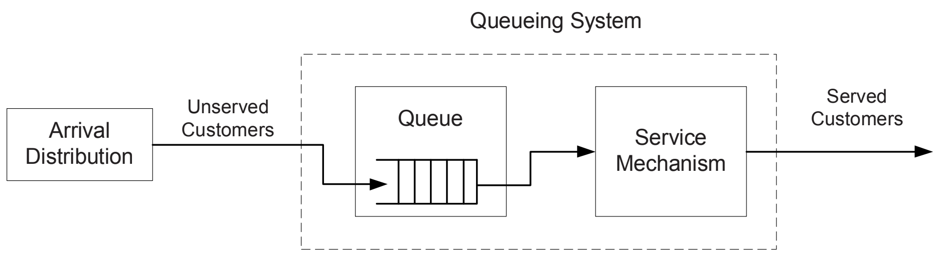


Figure 2.1: The queueing process.

Symbol	Distribution
M	Markovian
D	Deterministic
Geo	Geometric
G	General (arbitrary)

Table 2.1: Possible distributions of **A** and **B**.

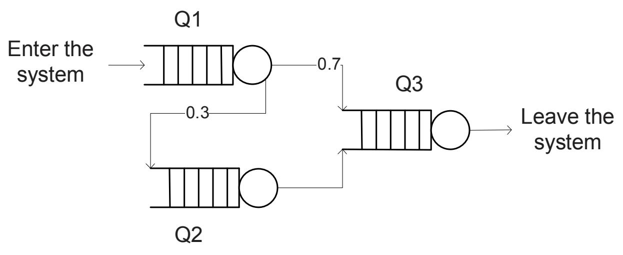
most frequently used *service time distribution* is the exponential distribution due to its Markovian nature. D. G. Kendall proposed a standard notation for classifying queueing systems into different types. This notation is,

$$\mathbf{A/B/C/D/E}$$

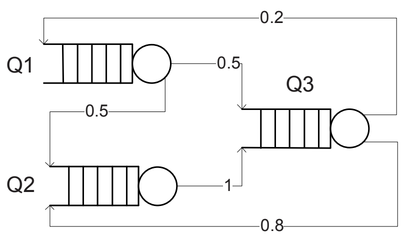
A refers to the *arrival distribution*, whereas **B** refers to the *service time distribution*. Table 2.1 shows some of the distribution types that **A** and **B** might take. **C** refers to the number of servers which might be 1 (single server), m (multiple servers), or ∞ (infinite servers). On the other hand, **B** refers to the maximum number of customers that can be accommodated in the system, i.e. the maximum buffer size. Finally, **E** represents the population size and it can be either N (i.e., limited) or ∞ (i.e., unlimited).

Queues can be interconnected together to form a network of queues, usually referred to as a *Queueing Network*. Queueing networks are classified into two categories; *open networks (also called Jackson networks)* and *closed networks (also called Gordon-Newell networks)*. An Open queueing network, see Figure 2.2(a), has external input from which new customers enter the system as well as an exit where customers leave the system. On the other hand, a closed queueing network, see Figure 2.2(b), has no external inputs or outputs. Therefore, customers circulate in the network and never leave. A third type of queueing networks is called *Mixed networks*. This type is open for some classes of customers and closed for other classes.

Values that appear on the arrows in Figures 2.2(a) and 2.2(b) are the routing probabilities between the queues in the network. For example, a customer leaving queue Q3 in



(a) Open Queueing Network



(b) Closed Queueing Network

Figure 2.2: Difference between open and closed queueing networks

Figure 2.2(b) has a probability of 0.2 to enter queue Q1 and a probability of 0.8 to enter queue Q2.

Queueing network are known to be very accurate and effective to model a vast variety of systems and provide several probabilities that can be used to measure the system performance. A closed queueing network fits our problem, as we have fixed number of customers (mobile relays) with no external departures or internal arrivals. In the next section we model the data delivery problem FWSN as a Gordon-Newell network.

2.3 Mathematical Model

Figure 2.3 shows the case-study network that will be used through out this thesis to illustrate our modeling approach. This network consists of six fragments labeled *(FRAG-1,...,FRAG-6)*. Hexagons that appear between a pair of fragments represent potential locations where mobile relays can stop to relay data from one fragment to another fragment. For example, a mobile relay at connection point 1 relays data from *FRAG-1* to *FRAG-2*, and a mobile relay at connection point 3 relays data from *FRAG-3* to *FRAG-6* and so on. From now on, connection points will be referred to as *Service Centers* (s/c).

Let s/c_k be the k^{th} service center that connects fragments i and j in the FWSN. We model s/c_k as an infinite-buffer/infinite-server queue. As a matter of fact, the number of servers and the buffer size need not to exceed k. But, in order to make the analysis tractable we use infinite-buffer/infinite-server queues. Each server offers service according to an exponentially distributed service time with a rate $\mu_k = \frac{1}{t^s_{ij}}$. We call such a queue a *relay queue.*

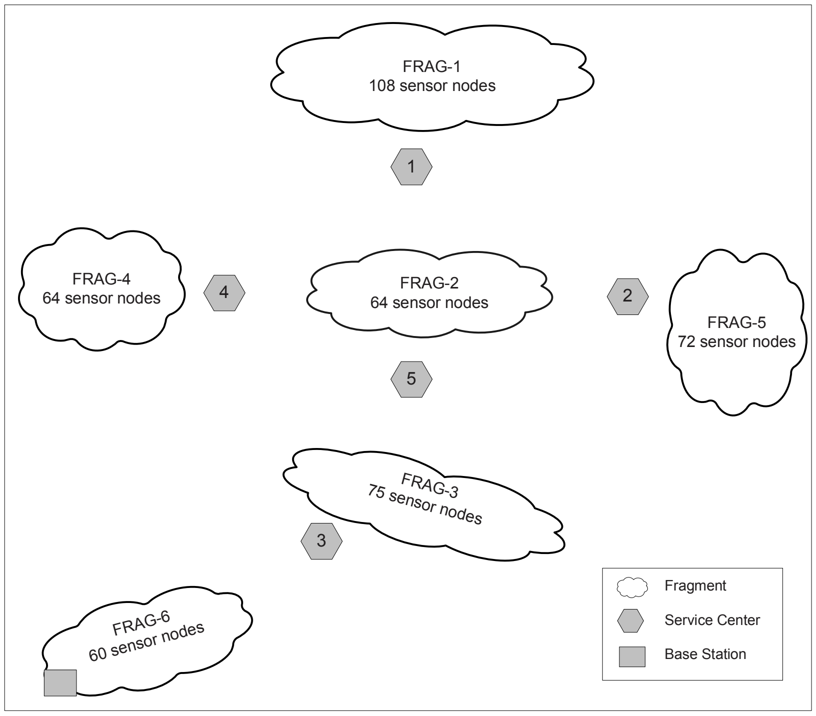


Figure 2.3: A case-study FWSN with six fragments and five service centers.

For the time being we assume that the routing probabilities between service centers are given, and we will revisit this issue later. Table 2.2 shows sample routing probabilities that are used for our case study, where q_{ij} represents the probability that a MR leaving s/c_i goes to s/c_j.

This model does not capture the effect of the trip time between service centers. To capture this time, the following modifications are introduced:

- For every pair of *relay queues* (i, j), for which $q_{ij} > 0$ we add an infinite-buffer/infinite-server queue, we call such a queue a *movement queue*, with a mean service rate

$\mu_i = \frac{L}{d_{ij}}$, where L is the MR speed and d_{ij} is the distance between s/c_i and s/c_j. We assume that the route that a MR takes from i to j is the same as that it takes from j to i, therefore $d_{ij} = d_{ji}$.

- The probability that a customer leaving queue i goes to the newly added movement queue is the same as the probability that a MR leaving s/c_i goes to s/c_j. On the other hand, the probability that a customer leaving the movement queue goes to queue j is 1.

In the remainder of this thesis, we will use the word *queue* to refer a queue of any type (i.e. movement or relay). To reference a particular type we use *"movement queue"* or *"relay queue"*.

We model the number of mobile relays in the network as the number of customers that circulate in the closed queueing network. Tables 2.2 and 2.3 summarize all the parameters associated with the case study in Figure2.3 and they will be used in all analysis and simulations regarding this case through out this work unless mentioned otherwise.

Using the parameters in Table 2.3 and the mapping procedure described earlier we construct the Gordon-Newell network model shown in Figure 2.4 for the case study shown in Figure 2.3. Gray nodes represent the *relay queues* and the other nodes represent the *movement queues*. μ_i represents the service rate of queue i. Appendix A explains how to solve the queueing network for equilibrium probabilities.

2.4 Summary

Data delivery in fragmented wireless sensor networks poses several questions on how to utilize a given number of mobile relays that move according to a certain policy (i.e., routing probabilities between service centers) to minimize the data latency. This problem

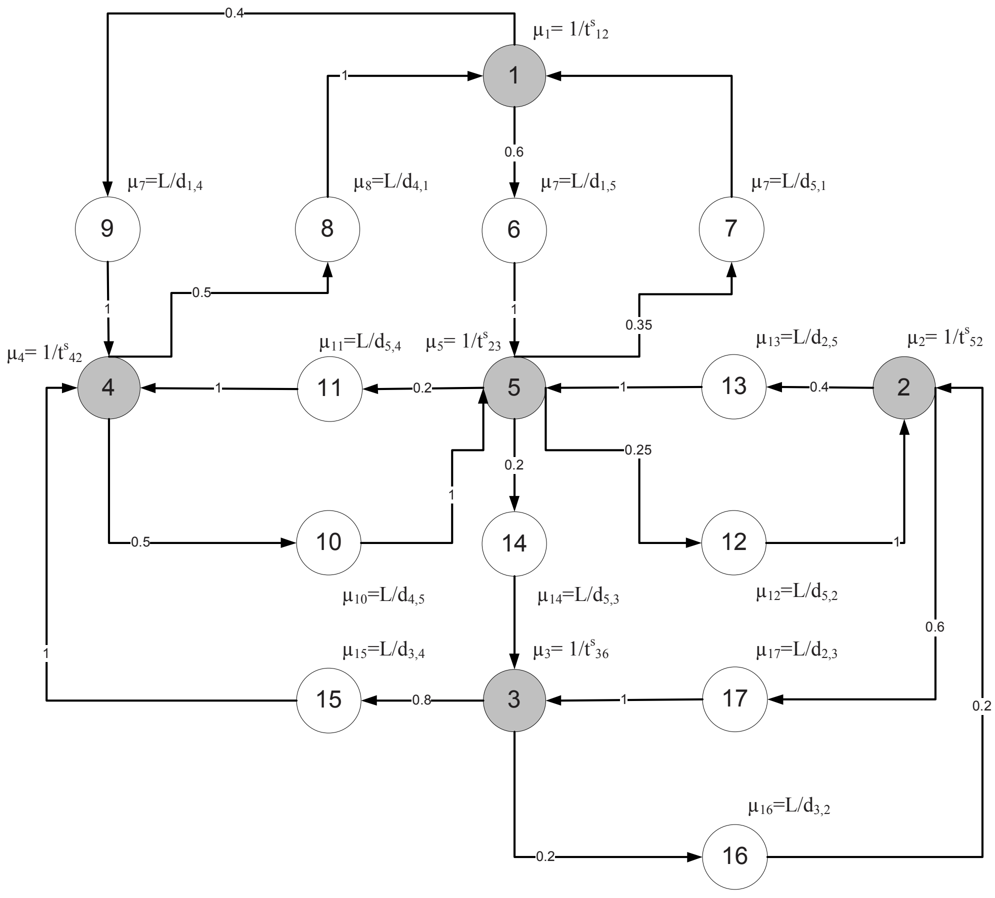


Figure 2.4: *Gordon-Newell Network* model for the case-study shown in Figure 2.3

was formally defined in Section 2.1 and modeled using a *Gordon-Newell* network in Section 2.3. In the next chapter we derive the probability distribution as well as the expected value of the idle time at any service center in the network.

Service center	q_{i1}	q_{i2}	q_{i3}	q_{i4}	q_{i5}
s/c_1	0.0	0.0	0.0	0.4	0.6
s/c_2	0.0	0.0	0.6	0.0	0.4
s/c_3	0.0	0.2	0.0	0.8	0.0
s/c_4	0.5	0.0	0.0	0.0	0.5
s/c_5	0.35	0.25	0.2	0.2	0.0

Table 2.2: Sample routing probabilities between different service centers which are used in our case study in Figure 2.3.

Network Parameter	Value
Num. of MRs	2
Num. of s/c's	5
Speed (L)	$1.2\ and\ 3.89m/s$
t^s_{ij}	$1min$
d_{14}	$391.76m$
d_{15}	$368.76m$
d_{25}	$401.6m$
d_{23}	$680.34m$
d_{34}	$463.1m$
d_{35}	$321.74m$
d_{45}	$357.94m$

Table 2.3: System parameters of the case-study shown in Figure 2.3.

CHAPTER 3 IDLE TIME DISTRIBUTION

The idle time of *relay queues* is a direct indication of the amount of time that fragments need to wait before receiving service. In the case of one service center per fragment those two time periods (idle time and waiting time to receive service) are exactly the same, whereas in the case where more than one service center is assigned to a certain fragment the waiting time depends on the routing protocol. For example, in a *hot-potato routing* scheme, the waiting time for a fragment to receive service is the minimum idle time of all service centers assigned to it. However, in a *minimum number of hops (i.e., shortest path)* routing, the waiting time is the idle time of the service center that lies on the shortest path. Therefore, it is of great importance to predict the idle time to give a clear picture of how well the system performs. This thesis considers the case of one service center per fragment only.

In this chapter, we first introduce some definitions that we then use to evaluate the distribution of the idle time at a queue. Then, we use this distribution to evaluate the average idle time at a queue and as a result at a service center. This idle time distribution can also be used to evaluate the distribution of the total time that the data might spend being relayed from one fragment to another before it eventually reaches the base station (i.e. the *end-to-end delay*).

This chapter is organized as follows. In Section 3.1, we first give some definitions before going through a detailed mathematical derivation to derive the idle time distribution at any service center. Then, we validate our analytical results by comparing them to results from a simulation model in Section 3.2. The effect of the movement policy on the performance is studied in Section 3.3. We conclude the findings of this chapter in Section 3.4.

3.1 Idle-time distribution

In this section we derive the idle time distribution at any queue. The idle time at a service queue represents the idle time at the service center that this queue models. We derive the idle time distribution in two forms:

(1) *Joint distribution*: in which the probability that *queue* i is idle for time $t \leq \tau$ is taken jointly with the probability that *queue* i has been idle at the reference time $(t = 0)$.

(2) *Conditional distribution*: in which the probability that *queue* i is idle for time $(t \leq \tau)$ is conditioned on the fact that *queue* i was idle at the reference time $(t = 0)$.

As we will see in Chapter 4, the idle time distribution in these two different forms is required for the evaluation of the *end-to-end* delay. We start by computing the *end-to-end* delay along certain path from the point where the first service center on that path becomes idle. Therefore, we use the *Conditional distribution* for the first service center and the *Joint distribution* for the remaining service centers on the path. We start this section by introducing some definitions and then derive the *Joint distribution*. The *Conditional distribution* is evaluated as the *Joint distribution* divided by the probability that the queue is idle at the reference time $(t = 0)$.

3.1.1 Definitions

- M is the number of queues in the network (this includes both relay and movement queues).

- K is the number of customers (i.e., mobile relays).

- $\overrightarrow{N} = \{n_1(\overrightarrow{N}), \ldots, n_M(\overrightarrow{N})\}$ is the state of the system, i.e., queueing network, in which the K customers are distributed over the M queues such that queue i has $n_i(\overrightarrow{N})$ customers in this state. Note that $\sum_{i=1}^{M} n_i(\overrightarrow{N}) = K$ for any state $\overrightarrow{N}$.

- Ω is the set of all possible system states, $|\Omega| = \binom{M+K-1}{K}$.

- μ_j is the state-independent service rate of *queue* j.

- q_{ij} is the probability that a customer leaving *queue* i will move to *queue* j (we assume that there is one class of customers in the system). In our queueing model, $q_{ii} = 0$.

- $\pi(\overrightarrow{N})$ is the steady-state probability of state $\overrightarrow{N}$, such that $\sum_{\overrightarrow{N} \in \Omega} \pi(\overrightarrow{N}) = 1$.

- n_i is used to refer to the number of customers at *queue* i regardless of the system state.

- E_i is the number of system states in which $n_i = 0$.

- $p^i_{idle}(t, \overrightarrow{N})$ is the probability that at time t, *queue* i is in state $\overrightarrow{N}$ in which $n_i(\overrightarrow{N}) = 0$, and n_i became zero at time $t = 0$ and remained so in $[0, t]$ (i.e., joint probability). In other words, this is the probability that *queue* i is idle for time greater than t. Note that, $p^i_{idle}(t, \overrightarrow{N})$ is defined only over network states in which $n_i = 0$.

- $\overrightarrow{p}^i_{idle}(t) = [p^i_{idle}(t, \overrightarrow{N}_1), \ \ldots, \ p^i_{idle}(t, \overrightarrow{N}_{E_i})]^T$.

- U_i is a row vector of ones such that $|U_i| = E_i$.

- T_i is a random variable that represents the idle time of *queue* i.

- $F_{T_i}(t) = probability\,(T_i \leq t)$, is the joint cumulative distribution function (CDF) of T_i (i.e., the joint distribution that queue i starts and idle period at the reference time ($t^* = 0$) and stays idle for time $T_i \leq t$).

- $f_{T_i}(t) = \frac{d}{dt}F_{T_i}(t)$, is the probability density function (pdf) of T_i.

- $F_{T_i|I}(t) = probability\{\ T_i \leq t \mid n_i = 0\ at\ t = 0\}$, i.e., the conditional distribution that queue i stays idle for time $T_i \leq t$ given that it starts an idle period at the reference time $t^* = 0$.

- $f_{T_i|I}(t) = \frac{d}{dt}F_{T_i\ |\ I}(t)$

3.1.2 The Joint Distribution ($F_{T_i}(t)$)

In an *infinite-server exponential service time queue* i, the probability of serving a customer within a very small time interval $\Delta t \rightarrow 0$, given that there are n_i customers in the queue, is given by:

$$n_i\mu_i\Delta t \ + \ o(\Delta t) \tag{3.1}$$

where $o(\Delta t)$ is a function that approaches zero faster than Δt. Therefore, the following *forward Chapman-Kolmogorov equation* holds;

$$\begin{aligned} p^i_{idle}(t+\Delta t, \overrightarrow{N}) \ &= \ p^i_{idle}(t, \overrightarrow{N}) \cdot prob(\overrightarrow{N}\ at\ t+\Delta t|\overrightarrow{N}\ at\ t) \\ &+ \sum_{\overrightarrow{N}^* \in \Omega} p^i_{idle}(t, \overrightarrow{N}^*) \cdot prob(\overrightarrow{N}\ at\ t+\Delta t|\overrightarrow{N}^*\ at\ t) \end{aligned} \tag{3.2}$$

In this Markovian system only one event (i.e., transition) is possible in a very short time period Δt. Therefore, the probability that the system will evolve from state $\overrightarrow{N^*}$ at t to state $\overrightarrow{N}$ at $t + \Delta t$ is given by:

$$prob(\overrightarrow{N}\ at\ t{+}\Delta t | \overrightarrow{N}^*\ at\ t) = \begin{cases} n_j(\overrightarrow{N}^*)\mu_j q_{jk}\Delta t + o(\Delta t) & \begin{array}{l} if\ \overrightarrow{N} = \overrightarrow{N}^* + \overrightarrow{1}_j - \overrightarrow{1}_k \\ \forall j,k \in \{1,2,\ldots,M\}.\ j,k \neq i \end{array} \\ \\ 0 & otherwise \end{cases} \tag{3.3}$$

Note that this probability is zero when the difference between the two states is more than one customer, because it means that a single transition can not move the system to state $\overrightarrow{N}$.

On the other hand, the probability of no transition in the time period $[t, t+\Delta t]$ is given by:

$$prob(\overrightarrow{N}\ at\ t + \Delta t | \overrightarrow{N}\ at\ t) = 1 - \sum_{\substack{j=1 \\ j\neq i}}^{M} n_j(\overrightarrow{N})\mu_j \Delta t + o(\Delta t) \tag{3.4}$$

Using equations (3.3) and (3.4) in equation (3.2) we get,

$$\begin{aligned} p^i_{idle}(t+\Delta t, \overrightarrow{N}) &= \sum_{\substack{j=1 \\ j\neq i}}^{M} \sum_{\substack{k=1 \\ k\neq i \\ n_k(\overrightarrow{N})>0}}^{M} n_j(\overrightarrow{N} + \overrightarrow{1}_j - \overrightarrow{1}_k)\mu_j \Delta t \cdot p^i_{idle}(t, \overrightarrow{N} + \overrightarrow{1}_j - \overrightarrow{1}_k) q_{jk} \\ &+ \left(1 - \sum_{\substack{j=1 \\ j\neq i}}^{M} n_j(\overrightarrow{N})\mu_j \Delta t\right) \cdot p^i_{idle}(t, \overrightarrow{N}) + o(\Delta t) \end{aligned}$$

Rearranging the terms we get,

$$\begin{aligned} \frac{p^i_{idle}(t{+}\Delta t, \overrightarrow{N}) {-} p^i_{idle}(t, \overrightarrow{N})}{\Delta t} &= \sum_{\substack{j=1 \\ j\neq i}}^{M} n_j(\overrightarrow{N}{+}\overrightarrow{1}_j{-}\overrightarrow{1}_k)\mu_j \cdot \sum_{\substack{k=1 \\ k\neq i}}^{M} p^i_{idle}(t, \overrightarrow{N}{+}\overrightarrow{1}_j{-}\overrightarrow{1}_k) q_{jk} \\ &- \sum_{\substack{j=1 \\ j\neq i}}^{M} n_j(\overrightarrow{N})\mu_j \cdot p^i_{idle}(t, \overrightarrow{N}) + \frac{o(\Delta t)}{\Delta t} \end{aligned}$$

Taking the limit as $\Delta t \rightarrow 0$, we obtain

$$\frac{d}{dt} \overrightarrow{p}^{\,i}_{idle}(t) = \mathbf{A}_i \overrightarrow{p}^{\,i}_{idle}(t)$$

Where $\mathbf{A}_i = [a^i_{xy}]$ is an $E_i \times E_i$ matrix such that:

$$a^i_{xy} = \begin{cases} -\sum_{\substack{j=1 \\ j\neq i}}^{M} n_j(\overrightarrow{N}_x)\mu_j & \text{if } x = y \\ n_j(\overrightarrow{N}_y)\mu_j q_{jk} & \text{if } \overrightarrow{N}_y = \overrightarrow{N}_x + \overrightarrow{1}_j - \overrightarrow{1}_k \\ 0 & \textit{otherwise} \end{cases} \tag{3.5}$$

Therefore,

$$\overrightarrow{p}^{\,i}_{idle}(t) = e^{\mathbf{A}_i t}\overrightarrow{p}^{\,i}_{idle}(0) \tag{3.6}$$

$p^i_{idle}(0, \overrightarrow{N})$ can be found easily using the steady state probabilities as well as the routing probabilities between service centers. Let ξ^{ik}_j be the transition of a customer from queue i to queue k (for $k \neq i$) that will make the system evolve from state $\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k$ to state $\overrightarrow{N}_j$ such that $n_i(\overrightarrow{N}_j) = 0$ and $n_k(\overrightarrow{N}_j) > 0$, i.e., $n_i(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k) = 1$. In other words, ξ^{ik}_j is the transition that will initiate an idle period at queue i. Let $prob(\xi^{ik}_j)$ be the probability that this transition takes place before any other transition. Then,

$$p^i_{idle}(0, \overrightarrow{N}_j) = \sum_{\substack{k=1 \\ k\neq i \\ n_k(\overrightarrow{N}_j)>0}}^{M} \pi(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k) \cdot prob(\xi^{ik}_j) \tag{3.7}$$

The steady state probability $\pi(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k)$ can be obtained easily using the *convolution algorithm* [40] (see Appendix A). On the other hand, $prob(\xi^{ik}_j)$ is given as:

$$prob(\xi^{ik}_j) = \frac{n_i(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k)\mu_i}{\sum_{l=1}^{M} \mu_l n_l(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k)} q_{ik} = \frac{\mu_i}{\sum_{l=1}^{M} \mu_l n_l(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k)} q_{ik} \tag{3.8}$$

Using equations (3.7) and (3.8) we get,

$$p^i_{idle}(0, \overrightarrow{N}_j) = \sum_{\substack{k=1 \\ k\neq i \\ n_k(\overrightarrow{N}_j)>0}}^{M} \frac{\mu_i q_{ik}}{\sum_{l=1}^{M} \mu_l n_l(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k)} \pi(\overrightarrow{N}_j + \overrightarrow{1}_i - \overrightarrow{1}_k) \tag{3.9}$$

We are ready now to give the final expression for the joint distribution $F_{T_i}(t)$ that we defined earlier,

$$F_{T_i}(t) = 1 - U_i e^{A_i t} \overrightarrow{p}^{\,i}_{idle}(0) \tag{3.10}$$

Note that, according to the definition we made earlier for $\overrightarrow{p}^{\,i}_{idle}(t)$, we have

$$\overrightarrow{p}^{\,i}_{idle}(0) = [p^i_{idle}(0, \overrightarrow{N}_1), \ldots, p^i_{idle}(0, \overrightarrow{N}_{E_i})]^T$$

3.1.3 Conditional Distribution

The next step is to evaluate $F_{T_i|I}$. Let $p^i_{idle}(t, \overrightarrow{N}|n_i = 0 \ at \ t = 0)$ be the probability that queue i is idle for more than t given that it was idle at time $t = 0$. Therefore,

$$p^i_{idle}(t, \overrightarrow{N}|n_i = 0 \ at \ t = 0) = e^{A_i t} p^i_{idle}(0, \overrightarrow{N}|n_i = 0 \ at \ t = 0) \tag{3.11}$$

But, $p^i_{idle}(0, \overrightarrow{N}|n_i = 0 \ at \ t = 0)$ is given by:

$$p^i_{idle}(0, \overrightarrow{N}|n_i = 0 \ at \ t = 0) = \frac{p^i_{idle}(0, \overrightarrow{N})}{\displaystyle\sum_{\substack{\overrightarrow{N}^* \in \Omega \\ n_i(\overrightarrow{N}^*)=0}} p^i_{idle}(0, \overrightarrow{N}^*)} \tag{3.12}$$

Substituting equation (3.11) in (3.12) we get,

$p^i_{idle}(0, \overrightarrow{N}|n_i = 0 \ at \ t = 0) =$

$$\frac{\displaystyle\sum_{\substack{k=1 \\ k\neq i \\ n_k(\overrightarrow{N})>0}}^{M} \left(\frac{\mu_i}{\displaystyle\sum_{l=1}^{M} \mu_l n_l(\overrightarrow{N} + \overrightarrow{1}_i - \overrightarrow{1}_k)} q_{ik} \pi(\overrightarrow{N} + \overrightarrow{1}_i - \overrightarrow{1}_k) \right)}{\displaystyle\sum_{\substack{\overrightarrow{N}^* \in \Omega \\ n_i(\overrightarrow{N}^*)=0}} \sum_{\substack{k=1 \\ k\neq i \\ n_k(\overrightarrow{N}^*)>0}}^{M} \frac{\mu_i}{\displaystyle\sum_{l=1}^{M} \mu_l n_l(\overrightarrow{N}^* + \overrightarrow{1}_i - \overrightarrow{1}_k)} q_{ik} \pi(\overrightarrow{N}^* + \overrightarrow{1}_i - \overrightarrow{1}_k)} \tag{3.13}$$

Finally, the conditional distribution $F_{T_i|I}(\tau)$ is given by,

$$F_{T_i|I}(\tau) = 1 - U_i e^{A_i \tau} \overrightarrow{p}^{\,i}_{idle}(0|n_i = 0 \ at \ t = 0), \tag{3.14}$$

where $\overrightarrow{p}^{i}_{idle}(0|n_i = 0 \; at \; t = 0)$ is given by,

$$\overrightarrow{p}^{i}_{idle}(0|n_i = 0 \; at \; t = 0) = [p^{i}_{idle}(0, \overrightarrow{N}_1|n_i = 0 \; at \; t = 0), \ldots, p^{i}_{idle}(0, \overrightarrow{N}_{E_i}|n_i = 0 \; at \; t = 0)]^T \tag{3.15}$$

3.2 Numerical Results

We applied the above expressions in equations (3.10) and (3.14) to evaluate $F_{T_i|I}(t)$ and $F_{T_i}(t)$ for all the service centers in our case study (Figure 2.3 and Tables 2.2 and 2.3). The results are shown in Figures 3.1 and 3.2. Notice that $F_{T_i|I}(0)$=0 because at t=0 s/c_i has just become idle and it will stay idle for non-zero time. However, $F_{T_i}(0)$ can be non-zero which is the probability that s/c_i is idle at time $t = 0$. We can also see that the probability of having an idle period of length less than or equal t at service centers s/c_2 and s/c_3 is greater than other service centers. This implies that the average idle time at these service centers is greater than others.

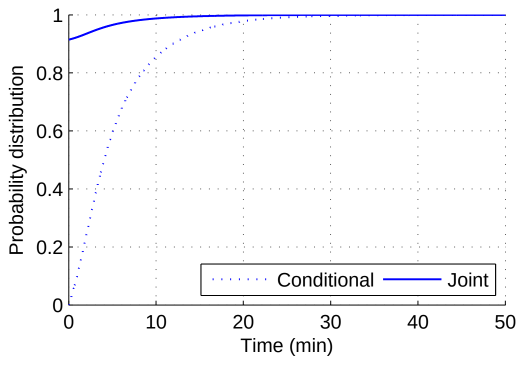
Using the idle-time distributions shown in Figures 3.1 and 3.2, we can obtain the mean idle time for any service center. The average idle time for service center i, denoted by $E[T_i|I]$ is given by:

$$\begin{aligned} E[T_i|I] &= \int_{t=0}^{\infty} \left(1 - F_{T_i|I}(t)\right) dt \\ &= \int_{t=0}^{\infty} U_i e^{A_i \tau} \overrightarrow{p}^{i}_{idle}(0|n_i{=}0 \; at \; t{=}0) dt \end{aligned} \tag{3.16}$$

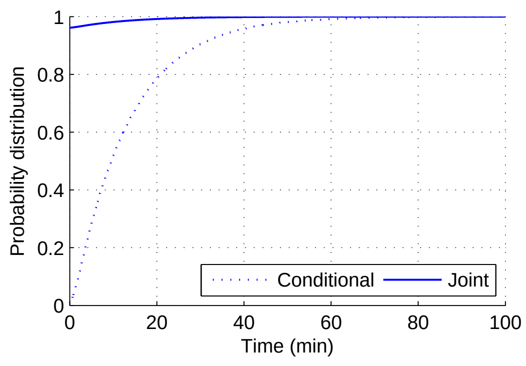
Evaluating equation (3.16) involves the integration of a matrix exponential ($e^{A_i t}$) which can be very complicated depending on the size of the matrix A_i. Therefore, we evaluate this expression numerically as follows:

$$E[T_i|I] = \sum_{k=0}^{\infty} \left(1 - F_{T_i|I}(k\Delta)\right) \Delta \tag{3.17}$$

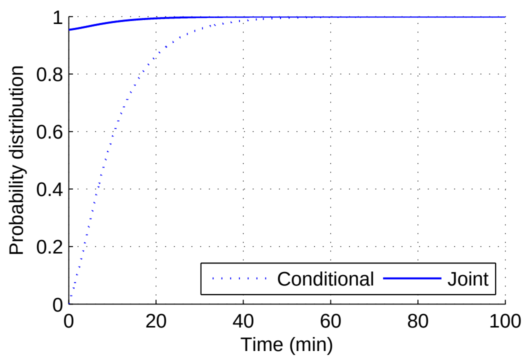
By keeping Δ small, we can obtain very accurate results (see Figures 3.4 and 4.6(b)). Figure 3.2 shows the average idle time at all service centers evaluated using equation



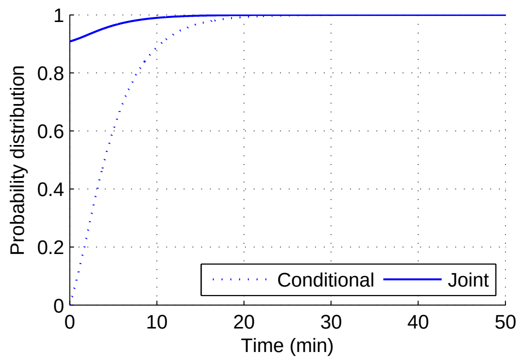
(a) $F_{T_1|I}(t)$ and $F_{T_1}(t)$



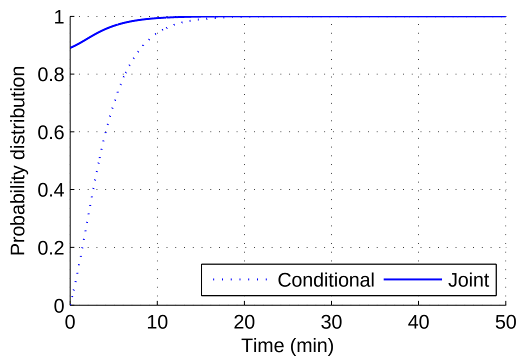
(b) $F_{T_2|I}(t)$ and $F_{T_2}(t)$



(c) $F_{T_3|I}(t)$ and $F_{T_3}(t)$

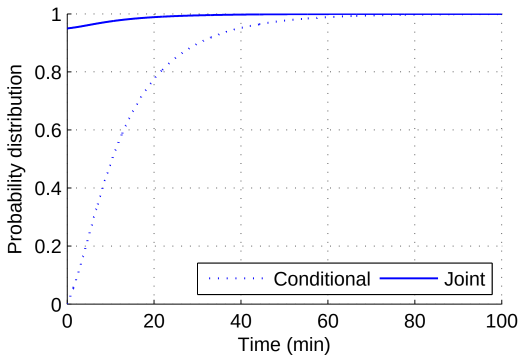


(d) $F_{T_4|I}(t)$ and $F_{T_4}(t)$

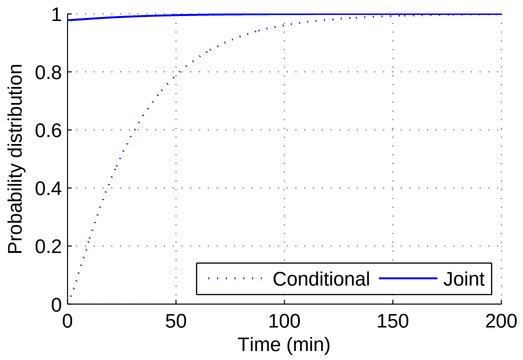


(e) $F_{T_5|I}(t)$ and $F_{T_5}(t)$

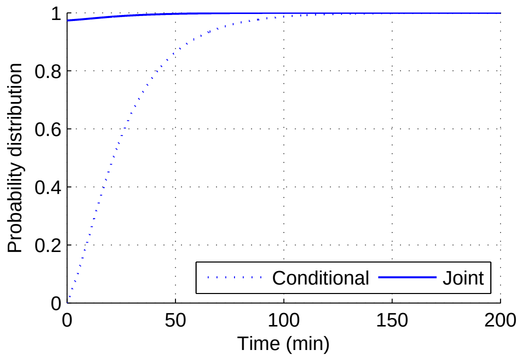
Figure 3.1: The conditional, $F_{T_i|I}(t)$, and joint, $F_{T_i}(t)$, idle time distributions at all service centers at a mobile relay speed of 3.89m/s



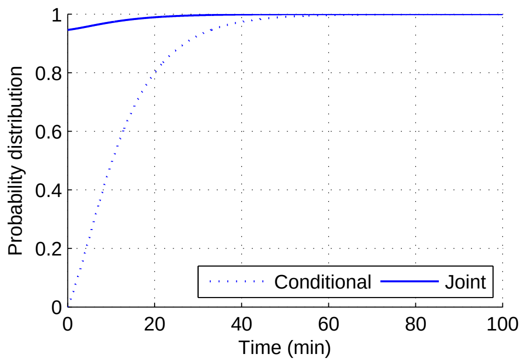
(a) $F_{T_1|I}(t)$ and $F_{T_1}(t)$



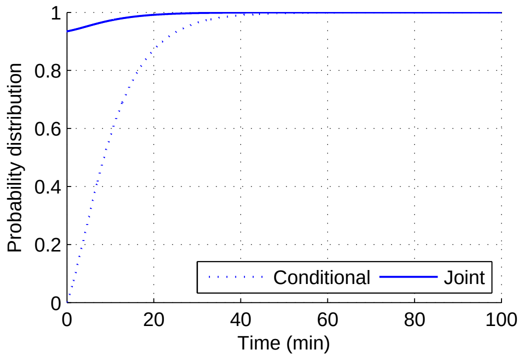
(b) $F_{T_2|I}(t)$ and $F_{T_2}(t)$



(c) $F_{T_3|I}(t)$ and $F_{T_3}(t)$



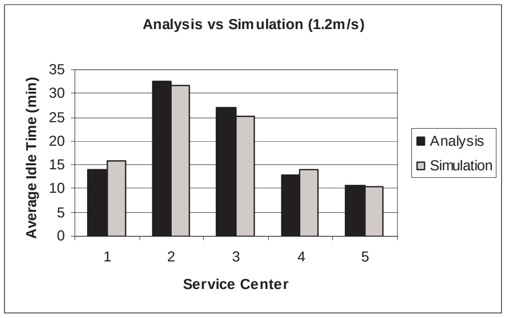
(d) $F_{T_4|I}(t)$ and $F_{T_4}(t)$



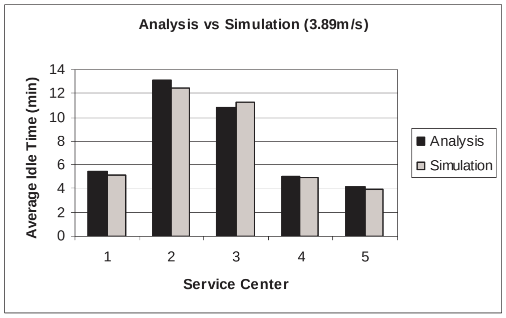
(e) $F_{T_5|I}(t)$ and $F_{T_5}(t)$

Figure 3.2: The conditional, $F_{T_i|I}(t)$, and joint, $F_{T_i}(t)$, idle time distributions at all service centers at a mobile relay speed of 1.2m/s

(3.17) at two different speeds, namely $1.2m/s$ and $3.89m/s$, with $\Delta = 0.01\ min$. As can be observed, as we expected from the distribution in Figures 3.1 and 3.2, some service centers (like 2 and 3) experience large idle periods compared to others (like 4 and 5). We can also see that increasing the speed from $1.2m/s$ to $3.89m/s$ reduces the average idle time over all service centers by an average of about 61%. As different service centers experience different delays, fragments will be served at different rates and therefore have different data latencies. To validate our analytical modeling approach we simulated the network using the TOSSIM simulator [41]. TOSSIM simulates TinyOs sensor networks at the bit level which guarantees a high level of fidelity [42]. We also used *TinyViz* as a GUI interface with TOSSIM. To simulate mobility, we used Tython [43]. Tython is a scripting language for TOSSIM that uses a Java implementation of the Python programming language. Using Tython it is possible to modify the simulation environment, like moving motes, injecting packets, pausing and resuming simulation, changing the radio model, and inspecting the simulation state. Figure 3.2 compares the analytical results to those obtained by simulation. It is evident that our analysis is very accurate.



(a) MRs move at a speed of $1.2m/s$



(b) MRs move at a speed of $3.89m/s$

Figure 3.3: The average idle time, $E[T_i|I]$, obtained through both simulation and analysis at all service centers

Figure 3.4 shows the effect of Δ on the accuracy of the average idle time, $E[T_i|I]$, obtained numerically using equation (3.17) for s/c_5 at a speed of $1.2m/s$. The value of the expected idle time using $\Delta = 1min$ is less by only 0.04% than that computed using $\Delta = 0.01min$, which implies that Δ does not crucially affect our estimations.

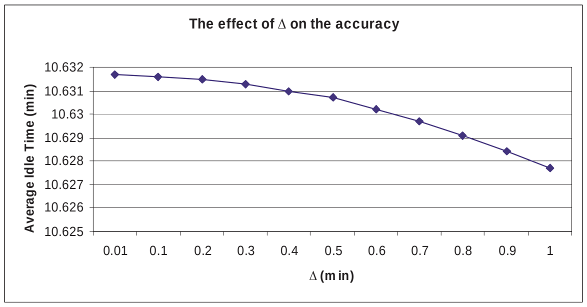


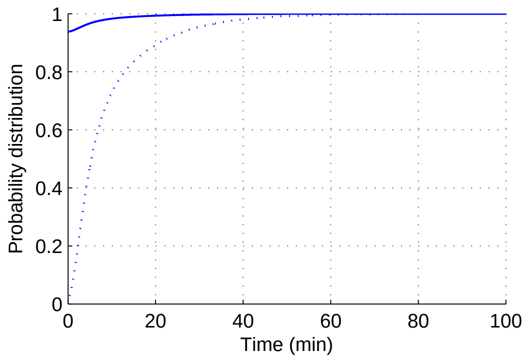
Figure 3.4: The effect of Δ on the accuracy of equation 3.17.

3.2.1 Additional Example

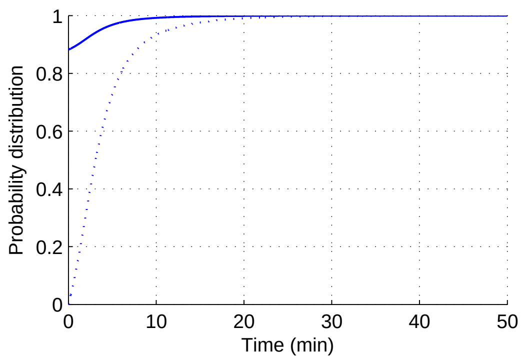
To confirm the accuracy of our model, we present another example that consists of five fragments and four service centers as shown in Figure 3.6. Figure 3.7 shows the closed queueing network model for this example. We obtained the average idle times at all service centers in this example using both analysis and simulation based on the parameters shown in Table 3.1. Idle time distributions are shown in Figure 3.5. We can tell from those distributions that s/c_1 and s/c_4 will experience larger idle times than s/c_2 and s/c_3, and that is what we can see in Figure 3.8. Figure 3.8 shows the average idle times at all service centers obtained using both simulation and analysis. The results confirm the accuracy of our model. The maximum error in the evaluations in Figure 3.8 is about (4%). Note that the policy used in this example produces an average idle time at s/c_1 that is almost equal to that at s/c_4, and also produces an average idle time at s/c_2 that is almost equal to that at s/c_3.

Network Parameter	Value
Num. of MRs	2
Num. of s/c's	4
Speed (L)	$3.89m/s$
t_{ij}^s	$1.2min$
d_{12}	$451.74m$
d_{23}	$334.18m$
d_{34}	$551.26m$

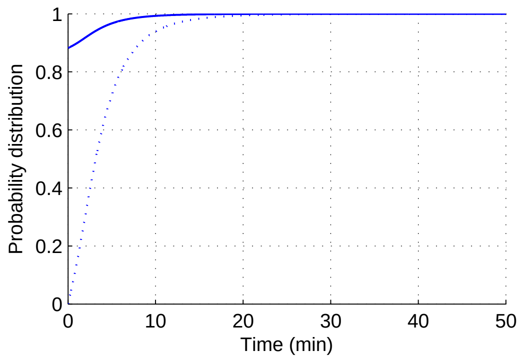
Table 3.1: System parameters of the example shown in Figure 3.6.



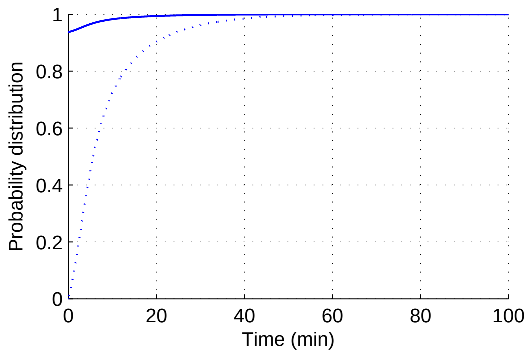
(a) $F_{T_1|I}(t)$ and $F_{T_1}(t)$



(b) $F_{T_2|I}(t)$ and $F_{T_2}(t)$



(c) $F_{T_3|I}(t)$ and $F_{T_3}(t)$



(d) $F_{T_4|I}(t)$ and $F_{T_4}(t)$

Figure 3.5: The conditional, $F_{T_i|I}(t)$, and joint, $F_{T_i}(t)$, idle time distributions at all service centers in the example shown in Figure 3.6 using the parameters in Table 3.1

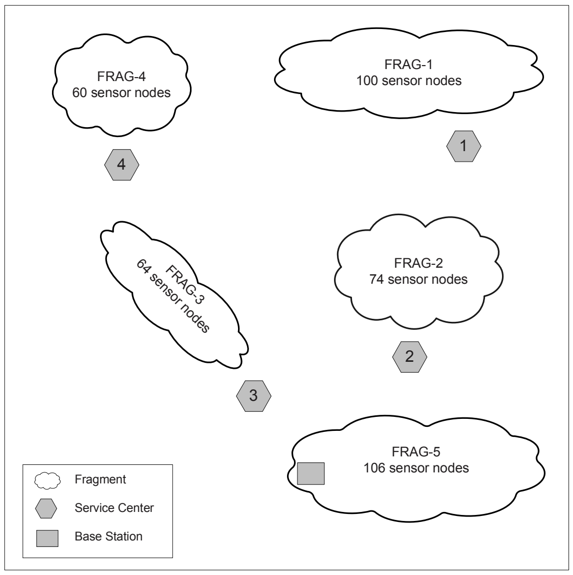


Figure 3.6: A sample FWSN with five fragments and four service centers.

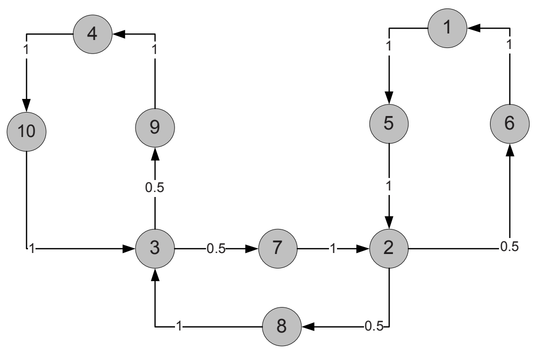


Figure 3.7: *Gordon-Newell Network* model for the example shown in Figure 3.6

3.3 Movement Policies

As we have observed in Section 3.2, service centers did not receive the same share of service and consequently experienced different average idle times with a large difference between the minimum (which was 10.63min at a speed of 1.2m/s and 4.10min at a speed of 3.89m/s) and the maximum (which was 32.61min at a speed of 1.2m/s and 13.1m/s) averages. This does not only affect the poorly-served fragments but it also affects other fragments. For instance, FRAG-3 in Figure 2.3 which is served poorly by s/c_3 will

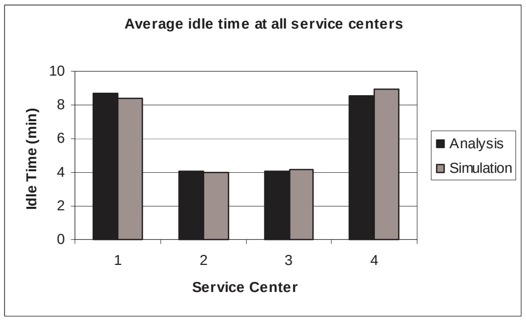


Figure 3.8: The average idle time at all service centers in the example shown in Figure 3.6

affect all other fragments because all data paths pass through FRAG-3. Therefor, it is important to guarantee that all service centers receive a fair share of service. Four parameters influence the performance of the system:

(1) Number of mobile relays,

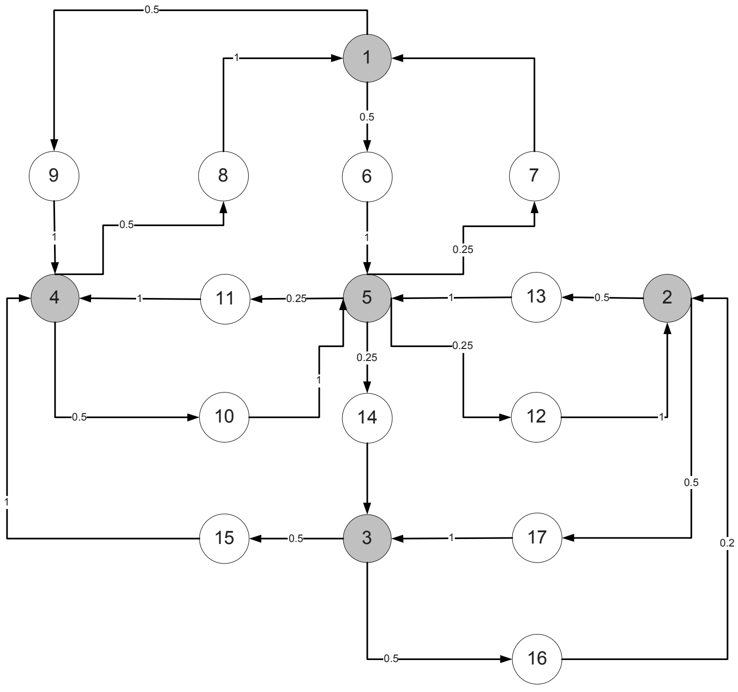
(2) Speed of mobile relays,

(3) Sojourn times at different service centers, and

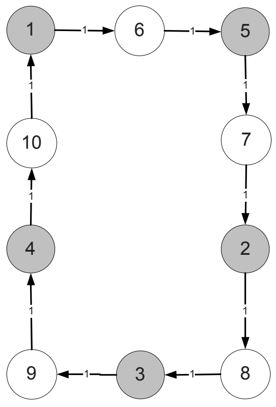
(3) The routing probabilities between different service centers in the network.

Assuming that sojourn times are strictly constrained, we are left with three parameters that may affect the average idle time; number of mobile relays, speed of movement, and movement policy. Intuitively, increasing the number and/or speed of mobile relays results in a decrease of the average idle time and consequently the *end-to-end* delay. But, how does the movement policy (i.e., routing probabilities between service centers) affect the average idle time? In order to answer this question, we propose three different movement policies; *uniform*, *distance based*, and *deterministic*. In a uniform policy, a mobile relay leaving a certain service center goes to any of the reachable neighboring service centers with the same probability. The distance based policy suggests that a mobile relay goes to closer neighboring service centers with higher probability than farther ones. The third policy assumes that a mobile relay follows a deterministic route through all the service centers (i.e. a cycle that goes over all service centers). Figure 3.9 shows the queueing network models for our case study defined in Chapter 2 under all the three policies.

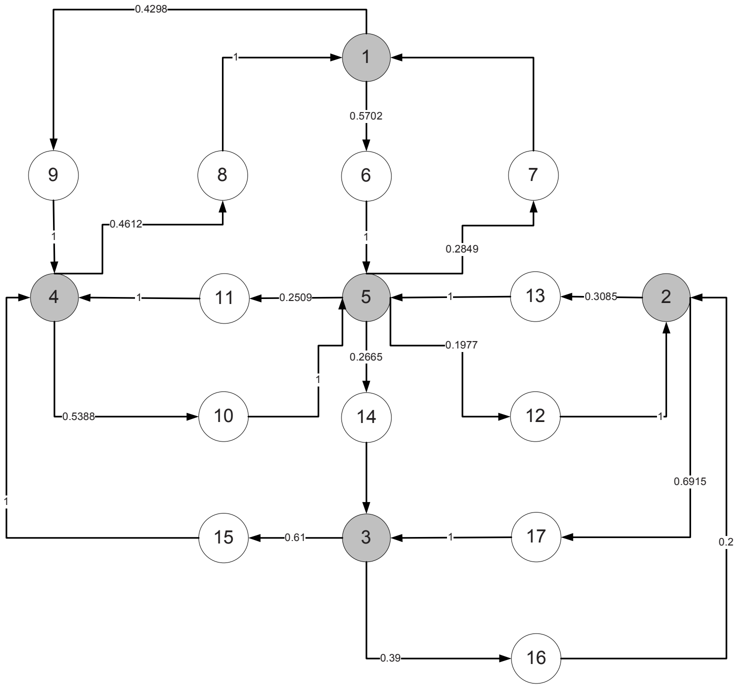
We modeled all three policies as well as the random policy shown in Figure 2.4 and evaluated their performance. Figure 3.10 shows the average idle times at all service centers using all four policies obtained by TOSSIM simulations using $1.2m/s$ and $3.89m/s$ speeds. Note s/c_2 and s/c_3 suffer more than others in all policies except the deterministic



(a) Queueing network model for the uniform policy



(b) Queueing network model for the deterministic policy

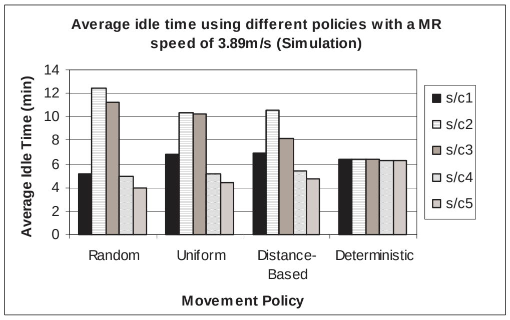


(c) Queueing network model for the distance based policy

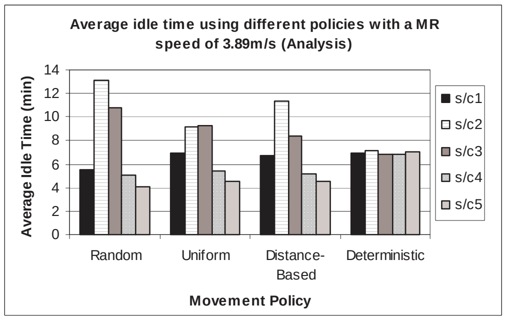
Figure 3.9: The *Gordon-Newell* queueing network for the uniform, deterministic, and distance based routing policies.

policy, and that s/c_3 lies on the data paths of $FRAG{-}1$ though $FRAG{-}5$. Therefore, the *end-to-end* delay of the data generated at those fragments will be negatively affected by the poor service at s/c_3. On the other hand, s/c_3 receives a good service under the deterministic policy compared to others. It is evident that the deterministic policy is the fairest among all policies. Every service center is idle for the time required to travel over the cycle. In our particular case study, a deterministic policy is optimal in terms of fairness because all sojourn times (i.e. t^s_{ij}) are equal.

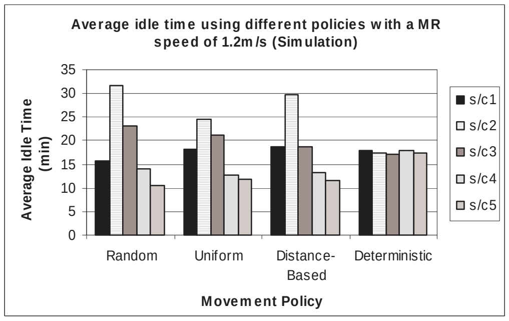
Note that the *Random* policy with a speed of $3.89m/s$ is better than a deterministic policy with a speed of $1.2m/s$. Also, Figure 3.10 shows that our analytical results are so close to those obtained by simulation. As a matter of fact, it turns out that optimizing the movement policy might be better than increasing the number of mobile relays in minimizing the maximum idle time in the network. To verify this, we simulated our case study with different numbers of mobile relays ranging from 1 to 4 under the random (the one shown in Figure 2.3) and the deterministic policies (the one shown in Figure 3.9(b)). For the same speed, as shown in Figure 3.11, the maximum idle time under the deterministic policy with two and three MRs is better than that under the random policy with three and four MRs respectively. Moreover, the maximum idle time under the deterministic policy with two MRs is very close to that under the random policy with four MRs. We conclude from this that the movement policy plays an important role in influencing the performance of the system. Another thing that we conclude from Figure 3.11 is that the effect of movement policy and the speed of movement degrades as the number of MRs increases.



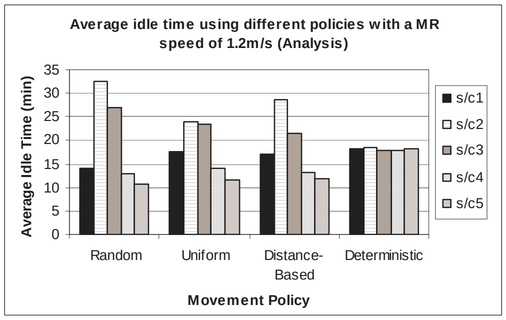
(a) Simulation at a speed of $3.89m/s$



(b) Analysis at a speed of $3.89m/s$



(c) Simulation at a speed of $1.2m/s$



(d) Analysis at a speed of $1.2m/s$

Figure 3.10: The average idle times at all service centers obtained by analysis and simulation using two different speeds, 1.2 and $3.89m/s$.

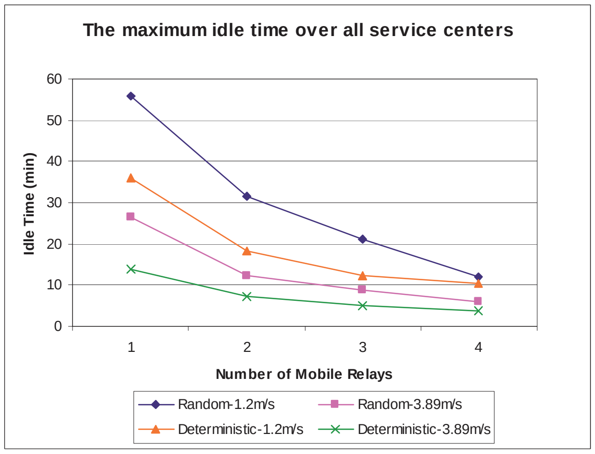


Figure 3.11: The maximum idle time among all service centers using both random and deterministic movement policies.

3.4 Summary

In this chapter we presented a framework for deriving the distribution of the idle time at any service center based on the queueing network model introduced in Chapter 2 (p.15). We used the model to generate numerical results for our case study in Figure 2.3. The comparison between analytical and simulation results showed that our model captures the behavior of the system accurately. Three movement policies were modeled and tested in this chapter, namely, *uniform*, *distance-based*, and *deterministic*. The analytical as well as the simulation results indicate that the movement policy has a significant effect on the idle time. It turns out that in some cases, optimizing the movement policy results in a performance improvement that exceeds that obtained by just adding more mobile relays to the network with unoptimizing movement policy.

CHAPTER 4 END-TO-END DELAY DISTRIBUTION

In this chapter we are interested in predicting the latency that the data might experience while being relayed between fragments before it eventually reaches the base station. We refer to this latency as the *"end-to-end delay"*. In Section 4.1, we introduce the *end-to-end* delay and explain some of the issues related to the evaluation of this delay. We comment on the exact method to evaluate the end-to-end delay in Section 4.2. Then, we propose two approximate approaches in Sections 4.3 and 4.4. In Section 4.5, we show some analytical results and compare them to results obtained by simulation. Finally, we summarizes the chapter in Section 4.6.

4.1 Introduction

In fact, the *end-to-end* delay is the time required to construct a path from a fragment to the base station, and here is how:
First of all, we call the connection between two fragments a segment. In Figure 2.3, the data generated at FRAG-1 will be relayed by an MR at service center s/c_1 to FRAG-2 (segment 1), and then through s/c_5 to FRAG-4 (segment 2). The last step is to relay the data through s/c_3 to the last fragment (segment 3). Then, the data latency is the time required to activate all three segments (i.e., by having mobile relays at the appropriate service centers). In Chapter 3 we derived the idle time distribution at any service center which is equivalent to the segment activation time. In this section we use this distribution to introduce approximate methods to evaluate the *end-to-end* delay. Before

introducing those methods, we first comment on the exact method and its complexity. The main criterion for designing those methods is the level of dependency to assume between the idle time periods of the service centers along the path. Three levels of dependency are used:

(1) *Complete Dependence*: where we take the dependency between all idle times into account, and this is the exact approach,

(2) *Complete Independence*: where we assume that all the idle times are completely independent, and

(3) *Partial Dependence*: where we take the dependency only between a service center and its predecessor on the path.

Assume that we have a path that connects n fragments, named $\{C_1,\ C_2 \ldots,\ C_n\}$, using $n-1$ service centers as shown in Figure 4.1.

If it takes time t to construct the path, then this implies that the time t must be divided

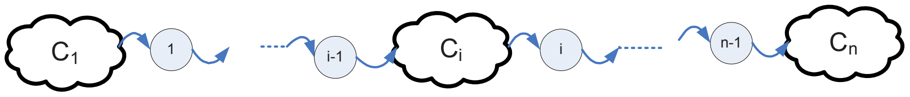


Figure 4.1: A path of n components and $n-1$ service centers

between all the service centers so that each service center is idle for some time between 0 and t. Let that division be $\{t_1, t_2, \ldots, t_{n-1}\}$ such that t_i is the idle time of s/c_i and $\sum_{i=1}^{n-1} t_i = t$, where the idle period of s/c_i starts at the end of the idle period of s/c_{i-1}. If s/c_i is non-idle following the end of t_{i-1}, then t_i must be 0. In this case the data would be relayed directly from fragment $i-1$ to fragment $i+1$. This is shown in Figure 4.2. Moreover, we do not allow t_1 to be zero because we start measuring the delay from the point s/c_1 becomes idle.

Since the understanding of the timing issue here is so critical for the derivations in the next section, we summarize the facts associated with having a path of $n-1$ service centers experience an *end-to-end* delay of duration t:

(1) $t_1 > 0$: the first node on the path must take a non-zero share of the path construction time.

(2) $\sum_{i=1}^{n-1} t_i = t$: the sum of all the shares must equal the path construction time.

(3) $t_i = 0$ implies that the number of MRs at s/c_i at the end of the idle period of s/c_{i-1} is at least one.

(4) $t_i > 0$ implies that there are no MRs at s/c_i at the end of the idle period of s/c_{i-1} and that would be the case for exactly t_i time units.

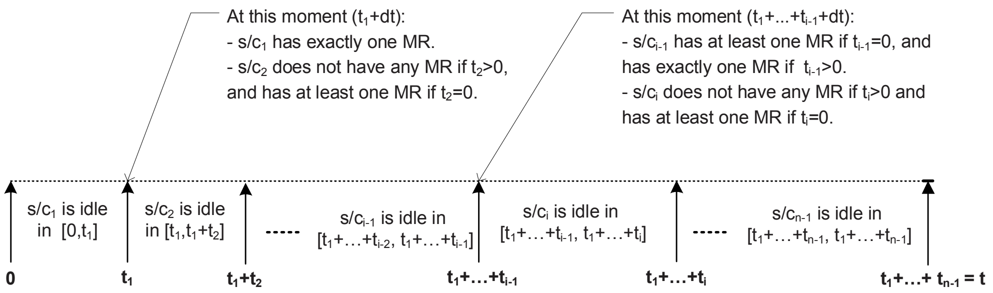

At this moment (t_1+dt):
- s/c_1 has exactly one MR.
- s/c_2 does not have any MR if t_2>0, and has at least one MR if t_2=0.
At this moment (t_1+...+t_{i-1}+dt):
- s/c_{i-1} has at least one MR if t_{i-1}=0, and has exactly one MR if t_{i-1}>0.
- s/c_i does not have any MR if t_i>0 and has at least one MR if t_i=0.
s/c_1 is idle in [0,t_1]
s/c_2 is idle in [t_1,t_1+t_2]
-----
s/c_{i-1} is idle in [t_1+…+t_{i-2}, t_1+…+t_{i-1}]
s/c_i is idle in [t_1+…+t_{i-1}, t_1+…+t_i]
-----
s/c_{n-1} is idle in [t_1+…+t_{n-1}, t_1+…+t_{n-1}]
0
t_1
t_1+t_2
t_1+…+t_{i-1}
t_1+…+t_i
t_1+…+ t_{n-1} = t

Figure 4.2: The timing of the path construction process

4.2 Exact End-to-End Delay

In order to understand the level of complexity of computing the path construction time, we make no approximations or simplifying assumptions regarding the level of dependency between the idle periods in this section. However, we need to simplify the computational complexity by using discrete summation instead of continuous integration. As we have seen in Chapter 3, evaluating the idle time at a service center requires the integration of a matrix exponential which might be complex depending on the size of the matrix. As the evaluation of the end-to-end delay might include the integration of multiple matrix exponentials, we relax the integration to a summation over a short period of length Δ time units. Throughout this chapter we will assume a Δ of $0.01min$. Let us call the service center s/c_i that has $t_i > 0$ an *idle* service center and the one with $t_i = 0$ a *busy* service center. With different combinations of sub-intervals we will have a maximum of $2^{n-1} - 1$ *busy/idle* combinations. Consider the example shown in Figure 4.3 where we have a path of five service centers. For the sake of illustration we picked the sub-intervals to be $(t_1, 0, 0, 0, t_5)$ for service centers $(s/c_1, \ldots, s/c_5)$, respectively. Therefore, the status of the system at $t_1 + dt$ must be:

- s/c_1 has exactly one MR because it has just finished its idle period, and only one transition can take place in a very short time period dt .

- Service centers s/c_2, s/c_3, and s/c_4 have at least one MR each.

- s/c_5 has no customers because it has to start its idle period right at the end of t_1.

We define ζ_i to be the condition that must be satisfied at the end of the idle period (a non-zero period) of s/c_i. In our example:

$$\zeta_1 = (n_2 > 0) \wedge (n_3 > 0) \wedge (n_4 > 0) \wedge (n_5 = 0)$$

$$\zeta_5 = \phi \text{ (no conditions)}$$

Before we talk about the exact method and its complexity we introduce some definitions:

- Let $p_i(t, \zeta_i)$ be the probability that s/c_i is idle for time t, and at the end of that idle period the system will be in a state in which the condition ζ_i is satisfied.

- Let $\mathcal{F}$ be a path of n service centers $\{s/c_1, ..., s/c_n\}$, and let $|\mathcal{F}|$ be the length of that path.

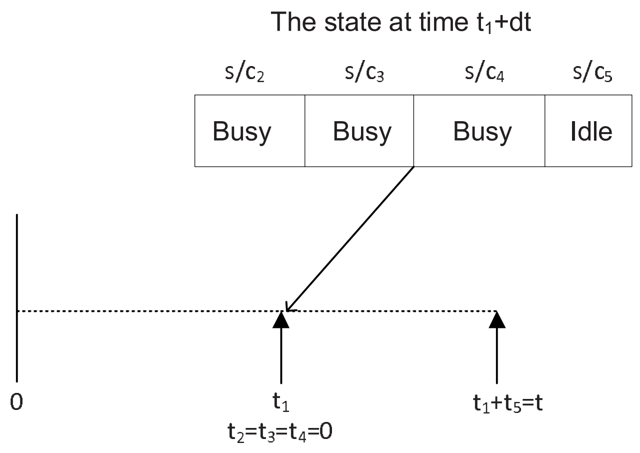


Figure 4.3: A sample *busy/idle* combination that might occur in a path of five service centers.

- Let $\psi(\digamma, t)$ be the probability mass function (*pmf*) of having the path $\digamma$ constructed in time t, i.e., the end-to-end delay *pmf*.

Then, $\psi(\digamma, t)$ can be evaluated as follows:

(1) Divide t over the $|\digamma|$ nodes, and specify the conditions at the end of every idle period as we did in the previous example.

(2) Evaluate

$$\prod_{\substack{i=1\\t_i>0}}^{|\digamma|} p_i(t_i, \zeta_i|\varsigma_i \text{ } at\ the\ beginning\ of\ t_i) \tag{4.1}$$

where ς_i is a condition defined as:

$$\varsigma_i = \begin{cases} \zeta_{i-1} & \text{if } i > 1 \\ n_i = 0 & \text{if } i = 1 \end{cases} \tag{4.2}$$

(3) Sum (4.1) over all possible sub-intervals.

Apparently, this method is computationally intractable because of ζ_i. Note that in the worst case we need to consider the state of the whole path and that gets worse as the length of the path gets longer, since we will have to consider the state of the whole queuing system. Therefore, we propose two approximations by relaxing ζ_i. First, we assume independence between the idle sub-intervals, i.e. $\zeta_i = \phi$. Second, we consider the dependence only between two consecutive service centers along the path, i.e., ζ_i conditions on a maximum of one service center only.

4.3 Approximation-I: Convolution (CONV)

Let T_i be a random variable that represents the idle time at s/c_i, and let X_j be a random variable that represents the construction time of path j. Then, $X_j = \sum_{i \in j} T_i$. If we assume that all the T_i's are independent, then the probability distribution function of X_j is the convolution of the all the distribution functions of the T_i's [44]. Therefore, using $\digamma$ and $\psi(\digamma, t)$ defined in the previous section, as well as $F_{T_i|I}(t)$ and $F_{T_i}(t)$ defined in Chapter 3 we have:

$$\psi(\digamma, t) = F_{T_1|I} \circledast F_{T_2} \cdots \circledast F_{T_n}(t) \tag{4.3}$$

Note that in equation (4.3) we used $F_{T_1|I}$ for the first service center on the path, and that is again because we do not allow the idle time at that service center to be zero. This approximation significantly reduces the complexity by ignoring the dependence between the idle periods. However, assuming independence underestimates the *end-to-end* delay since the presence of a MR at previous service centers is ignored, i.e. the probability of being idle/busy at any service center is evaluated without considering the status of previous service centers along the path. We will revisit this issue in Section 4.5 when we compare the *end-to-end* delay obtained using the convolution method and that obtained using the dynamic programming approach we propose in the next section.

4.4 Approximation-II: Dynamic Programming-Like Approach (DPA)

The problem of evaluating the distribution of the *end-to-end* delay possesses a trade-off between the accuracy of the solution and its complexity. The exact solution comes at the cost of high computational complexity. The convolution method, on the other hand, comes at the cost of low accuracy but with significantly lower computational complexity. In this section we propose a dynamic programming-like approach (DPA) which can be regarded as a compromise between the computational complexity and solution accuracy. This compromise is based on the relaxation of the level of dependency between idle periods, i.e., by only considering the dependency between every pair of consecutive service centers.

Since any service center will be in one of two different states (i.e., *busy* or *idle*) at any point in time, we have four different situations:

(1) s/c_i is *idle* for time t given that s/c_j was also *idle*; let the probability of this event be $p_{I|I}(i,j,t)$,

(2) The current service center (s/c_i) is *idle* for time t given that the previous service center (s/c_j) was *busy*; let the probability of this event be $p_{I|B}(i,j,t)$,

(3) s/c_i is *busy* given that s/c_j was *idle*; let the probability of this event be $p_{B|I}(i,j)$, and

(4) s/c_i is *busy* given that s/c_j was *busy*; let the probability of this event be $p_{B|B}(i,j)$.

Next we derive the formulas for all the four conditional probabilities listed above.

4.4.1 Conditional Probabilities

Let us start with the probability $p_{I|I}(i,j,t)$. Note that for the first service center on the path, $p_{I|I}(i,j,t) = F_{T_i|I}(t)$ (equation (3.14)). For the other service centers, it is the same as $F_{T_i|I}(t)$ except for the initial condition $\overrightarrow{p}^{\,i}_{idle}(0)$ (see equations (3.6) and (3.9)) where we have a different condition to be satisfied which is: s/c_j has just ended its idle period, i.e., n_j changed from 0 to 1 and hence $n_j = 1$ at the beginning of t_i. We refer to the probability of this scenario for a certain network state $\overrightarrow{N}$ as $h^i_{idle}(0, \overrightarrow{N}|n_j{=}1\ at\ t{=}0)$. This probability is given by:

$$h^i_{idle}(0, \overrightarrow{N}|n_j{=}1\ at\ t{=}0) = \begin{cases} \dfrac{prob(\overrightarrow{N}, n_j(\overrightarrow{N}){=}1\ at\ t{=}0)}{prob(n_j{=}1\ at\ t{=}0)} & \text{if } n_j(\overrightarrow{N}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $prob(\overrightarrow{N}, n_j(\overrightarrow{N}){=}1\ at\ t{=}0)$ is given by,

$$prob(\overrightarrow{N}, n_j(\overrightarrow{N}){=}1,\ at\ t{=}0) = \sum_{\substack{k=1\\k\neq j}}^{M} \frac{n_k(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)\mu_k}{\sum_{l=1}^{M}\mu_l n_l(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)} q_{kj}\pi(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j) \tag{4.5}$$

We derived equation (4.5) following the same approach we used to derive equation (3.9). We take the probability that the transition from queue k to queue j, that will lead to state $\overrightarrow{N}$, takes place before any other transition. The probability $prob(n_j{=}1,\ at\ t{=}0)$ is given as,

$$prob(n_j{=}1,\ at\ t{=}0) = \sum_{\substack{\overrightarrow{N}^*\in\Omega\\ n_j(\overrightarrow{N}^*)=1}} prob(\overrightarrow{N}^*, n_j(\overrightarrow{N}^*){=}1\ at\ t{=}0) \tag{4.6}$$

Using equations (4.5) and (4.6) we obtain,

$h^i_{idle}(0, \overrightarrow{N}|n_j{=}1 \ at \ t{=}0) =$

$$\begin{cases} \dfrac{\displaystyle\sum_{\substack{k=1\\k\neq j}}^{M} \frac{n_k(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)\mu_k}{\sum_{l=1}^{M}\mu_l n_l(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)} q_{kj}\pi(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)}{\displaystyle\sum_{\substack{\overrightarrow{N}^*\in\Omega\\ n_j(\overrightarrow{N}^*)=1}} \sum_{\substack{k=1\\k\neq j}}^{M} \frac{n_k(\overrightarrow{N}^*+\overrightarrow{1}_k-\overrightarrow{1}_j)\mu_k}{\sum_{l=1}^{M}\mu_l n_l(\overrightarrow{N}^*+\overrightarrow{1}_k-\overrightarrow{1}_j)} q_{kj}\pi(\overrightarrow{N}^*+\overrightarrow{1}_k-\overrightarrow{1}_j)} & \text{if } n_j(\overrightarrow{N})=1 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

Note that, even though the parameter i does not appear in equation (4.7), it implies that $h^i_{idle}(0, \overrightarrow{N}|n_j{=}1 \ at \ t{=}0)$ is defined only over network states in which $n_i(\overrightarrow{N}) = 0$. Let,

$$\overrightarrow{h}^i_{idle}(0|n_j{=}1) = [\overrightarrow{h}^i_{idle}(0, \overrightarrow{N}_1|n_j{=}1 \ at \ t{=}0), \ldots, \overrightarrow{h}^i_{idle}(0, \overrightarrow{N}_{E_i}|n_j{=}1 \ at \ t{=}0)]^T \tag{4.8}$$

Then, using equations (3.6) and (4.8) we get,

$$p_{I|I}(i,j,t) = \overrightarrow{U}_i(-\mathbf{A}_i)e^{\mathbf{A}_i t}\overrightarrow{h}^i_{idle}(0|n_j{=}1) \tag{4.9}$$

For $p_{I|B}(i,j,t)$, we do not know what the last transition was because we consider the dependency between consecutive queues only. But, we know that the initial state must have s/c_j busy, i.e. $n_j > 0$, and $n_i = 0$. Therefore, we approximate the probability of the initial state, which we refer to it as $h^i_{idle}(0, \overrightarrow{N}|n_j > 0 \ at \ t{=}0)$, using the steady state probabilities. Equation 4.10 defines $h^i_{idle}(t, \overrightarrow{N}|n_j > 0 \ at \ t{=}0)$.

$$h^i_{idle}(0, \overrightarrow{N}|n_j > 0 \ at \ t{=}0) = \begin{cases} \dfrac{\pi(\overrightarrow{N})}{\displaystyle\sum_{\substack{\overrightarrow{N}^*\in\Omega\\ n_j(\overrightarrow{N}^*)>0}} \pi(\overrightarrow{N}^*)} & \text{if } n_j(\overrightarrow{N}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

And again $h^i_{idle}(0,\overrightarrow{N}|n_j>0\ at\ t{=}0)$ is defined over network states in which $n_i=0$. Let,

$$\overrightarrow{h}^i_{idle}(0|n_j{>}0) = [\overrightarrow{h}^i_{idle}(0,\overrightarrow{N}_1|n_j{>}0\ at\ t{=}0),\ldots,\overrightarrow{h}^i_{idle}(0,\overrightarrow{N}_{E_i}|n_j{>}0\ at\ t{=}0)]^T \quad (4.11)$$

Then, using equations (3.6) and (4.11) we get,

$$p_{I|B}(i,j,t) = \overrightarrow{U}_i(-\mathbf{A}_i)e^{\mathbf{A}_i t}\overrightarrow{h}^i_{idle}(0|n_j>0) \quad (4.12)$$

The third conditional probability is $p_{B|I}(i,j)$. The condition that must be satisfied at the initial state for $p_{B|I}(i,j)$ is the same as that for $p_{I|I}(i,j,t)$, which is that s/c_j has just ended its idle period but this time finding s/c_i busy, i.e., $n_j{=}1$ and $n_i{>}0$. We define $h^i_{busy}(0,\overrightarrow{N}|n_j{=}1\ at\ t{=}0)$ as the probability of having the initial (at time 0) state $\overrightarrow{N}$ in which $n_i>0$ given that s/c_j has just ended its idle period, i.e., $n_j{=}1$, and $n_i{>}0$. Therefor,

$h^i_{busy}(0,\overrightarrow{N}|n_j{=}1\ at\ t{=}0) =$

$$\begin{cases} \dfrac{\displaystyle\sum_{\substack{k=1\\k\neq j}}^{M} \frac{n_k(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)\mu_k}{\displaystyle\sum_{l=1}^{M}\mu_l n_l(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)} q_{kj}\pi(\overrightarrow{N}+\overrightarrow{1}_k-\overrightarrow{1}_j)}{\displaystyle\sum_{\substack{\overrightarrow{N}^*\in\Omega\\ n_j(\overrightarrow{N}^*)=1}}\sum_{\substack{k=1\\k\neq j}}^{M} \frac{n_k(\overrightarrow{N}^*+\overrightarrow{1}_k-\overrightarrow{1}_j)\mu_k}{\displaystyle\sum_{l=1}^{M}\mu_l n_l(\overrightarrow{N}^*)} q_{kj}\pi(\overrightarrow{N}^*+\overrightarrow{1}_k-\overrightarrow{1}_j)} & \text{if } n_j(\overrightarrow{N})=1 \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

This time $h^i_{busy}(0,\overrightarrow{N}|n_j{=}1\ at\ t{=}0)$ is defined only over network states in which $n_i>0$. Equation (4.14) gives the final expression for $p_{B|I}(i,j)$.

$$p_{B|I}(i,j) = \sum_{\substack{\overrightarrow{N}\in\Omega\\ n_i(\overrightarrow{N})>0}} h^i_{busy}(0,\overrightarrow{N}|n_j{=}1\ at\ t{=}0) \quad (4.14)$$

The last conditional probability to find is $p_{B|B}(i,j)$. Let $h^i_{busy}(0,\overrightarrow{N}|n_j{>}0\ at\ t{=}0)$ be the probability of having the initial (at time 0) state $\overrightarrow{N}$ in which $n_i{>}0$ given that s/c_j

is busy (i.e. $n_j{>}0$). Then, using the same approximation we made to derive equation (4.10) we get,

$$h^i_{busy}(0,\overrightarrow{N}|n_j{>}0\ at\ t{=}0) = \begin{cases} \dfrac{\pi(\overrightarrow{N})}{\sum\limits_{\substack{\overrightarrow{N}^*\in\Omega \\ n_j(\overrightarrow{N}^*)>0}} \pi(\overrightarrow{N}^*)} & \text{if } n_j(\overrightarrow{N}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

Also, in this case, $h^i_{busy}(0,\overrightarrow{N}|n_j{>}0\ at\ t{=}0)$ is defined over network states in which $n_i{>}0$. Equation (4.16) gives the final expression for $p_{B|B}(i,j)$.

$$p_{B|B}(i,j) = \sum_{\substack{\overrightarrow{N}\in\Omega \\ n_i(\overrightarrow{N})>0}} h^i_{busy}(0,\overrightarrow{N}|n_j{>}0\ at\ t{=}0) \tag{4.16}$$

4.4.2 Recursive Approach for Calculating the *pmf* for the End-to-End Delay

Before we get into the details of our dynamic program, we start with some definitions.

- As defined before, $\digamma$ is a path of service centers.

- υ is used to index the service center over the path, i.e., $\digamma(\upsilon)$ is the υ^{th} service center along the path.

- φ is the status of a service center: $\varphi = 0$ means that the service center has just ended its idle period, $\varphi = 1$ means a busy service center, and $\varphi = 2$ means that the service center has started an idle period.

- $\alpha(t,\digamma,\upsilon,\varphi)$ is the probability that the υ^{th} service center along the path $\digamma$ is idle for time t given that its predecessor (i.e. $(\upsilon-1)^{th}$ service center) was in a status φ.

- $\gamma(\digamma,\upsilon,\varphi)$ is the probability that the υ^{th} service center along the path $\digamma$ is busy given that its predecessor (i.e. $(\upsilon-1)^{th}$ service center) was in a status φ.

- $\psi(t, \digamma, \upsilon, \varphi)$ is the probability that the *end-to-end* delay along the path $\digamma$ is t given that service center s/c_υ was in a status φ. Therefore, $\psi(\digamma, t) = \psi(t, \digamma, \digamma(1), 2)$.

Using the conditional probabilities we obtained earlier (viz., equations (4.9), (4.12), (4.14), and (4.16)) we express $\alpha(t, \digamma, \upsilon, \varphi)$ and $\gamma(\digamma, \upsilon, \varphi)$ as follows:

$$\alpha(t, \digamma, \upsilon, \varphi) = \begin{cases} p_{I|I}(\digamma(\upsilon), \digamma(\upsilon-1), t) & \text{if } \varphi\text{=0 and } \upsilon\text{>1} \\ p_{I|B}(\digamma(\upsilon), \digamma(\upsilon-1), t) & \text{if } \varphi\text{=1 and } \upsilon\text{>1} \\ f_{T_{\digamma(\upsilon)}|I}(t) & \text{if } \varphi\text{=2 and } \upsilon\text{=1} \end{cases} \tag{4.17}$$

$$\gamma(\digamma, \upsilon, \varphi) = \begin{cases} p_{B|I}(\digamma(\upsilon), \digamma(\upsilon-1)) & \text{if } \varphi = 0 \text{ and } \upsilon > 1 \\ p_{B|B}(\digamma(\upsilon), \digamma(\upsilon-1)) & \text{if } \varphi = 1 \text{ and } \upsilon > 1 \\ 0 & \text{if } \upsilon = 1 \end{cases} \tag{4.18}$$

The basic idea of the dynamic program is that each service center has two possible states, either it is busy (i.e., has at least one MR) or idle. The recursive step of the dynamic programming-like approach for the evaluation of $\psi(t, \digamma, \varphi, \upsilon)$ is given by equation (4.19).

$$\psi(t, \digamma, \varphi, \upsilon) = \gamma(\digamma, \upsilon, \varphi)\psi(t, \digamma, 1, \upsilon+1) + \sum_{k=1}^{t/\Delta} \alpha(k\Delta, \digamma, \upsilon, \varphi)\psi(t-k\Delta, \digamma, 0, \upsilon+1)\Delta \tag{4.19}$$

The term, $\gamma(\digamma, \upsilon, \varphi)\psi(t, \digamma, 1, \upsilon+1)$, is the probability that the current service center, viz.,s/c_υ, is busy and the rest of the path is idle for time t, and this is why the entire idle time t should be incurred over all downstream service centers. Moreover, we pass $\varphi = 1$ in the recursion so that in the following step in the recursion, the previous service center will be known as a busy center. The second term, $\sum_{k=1}^{t/\Delta} \alpha(k\Delta, \digamma, \upsilon, \varphi)\psi(t-k\Delta, \digamma, 0, \upsilon+1)\Delta$, is the probability that the current service center, viz., υ, is idle for time $k\Delta$ and an idle time of $t - k\Delta$ is incurred over the rest of the path.

As illustrated at the beginning of Section 4.2, Δ is used to transform the problem into a discrete form in order to simplify the numerical evaluation. Equation (4.20) shows the boundary conditions for the recursive formula above. For the case of $\upsilon = |\digamma|$, the equation returns one of two different values depending on the remaining time.

$$\psi(t, \digamma, \varphi, \upsilon) = \begin{cases} \gamma(\digamma, \upsilon, \varphi) & \text{if } \upsilon = |\digamma| \text{ and } t = 0 \\ \alpha(t, \digamma, \upsilon, \varphi) & \text{if } \upsilon = |\digamma| \text{ and } t > 0 \end{cases} \tag{4.20}$$

4.5 Results and Discussions

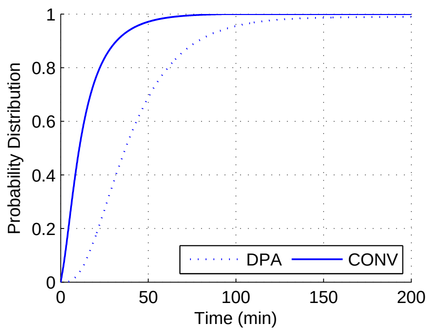
Figure 4.4 shows $\psi(t, \digamma)$ for all data paths in the network shown in Figure 2.3 obtained using both CONV and DPA approaches using two different speeds, 1.2 and $3.89m/s$. As this figure shows, the CONV approach overestimates the probability distribution function and therefore underestimates the average *end-to-end* delay. For instance, the probability that the path $\{1-5-3\}$ experiences an end-to-end delay that is less than or equal $20min$ at a speed of $1.2m/s$ is about 0.95 using the CONV approach and about 0.7 using the DPA approach. Figure 4.5 shows the average *end-to-end* delay for all the data paths in our case-study with the network parameters in Tables 2.3 and 2.2 obtained using the distributions in Figure 4.4 as well as TOSSIM/Tython simulation. The results shown in Figure 4.5 that the DPA approach predicts the *end-to-end* delay accurately. On the other hand, the CONV approach does not work well and it underestimated the average *end-to-end* delay by up to 66%, which makes this approach inappropriate for evaluating the *end-to-end* delay. However, the CONV approach will be useful to predict the *end-to-end* delay in large networks as we show in Chapter 5. Figure 4.5 also shows that increasing the speed from $1.2m/s$ to $3.89m/s$ reduces the end-to-end delay by an average of about 60%.

Figure 4.6 shows the effect of Δ on the accuracy and running time of the DPA approach. Figure 4.6(b) implies that varying Δ between 0.01 and 1 does not crucially affect the accuracy. For instance, the *end-to-end* delay evaluated at $\Delta = 1$ is just 2% less than that evaluated at $\Delta = 0.0125$. On the other hand, the running time, as

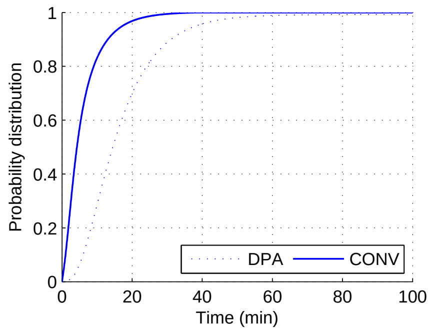
Figure 4.6(a) shows, increases exponentially as Δ decreases. It went from 0.9228 *min* at $\Delta = 1min$ to 109.25 *min* at $\Delta = 0.0125min$. Therefore, we can increase the value of Δ to reduce the running time without significantly degrading the accuracy.

4.6 Summary

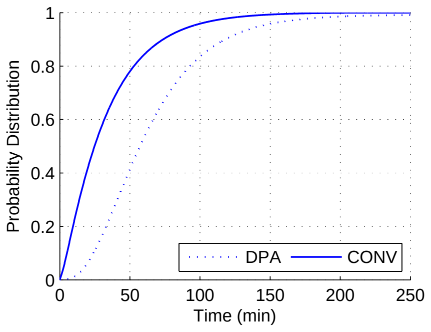
In this chapter, we proposed two approximate methods to evaluate the *end-to-end* delay. The first one is the convolution approach in which idle times at different service centers are assumed to be independent. This assumption overestimated the probability distribution and consequently underestimated the average *end-to-end* delay. The second method is the *Dynamic Programming-Like* approach. This approach considers the dependency only between a pair of consecutive service centers. The results show that this approach achieves high accuracy.



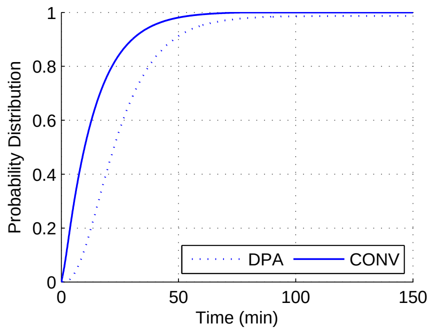
(a) Path {1,5,3} at a speed of 1.2m/s



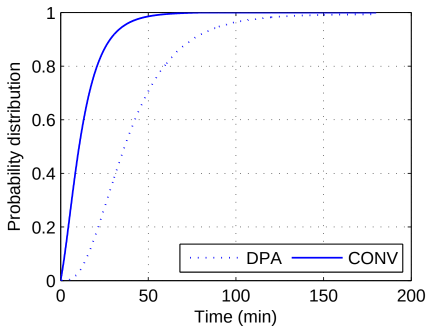
(b) Path {1,5,3} at a speed of 3.89m/s



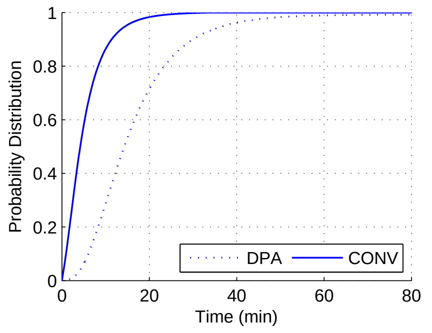
(c) Path {2,5,3} at a speed of 1.2m/s



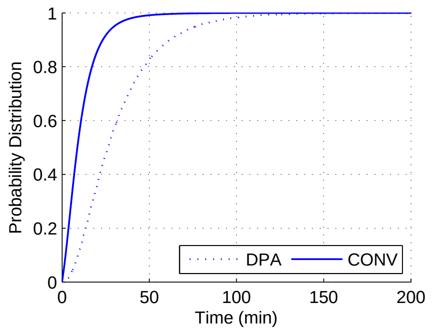
(d) Path {2,5,3} at a speed of 3.89m/s



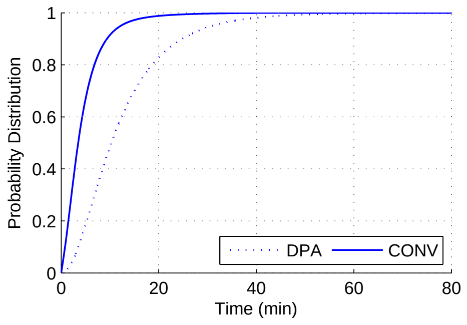
(e) Path {4,5,3} at a speed of 1.2m/s



(f) Path {4,5,3} at a speed of 3.89m/s

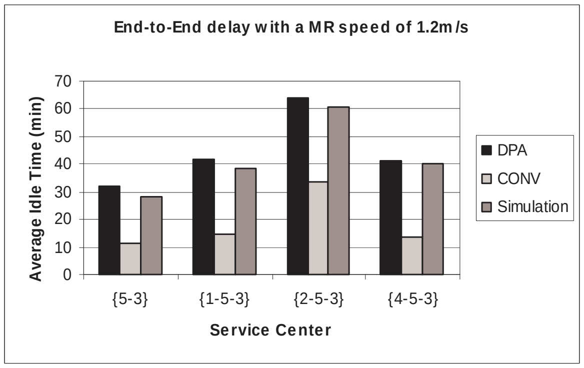


(g) Path {5,3} at a speed of 1.2m/s

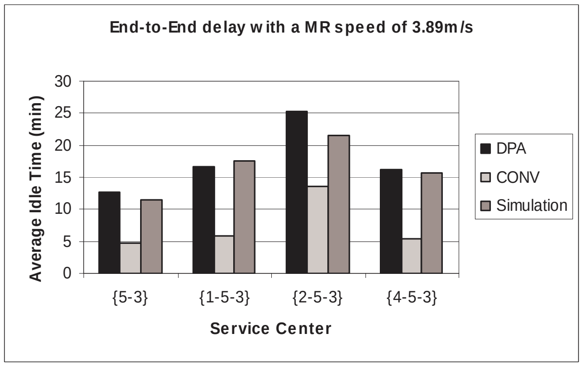


(h) Path {5,3} at a speed of 3.89m/s

Figure 4.4: The *end-to-end* delay probability distribution for all data paths in the case-study network of Figure 2.3 and Tables 2.2 and 2.3.

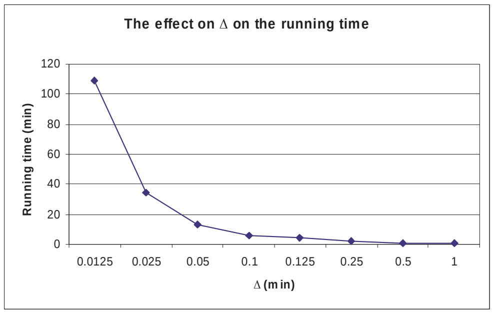


(a) The average *end-to-end* delay at a speed of $1.2 m/s$

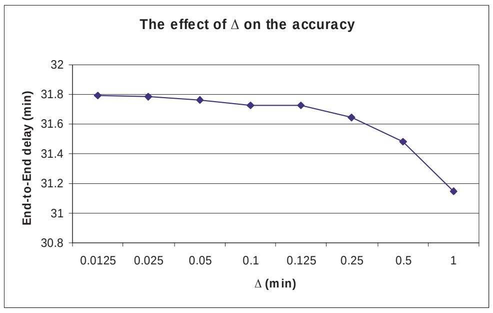


(b) The average *end-to-end* delay at a speed of $3.89 m/s$

Figure 4.5: The average *end-to-end* delay evaluated using convolution (CONV) and Dynamic programming approach (DPA), and its validation by simulation at two speeds, 1.2 and $3.89 m/s$



(a) Running time of the DPA for the path {5,3} with different values of Δ



(b) Average end-to-end delay along the path {5,3} using different values of Δ

Figure 4.6: The effect of Δ on the running time and accuracy of the DPA approach.

CHAPTER 5 PRACTICAL CONSIDERATIONS

In this chapter we address two important issues associated with the data delivery problem defined earlier. The first issue how to control MR mobility in FWSNs that extend over a large area. Such level of coverage might increase the distances that MRs need to travel. This increase results in longer idle times at service centers which means larger *end-to-end* delays. To deal with this scenario, we propose to divide the whole network into smaller partitions (i.e., groups of service centers) and deploy MRs locally in each partition. The second issue the computation of the sojourn time (t^s_{ij}) at a service center. In all previous analysis and simulations we used the same t^s_{ij} at all service centers. However, different service centers might need different sojourn times depending on the amount of generated data at a fragment as well as the amount of data relayed into this fragment. Based on the queueing model proposed in Chapter 2, we estimate the total amount of data (generated and relayed) that a fragment might hold. Using this estimation, we propose an iterative algorithm to compute t^s_{ij} for a service center.

5.1 Mobility Control in Large Networks

In large systems, like the one shown in Figure 5.1, two possible schemes to utilize mobile relays may be used. First, to let MRs patrol the whole network. Second, to divide the field into regions and assign a group of MRs to each region so that MRs need to patrol each region locally instead of patrolling the whole field. In this section we evaluate the performance of both schemes using a deterministic movement policy. But first, we need to answer a fundamental question: *"How to divide the field"?* For this purpose, we define two criteria to divide a network field of size A into n regions:

(1) *Area-Based*: divide the whole field into n regions, each of which is of area $\dfrac{A}{n}$. Then, the available MRs are distributed over the n regions based on the number of service centers in each region. We refer to the service centers that belong to a certain region a group. This approach is simple, and is reasonable if the network is dense and sensors are uniformly deployed.

(2) *Proximity-Based*: define a threshold distance d_{thresh}. Then, build a graph that contains all service centers and has no edges. For every pair of service centers that are less than d_{thresh} apart, extend an edge between them so that the graph will be divided into a number of connected subgraphs depending on the value of d_{thresh}. We call a the service centers that belong to one connected subgraph a group. The next step is to distribute the available number of mobile relays over the connected subgraphs based on the number of service centers in each subgraph, the larger the size of the subgraph the greater the number of mobile relays assigned to it. However, lower and upper bounds on the number of subgraphs can be achieved by tuning d_{thresh}.

There are two important issues regarding the distribution of MRs over regions. First, a region (or a graph) of one or two service centers should not receive more than one mobile relay according to the definition of the data delivery problem in FWSNs we introduced in Chapter 2. Second, if we are left with one MR, then the priority of assigning an MR to a region is given for the region that is spread over a larger area, i.e., the diameter of a group.

We study the network shown in Figure 5.1 with six mobile relay nodes with the network parameters shown in Table 5.1. Figure 5.2 shows the area-based devision of the network in Figure 5.1. Group A, which has four service centers, receives two MRs, group

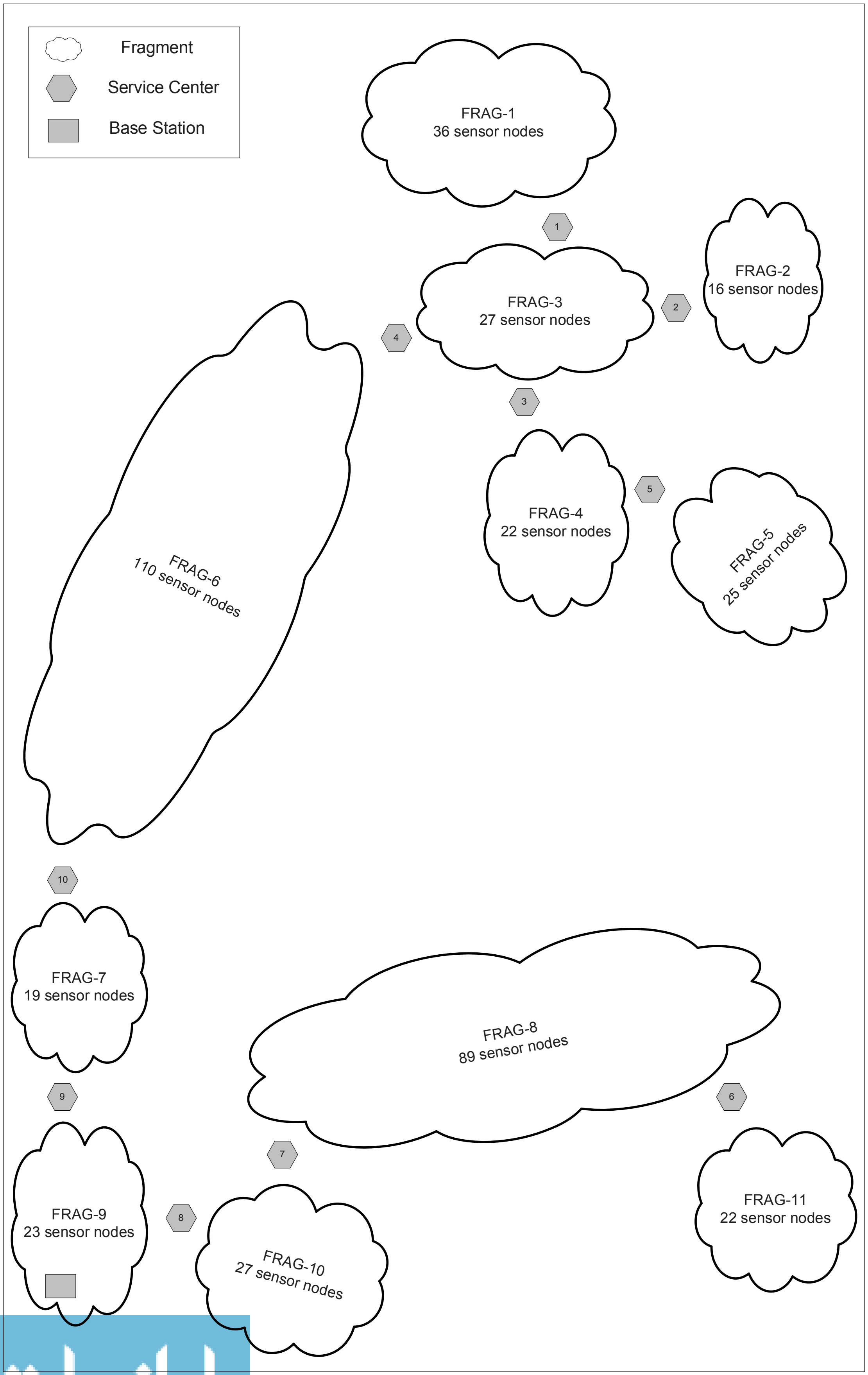


Figure 5.1: A large topology with (11) fragments and (10) service centers.

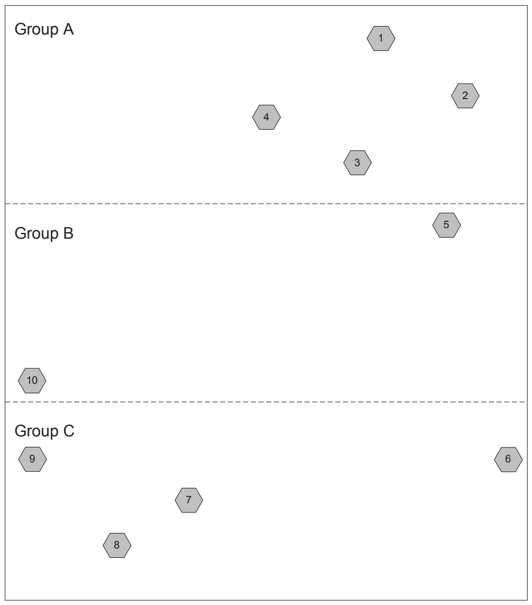


Figure 5.2: Area-Based Division

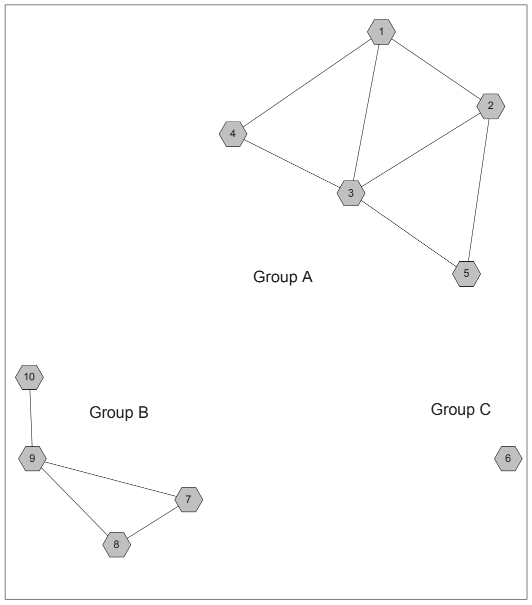


Figure 5.3: Proximity-Based Division

Parameter	Value	Parameter	Value
Num. of MRs	6	$d_{4,10}$	$1507.688m$
Num. of s/c's	10	$d_{5,6}$	$1451.9m$
Speed (L)	1.2 *and* $3.89m/s$	$d_{5,10}$	$1670.039m$
t^s_{ij}	$2min$	$d_{6,8}$	$1335.9m$
$d_{1,2}$	$340.032m$	$d_{7,8}$	$283.04m$
$d_{1,3}$	$421.472m$	$d_{7,9}$	$538.592m$
$d_{1,4}$	$462.88m$	$d_{7,10}$	$832.416m$
$d_{2,3}$	$422.912m$	$d_{8,9}$	$401.6m$
$d_{2,5}$	$434.624m$	$d_{3,4}$	$339.104m$
$d_{2,6}$	$1571.6m$	$d_{3,5}$	$361.92m$

Table 5.1: Parameters of the network shown in Figure 5.1

B, which has two service centers, receives one MR, and group C, which has four service centers, receives three MRs. Note that group C and group A have the same number of service centers, but group C received the sixth MR because it spreads over a larger area. On the other hand, Figure 5.3 shows a proximity based division of this network with $d_{thresh} = 550m$. In this devision, group A receives three MRs, group B receives two MRs, and group C receives one MR.

We simulated all three scenarios using deterministic movement policies with two different speeds of MRs namely, $1.2m/s$ and $3.89m/s$. As Figure 5.4 shows, the average idle time at all service centers using the *Proximity-Based* approach is upper bounded by that obtained using the *Whole-Network* approach. For instance, the idle time at service centers $\{1,\ 2,\ 3,\ 4,\ and\ 5\}$ is reduced by an average of 42% using the *Proximity-Based* from that using the *Whole-Network* approach at a speed of $1.2m/s$. For a speed of $3.89m/s$, the reduction is 35%, as shown in Figure 5.5, which implies that the benefit from dividing the network decreases as the speed of movement increases.

The *Area-Based* approach has the disadvantage that a small number of service centers might be scattered over a large area. Group B in Figure 5.2 is an example of such scenario. The simulation results in Figures 5.4 and 5.5 show that service centers s/c_5 and s/c_{10} experience very high idle times compared to other service centers under the *Area-Based* division. This is because MRs were assigned on the number of service centers in a group without considering the distances between the service centers. However, s/c_5 and s/c_{10} receive good service under the other two approaches.

Figures 5.6 and 5.7 show the average and maximum idle times over all service centers in the network. It is evident that the *Area-Based* approach performs the worst, whereas the *Proximity-Based* approach outperforms all others. The average is less, under the *Proximity-Based* approach, by 27% at $3.89m/s$ and 33% at $1.2m/s$ than that under the *Whole-Network* approach. The average idle time at s/c_6 is zero because a MR was assigned to this service center alone.

Figure 5.8 shows the end-to-end delay, for data generated at all fragments, using all three approaches. It is evident that the *proximity-based* approach performs better than the *whole-network* approach for some fragments, $FRAG-1$ through $FRAG-6$, and both approaches were roughly equally efficient for the remaining fragments. However, the area-based approach is the worst among all the others.

Figure 5.9 compares the end-to-end delay for four fragments, $FRAG-1$ through $FRAG-4$, obtained by simulation to that obtained by analysis. To estimate the end-to-end delay in a divided network, we use the DPA approach proposed in Chapter 4 within the same group, and the CONV approach between different groups. In other words, the delay in each group is evaluated using the DPA approach, and then the delay distributions of all groups that form a path to the sink are convolved to get the total end-to-end delay. This method had an average error of 22.6% even though the error was only about 3% for $FRAG - 2$.

5.2 Engineering the Service Time

The assumption that t^s_{ij} is the same for all service centers is not a practical one, since different fragments with different sizes and different locations in the network require different service times. This time depends on three parameters of the source-fragment:

(1) The size of the fragment, in terms of the number of sensor nodes,

(2) The average data generation rate by each sensor within the fragment, and

(3) The average amount of relayed data that will be temporarily buffered at the fragment.

Before we get into details, we first introduce some definitions. For a fragment i, let:

- S_i be the number of sensor nodes in fragment i.
- ρ_i be the data generation rate in $bits/sec$ of a sensor in fragment i.

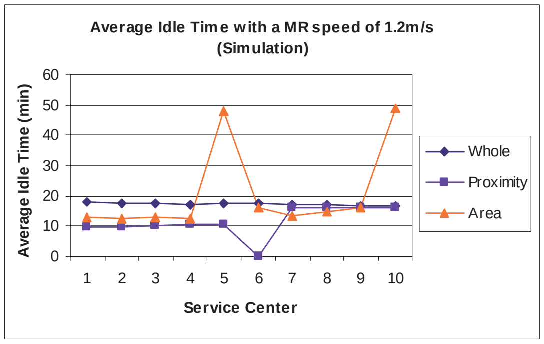


Figure 5.4: Average idle time (speed=1.2m/s)

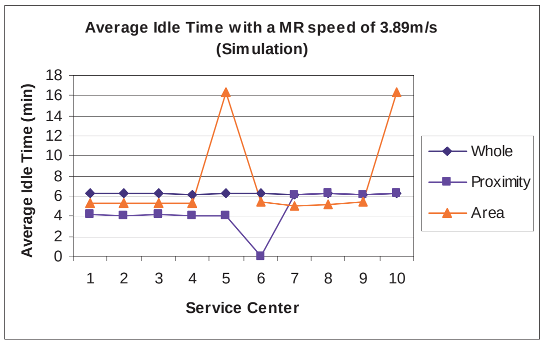


Figure 5.5: Average idle time (speed=3.89m/s)

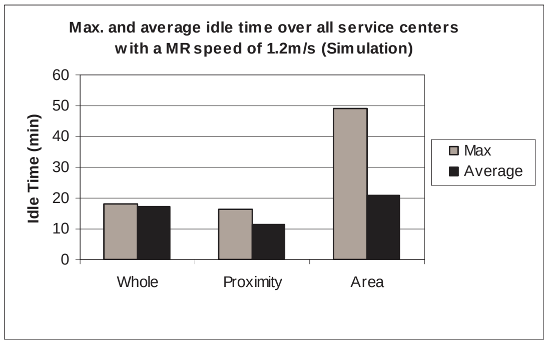


Figure 5.6: Maximum and average idle time (speed=1.2m/s)

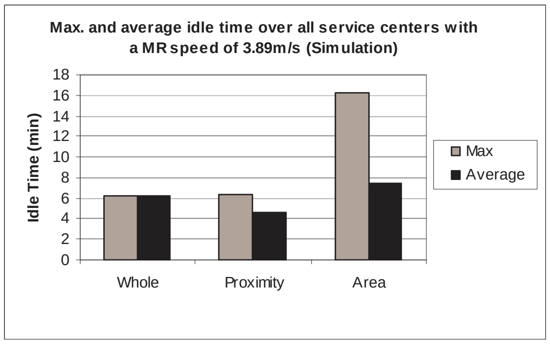


Figure 5.7: Maximum and average idle time (speed=3.89m/s)

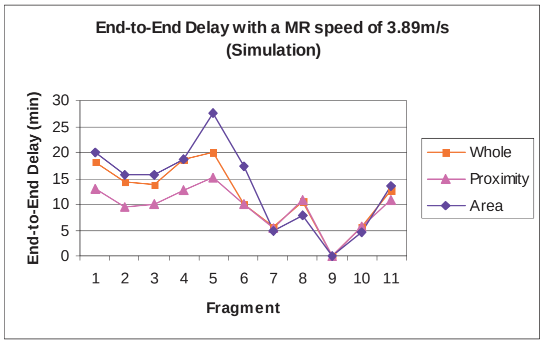


Figure 5.8: End-to-end delay (speed=3.89m/s)

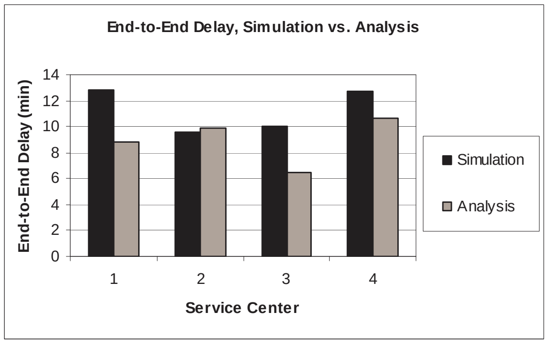


Figure 5.9: End-to-end delay obtained by analysis and simulation (speed=3.89m/s)

- R be the sensor's data transmission rate in $bits/sec$.
- ς_i be the average amount of relayed data, in bits, that will be temporarily buffered in fragment i.
- σ_i be the average total amount of data (generated and relayed) that is temporarily buffered in fragment i.
- H_i is a set of all fragments that $FRAG{-}i$ lies on their data paths toward the sink. For example, in our case study in Figure 2.3, H_6=$\{1,2,3,4,5\}$, H_3=$\{1,2,4,5\}$, H_2=$\{1,4,5\}$, and H_1=H_4=H_5=$\{\}$.

For a service center s/c_k, let:

- $s(k)$ be the source fragment that an MR at s/c_k relays data from. For example, in Figure 2.3, $s(1) = 1$, $s(2) = 5$, $s(3) = 3$, $s(4) = 4$, and $s(5) = 2$.
- $d(k)$ be the destination fragment that an MR at s/c_k relays data to. For example, in Figure 2.3, $d(1) = 2$, $d(2) = 2$, $d(3) = 6$, $d(4) = 2$, and $d(5) = 3$.

By considering the idle time and the following sojourn time at s/c_k to form an alternating renewal process, equation (5.1) evaluates the average of the total amount of data relayed into fragment $s(k)$ by summing the data generation rates at all the fragments whose data is relayed through our $s(k)$ during the average idle time at s/c_k as well as the sojourn time.

$$\varsigma_{s(k)} = \begin{cases} \left(t^s_{s(k),d(k)} + E[T_k|I]\right) \sum_{i \in H_{s(k)}} S_i \rho_i & \text{if } |H_{s(k)}| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Then, equation (5.2) is used to estimate the average amount of data that might be buffered at a fragment, relayed into and generated within the fragment.

$$\sigma_{s(k)} = \left(t^s_{s(k),d(k)} + E[T_k|I]\right) S_{s(k)} \rho_{s(k)} + \varsigma_{s(k)} \tag{5.2}$$

We propose an algorithm that rely on this approximation to calculate t^s_{ij}. Algorithm 5.1 first initializes t^s_{ij}, and then solves the closed queueing network and calculates the average idle time at all service centers using the equation (3.17). The next step is to calculate new t^s_{ij}'s based on the average amount of buffered data evaluated using equation (5.2). This process is repeated until all the differences between the new values of t^s_{ij} and the old ones are less than a predetermined threshold.

Algorithm 5.1: Calculate t^s_{ij}

```
1: Initialize $t^s_{ij}$ $\forall\ i, j$
2: **repeat**
3:    Solve the queueing network to find the average idle time $E[T_k|I]$ for every service
      center $s/c_k$.
4:    **for all** service center $s/c_k$ **do**
5:       Set $i = s(k)$ and $j = d(k)$
6:       Evaluate $\sigma_i$ using Equation (5.2)
7:       $\epsilon_k = |t^s_{ij} - \frac{2\sigma_i}{R}|$ 1
8:       $t^s_{ij} = \frac{2\sigma_i}{R}$
9:    **end for**
10: **until** $\epsilon_k < threshold \ \ \forall k$
```

Figure 5.10 shows σ_i for fragments 1 through 5 obtained using equation (5.2) as well as simulation for our case study in Figure 2.3, using the parameters in Table 2.2 and the following parameters:

- $\rho_i = 1pkt/min = 4.8bit/sec \ \forall\ i$, i.e., based on a packet size of 36 bytes.

- $R = 38.4kbps$.

- Two MRs that move at a speed of $3.89m/s$.

- S_i's are as shown in Figure 2.3.

Figure 5.10 shows that the proposed algorithm achieves an accurate estimation of the average load, where the average load is the average amount of buffered data in a fragment. The maximum error, compared to simulation results, is 10%. Figure 5.11 shows

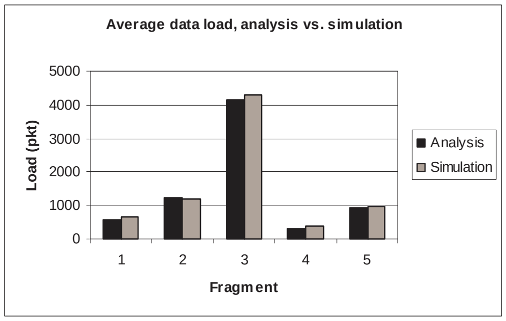


Figure 5.10: The average load of buffered data at fragments 1 through 5 for the example in Figure 2.3 and Table 2.3.

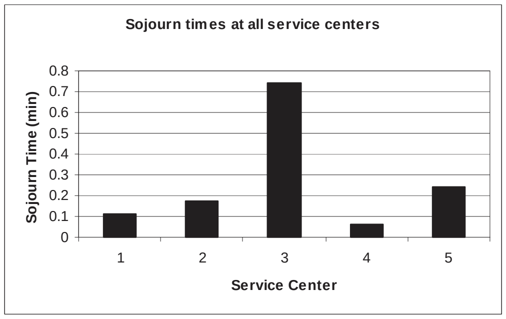


Figure 5.11: The new calculated sojourn time at all service centers for the example in Figure 2.3 and Table 2.3.

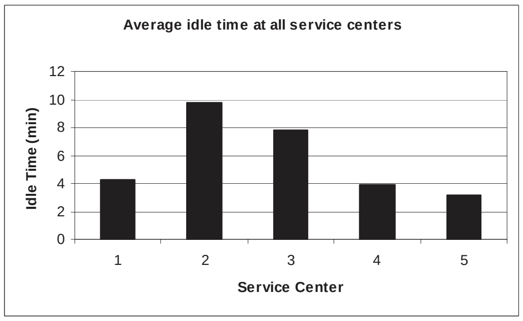


Figure 5.12: The average idle time at all service centers using the new sojourn times for the example in Figure 2.3 and Table 2.3.

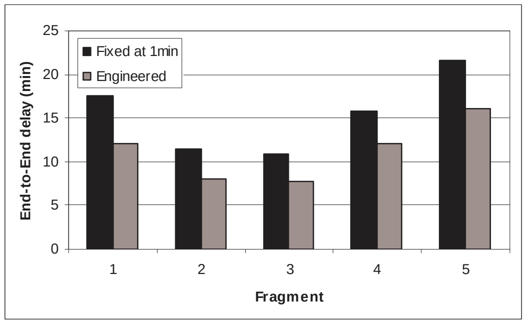


Figure 5.13: The average idle time at all service centers using the new sojourn times for the example in Figure 2.3 and Table 2.3.

the new calculated average sojourns times. Note that $1min$ that was assumed before is much longer than the required time. The sojourn times were reduced by an average of 73%.

The new average idle times at all service centers after using sojourn times in Figure 5.10 are shown in Figure 5.12. The maximum reduction of the average idle time is about 25%.

As another example, we consider the example in Figure 3.6 with the parameters given in Table 3.1. We used a data generation rate of $4pkt/min$ and a data transmission rate of $74kbps$. Figure 5.14 shows the average data load in fragments 1 through 4. Again, the proposed algorithm approximates the average data load with good accuracy, maximum error is (6.25%). The new calculated sojourn times are shown in Figure 5.15. The

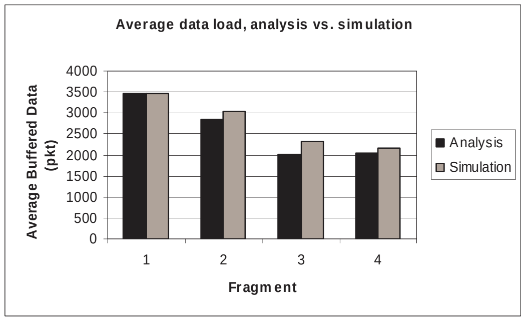


Figure 5.14: The average load of buffered data at fragments 1 through 5 for the example in Figure 3.6 and Table 3.1.

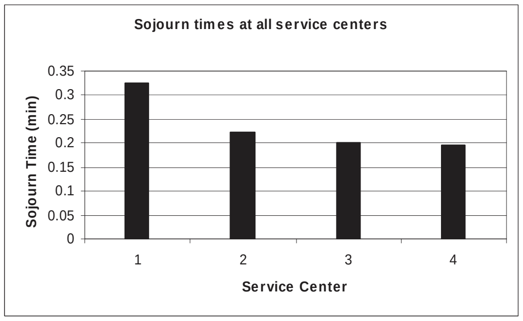


Figure 5.15: The new calculated sojourn time at all service centers for the example in Figure 3.6 and Table 3.1.

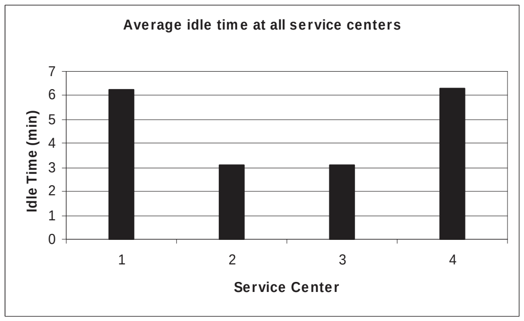


Figure 5.16: The average idle time at all service centers using the new sojourn times for the example in Figure 3.6 and Table 3.1.

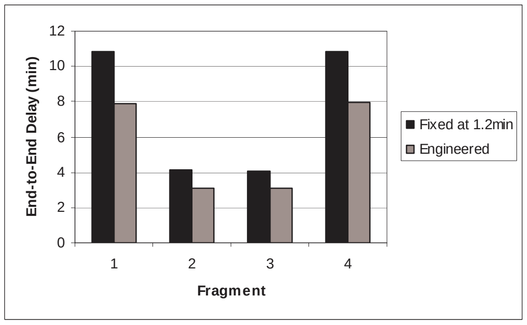


Figure 5.17: The average idle time at all service centers using the new sojourn times for the example in Figure 3.6 and Table 3.1.

sojourn times were reduced by an average of 80%.

As shown in Figure 5.16, engineering the service time resulted in an improvement on the average idle time of about 25% from the case of fixed t_{ij}^{s} for all service centers. The end-to-end delay was also improved as shown in Figures 5.13 and 5.17. This average improvement was 28% for the case-study in Figure 2.3 and 25% for the example of Figure 3.6.

5.3 Fragment Detection

In this section we propose some general guidelines on how to detect fragments in a network and provide information about the locations of fragments to the base station

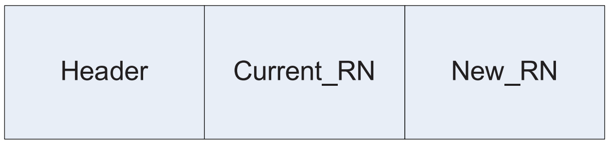


Figure 5.18: General structure of the NEW_FRAGMENT packet.

(BS). For a sensor node i, let:

- $ID(i)$ be the unique identification number (ID) of node i.
- $RN(i)$ be the ID of the reference of node i. Initially, and as long as the network is connected, the base station is the reference to all nodes in the network, therefore, $RN(i) = ID(BS)\ \forall i$.
- $BN(i)$ be a $TRUE/FALSE$ flag used to indicate whether node i has detected failures, i.e., lies on the boundary of a failure region. Initially, $BN(i) = FALSE\ \forall i$.

As shown in Algorithm 5.2, when a node i detects the failure of one or more of its neighbors, it reports that failure to its reference node, i.e., $RN(i)$. The $RN(i)$ must respond back with an acknowledgment (ACK) message. If node i does not receive an ACK message within a *Timeout* period, it assumes that the failure it has just observed or some other failure in the network has caused a network fragmentation. Therefore, it floods the network, i.e., the reachable nodes, with a *NEW_FRAGMENT* message. The structure of this message is shown in Figure 5.18, and the flooding process is described in Algorithm 5.3. The smallest RN is adopted by all sensor nodes in the fragment and is set as the new reference node. This will make the process recursive and will allow the detection of all possible fragments.

On the other hand, when the base station receives (or detects) any failure reports, it releases a mobile agent to collect information about the fragments. It collects information, i.e., locations, about boundary nodes, i.e., those with $BN(i) = TRUE$. When the mobile agents is done with the boundary of one fragment, it moves according to a certain

Algorithm 5.2: FRAGMENT DETECTION

```
1:  for all Sensor node $i$ do
2:      $BN(i) = FALSE$
3:      $RN(i) = ID(BS)$
4:  end for
5:  loop
6:      if a neighboring node failed then
7:          Report the failure to the $RN(i)$
8:          $BN(i) = TRUE$
9:          if no ACK is received from the $RN(i)$ then
10:             $createFragment(i)$
11:         end if
12:     end if
13: end loop
```

Algorithm 5.3: $createFragment(i)$

```
1: NEW_FRAGMENT pkt
2: $pkt.Current\_RN = RN(i)$
3: $pkt.New\_RN = ID(i)$
4: $Broadcast(pkt)$
```

strategy to the next fragment and collects information from there too. Eventually, the mobile agent delivers the collected information back to the base station.

5.4 Summary

We have addressed the issue of data delivery of FWSN that spread over large areas. Three approaches were evaluated, whole-network, proximity-based division, and area-based division. The results show that, in general, proximity-based division is the best among all others. However, this depends on the network topology. The benefit from proximity division degrades as the speed of mobile relays, or equivalently the distances between service centers, decreases.

We have also studied the issue of engineering the sojourn times at service centers. An algorithm to estimate sojourn times which are just sufficient to deliver data was introduced, and was shown to improve the *end-to-end* delay performance. We also provided some general guidelines on how to detect fragments and gather information about their locations.

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

A new form of network disconnection called *Fragmented Wireless Sensor Network* (*FWSN*) was addressed in this thesis. We proposed the use of resource rich mobile agents that move in the field and operate as data relays between fragments to eventually deliver data to the base station. A mathematical model based on modeling the network and the mobility of mobile relays as a queueing network was presented and used to evaluate the performance of a FWSN. The queueing network model was developed to capture a number of parameters including number and speed of MRs as well as the movement policy. Using steady state probabilities from the model, we then evaluated the distribution of the delay to deliver data between two fragments.

We also provided two approximations to evaluate the end-to-end delay, the CONV approach which assumes independence between idle times and the DPA approach which considers the dependence only between two consecutive service centers along the path. The DPA approach achieves high accuracy, whereas the CONV approach underestimates the end-to-end delay. Despite this inaccuracy, the CONV approach works with reasonable accuracy in divided FWSNs.
The results show that our model accurately evaluates the *fragment-to-fragment* and *fragment-to-sink* delays. It also suggest that optimizing the movement policy might lead to a better performance than just adding more MRs to an unoptimized policy.

Moreover, we studied the issue of engineering the sojourn time, i.e., the amount of time that an MR needs to spend at a service center to relay data. An algorithm to estimate sojourn times which are just sufficient to deliver data was introduced, and was shown to be accurate and improves the *end-to-end* delay performance.

Depending on the network topology, mobile data relays (MDR) might not be the best possible solution to deliver data in FWSNs. Instead, mobile data collectors may perform better. Moreover, a mobile sink or multiple mobile sinks might also perform better if sink mobility is feasible. Our model can be used to evaluate the *end-to-end* delay in both *"mobile data collectors (MDC)"* and *"mobile base stations (MBS)"* approaches. In the case of mobile base stations, the idle time at a service center is the *end-to-end* delay of the data generated at the fragment being served by that service center.
Modeling mobile data collectors is more intricate. The *end-to-end* delay for a fragment i that is served by service center s/c_k in the case of MDCs consists of two main parts:

1. The time until a MDC reaches the fragment and starts collecting data. This is exactly the idle time at a service center that we derived in Chapter 3.

2. The time until a MDC reaches the sink and delivers the data. This time can be evaluated as follows:

   (a) Solve the queueing network with one customer, and then

   (b) Derive the idle time distribution, using the same approach in Chapter 3, at the sink service center with the following initial condition: s/c_k *has started an idle period*. This idle time will be the travel time from s/c_k to the sink service center.

Therefore, our queueing network model can be used to evaluate the performance of all the three approaches (MDR, MDC, and MBS).
Our future work will include several related issues that have not been addressed in this work:

- Application specific MAC protocols: it is evident that sensor nodes in each fragment need to turn their radio on when MRs are present at the appropriate service center. Therefore, we believe that a significant amount of power could be saved by designing a MAC protocol that is specific for FWSN.

- We did not study the case when multiple data paths from a fragment to the base station exist. Therefore, we plan to extend our work to model this case.

- We plan to extend our modeling approach to model two cases; first, when an MR needs to move between two adjacent fragments to relay data, and second, when more than one MR can line up to form a communication path between two distant fragments.

- As the results in Chapter 3 indicate, optimizing the movement policy might lead to a better performance than that achieved by just increasing the number of mobile relays without enhancing their mobility scheme. Therefore, we plan to consider this optimization problem in our future work both, independently from other factors and jointly with the number of MRs, speed of a MR, and the cost of a route (i.e., probability of failure on a route).

- Sensor nodes in the areas outlined by the dashed lines and labeled $S1$, $S2$, and $S3$ in Figure 6.1 suffer more than other nodes in terms of power consumption because they have to relay more traffic than others. Such unbalance in the load distribution is a consequence of random deployment. Spots like $S1,\ S2,\ and\ S3$ are usually referred to as *hot spots* or *hot zones*.

  We plan to use mobile relays to help the sensor nodes in each hot zone by spending some time (we call this time the *sojourn time*) at that zone, and act to relay traffic on behalf of sensor nodes in the hot spot. Once an MR is present in a hot zone, all sensor nodes in that zone turn off their radio transceivers to save power. We can use the same queueing network model presented in Chapter 2 to evaluate the average idle time at a hot zone, i.e., the time duration in which a hot zone did not have any MR.

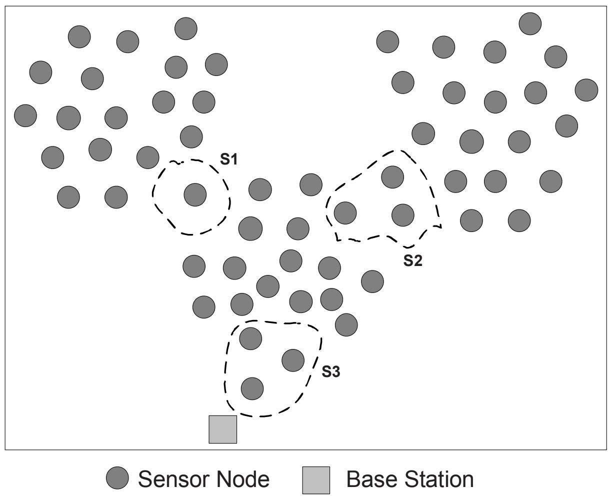


Figure 6.1: Random deployment might cause some sensor nodes to run out of power and die early because of unbalanced load distribution.

APPENDIX A THE EQUILIBRIUM DISTRIBUTION OF A GORDON-NEWELL QUEUEING NETWORK

Consider a *Gordon-Newell* network [45] with M queueing nodes and a constant population of K customers that circulate in the network. Each queueing node i consists of a FIFO queue and an infinite number of service channels that have the same service time which is drawn independently from an exponential distribution with mean $\frac{1}{\mu_i}$. A customer leaving queue i goes to queue j with probability q_{ij}.
Let n_i be the number of customers at queue i, and let $\overrightarrow{N} = [n_1, \ldots, n_M]$ be the state of the network such that $\sum_{i=1}^{M} n_i = K$ and Ω_K be the set of all possible network states with K customers. It has been shown in [45] that the steady state (equilibrium) distribution of the network state is given by,

$$\pi(\overrightarrow{N}) = \frac{1}{G(M,K)} \prod_{i=1}^{M} \left(\frac{\lambda_i}{\mu_i}\right)^{n_i} \tag{A.1}$$

where λ_i is the effective arrival rate at queue i which can be obtained by solving the set of dependent equations

$$\lambda_i = \sum_{j=1}^{M} \lambda_j q_{ji}, \quad 1 \leq i \leq M \tag{A.2}$$

One of the λ_i's should be assigned a non-zero positive value, say 1, in order to solve for the other arrival rates. On the other hand, $G(M,K)$ is a normalization constant that makes all the $\pi(\overrightarrow{N})$ sum to one. Therefore,

$$G(M,K) = \sum_{\overrightarrow{N} \in \Omega_K} \prod_{i=1}^{M} \left(\frac{\lambda_i}{\mu_i}\right)^{n_i} \tag{A.3}$$

The computational complexity of equation (A.3) is exponential because the summation includes $\binom{M+K-1}{K}$ terms. J. P. Buzen proposed a recursive method, called the *Convolution* algorithm, to evaluate $G(M, K)$ in $O(MK)$ iterations [40]. Below, we describe this algorithm.

The Convolution Algorithm to Compute G(M,K)

- For any $m = 1, \ldots, M$ define:

$$\rho_m = \frac{\lambda_m}{\mu_m} \tag{A.4}$$

- Let $k = 1, \ldots, K$ and $m = 1, \ldots, M$, then $G(M, K)$ can be computed recursively using the following recursive formula:

$$G(m, k) = G(m-1, k) + \rho_m G(m, k-1) \tag{A.5}$$

- There are two boundary conditions for this formula:

$$G(m, 0) = 1, \quad m{=}0, \ldots, M \tag{A.6}$$

$$G(0, k) = 0, \quad k{=}1, \ldots, K \tag{A.7}$$

APPENDIX B GLOSSARY

- $\boldsymbol{M}$: Total number of queues in a closed queueing network.

- $\boldsymbol{K}$: Population size in a closed queueing network.

- $\boldsymbol{n_i}$: Number of customers at queue i.

- $\overrightarrow{\boldsymbol{N}}$: A state of a closed queueing network such that $\overrightarrow{N} = \{n_1(\overrightarrow{N}), \ldots, n_M(\overrightarrow{N})\}$. Therefore, $\sum_{i=1}^{M} n_i(\overrightarrow{N}) = K$.

- $\boldsymbol{\Omega}$: The set of all possible system states, $|\Omega| = \binom{M+K-1}{M}$.

- $\boldsymbol{u_j}$: The state-independent service rate of *queue* j.

- $\boldsymbol{q_{ij}}$: The probability that a customer leaving *queue* i goes to *queue* j.

- $\boldsymbol{\pi(\overrightarrow{\mathbf{N}})}$: The steady-state probability of state $\overrightarrow{N}$.

- $\boldsymbol{E_i}$: The number of system states in which $\boldsymbol{n_i}\mathbf{=0}$.

- $\boldsymbol{p^i_{idle}(t, \overrightarrow{N})}$: The probability that *queue* i is idle for time greater than t.

- $\boldsymbol{\overrightarrow{p}^{\,i}_{idle}(t)}$: A column vector of $p^i_{idle}(t, \overrightarrow{N})$ over all the states in which n_i=0. Therefore, $\overrightarrow{p}^{\,i}_{idle}(t) = [p^i_{idle}(t, \overrightarrow{N}_1),\ \ldots,\ p^i_{idle}(t, \overrightarrow{N}_{E_i})]^T$.

- $\boldsymbol{U_i}$: A row vector of ones such that $|U_i| = E_i$.

- $\boldsymbol{T_i}$: A random variable that represents the idle time of *queue* i.

- $\boldsymbol{F_{T_i}(t)}$: The cumulative density function of T_i, i.e., $F_{T_i}(t) = probability\{T_i \leq t\}$.

- $\boldsymbol{f_{T_i}(t) = \frac{d}{dt}F_{T_i}(t)}$: The probability density function of T_i, i.e., $f_{T_i}(t) = \frac{d}{dt}F_{T_i}(t)$.

- $\boldsymbol{F_{T_i|idle}(t)}$: The cumulative density function of T_i given that queue i is idle at the reference time t=0. Therefore, $F_{T_i|idle}(t) = probability\{\ T_i \leq t \mid n_i{=}0\ at\ t{=}0\}$.

- $\boldsymbol{f_{T_i|idle}(t)}$: The probability density function of T_i given that queue i is idle at the reference time t=0. Therefore, $f_{T_i|idle}(t) = \frac{d}{dt}F_{T_i\ |\ idle}(t)$.

- $\boldsymbol{s/c_i}$: The i^{th} service center in a fragmented wireless sensor network.

- $\boldsymbol{p_i(t, \zeta_i)}$: The probability that s/c_i is idle for time t and at $t + \Delta t$ the condition ζ_i is satisfied.

- $\boldsymbol{p_{I|I}(i, j, t)}$: The probability that s/c_i is idle for time t given that s/c_j was also idle.

- $\boldsymbol{p_{I|B}(i, j, t)}$: The probability that s/c_i is idle for time t given that s/c_j was busy.

- $\boldsymbol{p_{B|I}(i, j)}$: The probability that s/c_i is busy given that s/c_j was idle.

- $\boldsymbol{p_{B|B}(i, j)}$: The probability that s/c_i is busy given that s/c_j was busy.

- $\boldsymbol{h^i_{idle}(0, \overrightarrow{\mathrm{N}}|n_j{=}1\ at\ t{=}0)}$: The probability of $\overrightarrow{N}$, where $n_i(\overrightarrow{N})$=0, given that n_j changed from 0 to 1.

- $\boldsymbol{h^i_{idle}(0, \overrightarrow{\mathrm{N}}|n_j{>}0\ at\ t{=}0)}$: The probability of $\overrightarrow{N}$, where $n_i(\overrightarrow{N})$=0, given that n_j>0.

- $\boldsymbol{h^i_{busy}(0, \overrightarrow{\mathrm{N}}|n_j = 1\ at\ t{=}0)}$: The probability of $\overrightarrow{N}$, where $n_i(\overrightarrow{N}) > 0$, given that $n_j > 0$.

- $\boldsymbol{h^i_{busy}(0, \overrightarrow{\mathrm{N}}|n_j > 0\ at\ t{=}0)}$: The probability of $\overrightarrow{N}$, where $n_i(\overrightarrow{N}) > 0$, given that $n_j > 0$.

- $\boldsymbol{\varphi}$: The status of a service center: $\varphi = 0$ means that the service center has just ended its idle period, $\varphi = 1$ means a busy service center, and $\varphi = 2$ means that the service center has started an idle period.

- $\digamma$: A path of n service centers $\{s/c_1, ..., s/c_n\}$

- $\boldsymbol{\alpha(t, \digamma, \upsilon, \varphi)}$: The probability that the υ^{th} service center along the path $\digamma$ is idle for time t given that its predecessor (i.e., $(\upsilon-1)^{th}$ service center) was in a status φ.

- $\boldsymbol{\gamma(\digamma, \upsilon, \varphi)}$:The probability that the υ^{th} service center along the path $\digamma$ is busy given that its predecessor (i.e., $(\upsilon-1)^{th}$ service center) was in a status φ.

- $\boldsymbol{\psi(\digamma, t)}$: The probability mass function of the end-to-end delay along $\digamma$.

- $\boldsymbol{s(k)}$: The fragment that an MR at s/c_k relays data from.

- $\boldsymbol{d(k)}$: The fragment that an MR at s/c_k relays data to.

- $\boldsymbol{H_i}$: A set of all fragments that $FRAG-i$ lies on their data paths toward the sink.

- $\boldsymbol{S_i}$: The number of sensor nodes in fragment i.

- $\boldsymbol{\rho_i}$: The data generation rate in $bits/sec$ in fragment i.

- $\boldsymbol{R}$: The data transfer rate in $bits/sec$.

- $\boldsymbol{\varsigma_i}$: The average amount of relayed data, in bits, that will be temporarily buffered in fragment i.

- $\boldsymbol{\sigma_i}$: The average total amount of data (generated and relayed) that will be temporarily buffered in fragment i.

Bibliography

[1] T. Voigt, H. Ritter, and J. Schiller. Utilizing solar power in wireless sensor networks. *The 28th Annual IEEE International Conference on Local Computer Networks (LCN)*, October 2003.

[2] Crossbow Technology Inc. http://www.xbow.com, January 2007.

[3] L. Iftode, C. Borcea, and P. Kang. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, chapter Cooperative Computing in Sensor Networks. CRC Press, 2004.

[4] D. Eickstedt and M. Benjamin. Cooperative target tracking in a distributed autonomous sensor network. *Proceedings of OCEANS*, September 2006.

[5] A. Mainwaring, D. Culler, et al. Wireless sensor networks for habitat monitoring. *The 1st ACM international workshop on wireless sensor networks and applications (WSNA)*, September 2002.

[6] S. Kim, S. Pakzad, et al. Wireless sensor networks for structural health monitoring. *The 4th international conference on embedded networked sensor systems (SenSys)*, November 2006.

[7] T. Yan, T. He, and J. Stankovic. Differentiated surveillance for sensor networks. *The 1st international conference one embedded networked sensor systems (SenSys)*, November 2003.

[8] R. Gupta and S. Das. Tracking moving targets in a smart sensor network. *The vehicular technology conference (VTC)*, October 2003.

[9] J. Aslam, Z. Butler, et al. Tracking a moving object with a binary sensor network. *The 1st international conference on embedded networked sensor systems (SenSys)*, November 2003.

[10] A. Knaian. A wireless sensor network for smart roadbeds and intelligent transportation systems. MS Thesis, MIT Department of Electrical Engineering and Computer Science and the MIT Media Laboratory., May 2000.

[11] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, June 2004.

[12] H. Yang, F. Ye, et al. Toward resilient security in wireless sensor networks. *The 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, May 2005.

[13] G. Xing, X. Wang, et al. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, August 2005.

[14] J. Al-Karaki and A. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, December 2004.

[15] S. Bergbreiter and K.S.J. Pister. Cotsbots: An off-the-shelf platform for distributed robotics. *Proceeding of the IEEE/RSJ Intl. Conference on Intelligent Robots and System*, October 2003.

[16] M. McMickell, B. Goodwine, and L. Montestruque. Micabot: A robotic platform for large-scale distributed robotics. *IEEE International Conference on Robotics and Automation*, September 2003.

[17] E. Ekici, Y. Gu, and D. Bozdag. Mobility based communication in wireless sensor networks. *IEEE Communications Magazine*, July 2006.

[18] D. Chae, K. Han, et al. Power saving mobility protocol for sensor network. *The Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WSTFEUS'04)*, May 2004.

[19] A. Kansal, M. Rahimi, et al. Controlled mobility for sustainable wireless sensor networks. *The Sensor and Ad Hoc Communications and Networks, IEEE SECON*, October 2004.

[20] B. Liu, P. Brass, et al. Mobility improves coverage of sensor networks. *The 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, May 2005.

[21] T. Chin, P. Ramanathan, and Kewal Saluja. Analytic modeling of detection latency in mobile sensor networks. *The 5th international conference on Information processing in sensor networks (IPSN)*, April 2006.

[22] N. Bisnik, A. Abouzeid, and V. Isler. Stochastic event capture using mobile sensors subject to a quality metric. *The 12th annual international conference on Mobile computing and networking (MobiCom)*, September 2006.

[23] R. Shah, S. Roy, et al. Data mules: Modeling a three-tier architecture for sparse sensor networks. *The first IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[24] S. Jain, R. Shah, et al. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications*, June 2006.

[25] A. Kansal, A. Somasundara, et al. Intelligent fluid infrastructure for embedded networks. *The 2nd international conference on Mobile systems,applications, and services (MobiSys)*, June 2004.

[26] J. Luo and J. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. *The 24th IEEE INFOCOM*, March 2005.

[27] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. *IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[28] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas. Sink mobility protocols for data collection in wireless sensor networks. *The international workshop on Mobility management and wireless access (MobiWac)*, October 2006.

[29] J. Luo, J. Panchard, et al. Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. *The International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2006.

[30] Z. Vincze, D. Vass, et al. Adaptive sink mobility in event-driven multi-hop wireless sensor networks. *The first international conference on integrated Internet ad hoc and sensor networks*, May 2006.

[31] M. Younis, M. Bangad, and K. Akkaya. Base station repositioning for optimized performance of sensor networks. *Proceedings of the IEEE VTC*, October 2003.

[32] W. Wang, V. Srinivasan, et al. Using mobile relays to prolong the lifetime of wireless sensor networks. In *The Eleventh Annual International Conference on Mobile Computing and Networking (MobiHoc)*, 2005.

[33] H. Jun, W. Zhao, et al. Trading latency for energy in densely deployed wireless ad hoc networks using message ferrying. *Ad Hoc Networks*, May 2007.

[34] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. *The 5th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, May 2004.

[35] Y. Mei, C. Xian, et al. Repairing sensor network using mobile robots. *The ICDCS International Workshop on Wireless Ad hoc and Sensor Networks*, July 2006.

[36] J. Bredin, E. Demaine, et al. Deploying sensor networks with guaranteed capacity and fault tolerance. *The 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, May 2005.

[37] S. Bose. *An Introduction to Queueing Systems*. Kluwer Academic/Plenum Publishers, 2002.

[38] T. Robertazzi. *Computer Networks and Systems: Queueing Theory and Performance Evaluation*. Springer, 3 edition, 2000.

[39] O. J. Boxma. *Queuing Theory and Its Applications*. North-Holland, 1988.

[40] J. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, September 1973.

[41] P. Levis. Tossim: Accurate and scalable simulation of entire tinyos applications. *The First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2003.

[42] Philip Levis And. *TOSSIM: A Simulator for TinyOS Networks*, March 2007. User's manual, in TinyOS documentation. "www.cs.berkeley.edu/ pal/pubs/nido.pdf".

[43] M. Demmer, P. Levis, et al. Tython: a dynamic simulation environment for sensor networks. Technical report, University of California, Berkeley, 2005.

[44] R. Nelson. *Probability, Stochastic Processes, And Queueing Theory: The Mathematics of Computer Performance Modeling.* Springer-Verlag, 1995.

[45] W. Gordon and G. Newell. Closed queuing systems with exponential servers. *Operations Research*, April 1967.