Delay/Fault-Tolerant Mobile Sensor Networks (DFT-MSN’s): A New Paradigm for Pervasive Information Gathering

A Dissertation

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Yu Wang

Spring 2007
Delay/Fault-Tolerant Mobile Sensor Networks (DFT-MSN’s): A New Paradigm for Pervasive Information Gathering

Yu Wang

APPROVED:

Hongyi Wu, Chair
Assistant Professor of Computer Science

Magdy Bayoumi
Lamson Professor of Computer Engineering

Dmitri Perkins
Assistant Professor of Computer Science

Nian-Feng Tzeng
Lockheed Martin Professor of Computer Engineering

C. E. Palmer
Dean of the Graduate School
This dissertation is dedicated to my family for their support and encouragement. Their love and patience gave me the strength and hope to finish this work.
ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Dr. Hongyi Wu for his outstanding guidance and continuous support during my Ph.D. study. He is knowledgable, considerate, enthusiastic, and always supportive. It was really a pleasure to work with him. I would like to thank my dissertation committee members, Dr. Dmitri Perkins, Dr. Magdy Bayoumi, and Dr. Nian-Feng Tzeng, for their encouragement and advice. I would like to acknowledge Professor Feng Lin, who is with Sichuan University and was a visiting scholar at our center, for his support of my research work. In addition, I would like to thank all the members of our lab for the happy time we had in the last couple of years. Finally, I would like to thank my wife and my parents for their constant encouragement and support.
Table of Contents

1 Introduction .............................................................................................................. 1

2 Background and Related Works ........................................................................... 7
  2.1 Background ........................................................................................................ 7
    2.1.1 Pervasive Computing .................................................................................. 7
    2.1.2 Wireless Mobile System ........................................................................... 9
    2.1.3 Sensor Networks ....................................................................................... 11
  2.2 Related Work ..................................................................................................... 15
    2.2.1 Routing Protocols in Traditional Sensor Networks ................................. 15
    2.2.2 Data Delivery in Delay Tolerant Networks ............................................. 19

3 Data Delivery in DFT-MSN .................................................................................. 24
  3.1 An Overview of DFT-MSN ................................................................................ 24
  3.2 Studies of Two Basic Approaches .................................................................. 27
    3.2.1 Basic Approach I: Direct Transmission .................................................. 28
    3.2.2 Basic Approach II: Flooding .................................................................... 34
    3.2.3 Observation from The Two Basic Approaches ....................................... 39
  3.3 Replication-Based Efficient Data Delivery Scheme (RED) ......................... 40
    3.3.1 Data Delivery ............................................................................................ 40
    3.3.2 Message Management .............................................................................. 43
  3.4 Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD) .... 46
    3.4.1 FAD Parameter: Message Fault Tolerance ........................................... 47
    3.4.2 FAD Data Delivery Scheme .................................................................... 48
  3.5 Simulation Results ............................................................................................ 50

4 Protocol Design and Optimization for DFT-MSN ............................................. 60
  4.1 Cross-Layer Data Delivery Protocol for DFT-MSN ......................................... 61
    4.1.1 Asynchronous Phase ................................................................................ 61
    4.1.2 Synchronous Phase .................................................................................. 63
  4.2 Protocol Optimizations ...................................................................................... 64
    4.2.1 Periodic Sleeping ..................................................................................... 65
    4.2.2 Collision Avoidance During RTS Transmission ...................................... 67
    4.2.3 Collision Avoidance During CTS Transmission ...................................... 69
  4.3 Simulation Results ............................................................................................ 71

5 Analytic Study for DFT-MSN ............................................................................. 80
  5.1 Generic Analytic Model for DFT-MSN ............................................................ 80
    5.1.1 General Assumptions ............................................................................... 82
    5.1.2 Service Time ............................................................................................. 83
    5.1.3 Message Arrival ......................................................................................... 85
    5.1.4 Queuing Characteristics .......................................................................... 85
List of Tables

Table 3.1  Summary of differences between DFT-MSN and conventional WSN . 27
Table 3.2  Default simulation parameters . . . . . . . . . . . . . . . . . . . . . . 51
Table 5.1  Summary of major symbols and notations used in the analysis . . . . 80
Table 6.1  Experimental results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108
List of Figures

Figure 2.1 Sensor network ............................................. 12
Figure 2.2 Typical components of a sensor node ...................... 14
Figure 3.1 An overview of DFT-MSN architecture. $S_1$-$S_{10}$: sensors; 
$HES_1$-$HES_3$: high end sensors (sinks); $AP_1$-$AP_8$: access points of backbone network .................................................. 25
Figure 3.2 Performance of direct transmission with infinite buffer space under $N = 100, n = 10, T = 50, w = 150, a = 0.0314$ .................. 30
Figure 3.3 Performance of direct transmission with finite buffer space under $N = 100, n = 10, K = 20, T = 20, w = 20, a = 0.0314$ .............. 31
Figure 3.4 Performance of flooding schemes .......................... 36
Figure 3.5 Determining optimal $b$ to minimize $S$ in erasure coding ........ 44
Figure 3.6 Update of delivery probability ............................. 52
Figure 3.7 Impact of the number of sink nodes ....................... 55
Figure 3.8 Impact of maximum queue length .......................... 56
Figure 3.9 Impact of nodal speed ($m/s$) .............................. 58
Figure 3.10 Impact of node density ................................. 59
Figure 4.1 Proposed cross-layer data delivery protocol: (a) Sender; (b) and 
(c) Qualified neighbors; (d) Unqualified neighbors; (e) Possible new arrivals 62
Figure 4.2 Impact of number of sink nodes .......................... 76
Figure 4.3 Impact of number of sensor nodes ......................... 77
Figure 4.4 Impact of maximum queue size ............................ 78
Figure 4.5 Impact of average nodal speed ............................ 79
Figure 5.1 Direct delivery scheme with uniform distribution mobility pattern . 97
Figure 5.2 Direct delivery scheme with power-law distribution mobility pattern 98
Figure 5.3 ZebraNet’s history-based scheme with power-law distribution 
mobility pattern .......................................................... 99
Figure 5.4  Simple replication-based scheme with power-law distribution mobility pattern ................................................................. 101

Figure 5.5  Simple replication-based scheme with power-law distribution mobility pattern ................................................................. 102

Figure 6.1  Message format ......................................................................................................................................................... 106

Figure 6.2  Experimental scenario. (The circular boundary is for illustration only. Actual boundary is irregular.) ........................................ 107
Chapter 1
Introduction

The concept of Ubiquitous/Pervasive Computing was first introduced in the early nineties as the third wave of computing that follows the eras of the mainframe computer and the personal computer. The ultimate goal is to develop the seamless integrated environments saturated with computing and communications capability yet gracefully adapted to human users. The major challenge in turning this into reality is the development of distributed system architectures that can effectively bridge between the information world and the physical world. The current trend focuses on embedding the computing devices into the physical objects and surroundings where people work and live, as well as deploying wearable devices that people can carry or wear during their daily life.

One of the key elements of pervasive computing is information gathering, which collects information from the ubiquitous equipment for further storage or processing. Pervasive information gathering plays an important role in many applications. One typical example is flu virus tracking, where the goal is to collect data of flu virus (or any epidemic disease in general) in the area with high human activities in order to monitor and prevent the explosion of devastating flu. Another example is air quality monitoring, where the goal is to track the average toxic gas breathed by people everyday. The aforementioned applications share several unique characteristics. First, the data gathering is human-oriented. More specifically, while samples can be collected at strategic locations for flu virus tracking or air quality monitoring, the most accurate and effective measurement shall be taken where the people are, making it a natural approach to deploy wearable sensing units that closely adapt to human activities. Second, we observe that delay and faults are usually tolerable in such applications, which aim at gathering massive
information from a statistical perspective and to update the information base periodically. In addition, this information gathering should be transparent, without any interference in people’s daily lives. For example, a person should not be asked to take special actions (e.g., to move to a specific location) to facilitate information acquisition and delivery.

Information gathering relies on sensors. Advancement in integrated, low-power, wireless communication devices and sensors makes it possible to deploy deeply embedded wireless sensor networks (WSN) for various purposes, such as surveillance detection, wildlife tracking, virus monitoring, military usages, etc. A typical sensor network is a self-organized network that consists of a large number of sensor nodes that are densely deployed in the area of interest. Each sensor node is equipped with various sensing units, which can capture events of interest, and a wireless transceiver, which transmits the captured events back to some special central processing nodes, namely sink nodes. Since the deployment (e.g., position information) of each individual sensor node is not required to be pre-determined, the sensors can be randomly scattered around the phenomenon that may be inaccessible or unfriendly to human beings, such as chemistry experimental field, tropical forest, etc. Depending on the self-organizing capability, the sensors can then form a well-connected mesh network for data collection.

Although a large number of data delivery schemes have been proposed for the densely deployed, well connected wireless sensor networks, as to be discussed in Chapter 2, they may not work effectively in the aforementioned application scenarios for pervasive information gathering, because the connectivity between the mobile sensors is poor, and thus it is difficult to form a well connected mesh network for transmitting data through end-to-end connections from the sensor nodes to the sinks. In order to address this problem, we propose a Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN) for pervasive information gathering [1, 2, 3, 4]. The idea of DFT-MSN is originated from the Delay Tolerant Network, which was initially introduced to deal with the interplanetary
data transmission. In such a situation, the networking environments have totally different characteristics compared to traditional networks, such as wide variations in delivery delay and the lacking of peer-to-peer connection [5, 6]. A DFT-MSN has a two-layer hierarchical architecture. The upper layer in this hierarchy is a traditional backbone network with wireless access interface. The lower layer consists of two types of nodes, the wearable sensor nodes and the high-end sink nodes. The former are attached to mobile objects, gathering target information and forming a loosely connected mobile sensor network for information delivery. The latter can be either mobile nodes (e.g., mobile phones or personal digital assistants with sensor interfaces carried by people) or static nodes (e.g., personal computers at strategic locations), collecting data from the sensor nodes and forwarding them to the end user.

This dissertation focuses on the development of simple and efficient data delivery schemes tailored for DFT-MSN. Motivated by the Delay-Tolerant Network (DTN) [5] and pertinent work to be discussed in Chapter 2, we first study two basic approaches, namely, direct transmission and flooding. We analyze their performance by using queuing theory and statistics. Based on the analytic results that show the tradeoff between data delivery delay/ratio and transmission overhead, we introduce an optimized flooding scheme that minimizes transmission overhead in flooding. Then, we propose two simple and effective DFT-MSN data delivery schemes, namely Replication-Based Efficient Data Delivery Scheme (RED) and Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD). The RED scheme utilizes the erasure coding technology tailored for DFT-MSN in order to achieve the desired data delivery ratio with minimum overhead. It consists of two key components for data transmission and message management, respectively. The former makes decision on when and where to transmit data messages according to the delivery probability, which is the likelihood that a sensor can deliver data messages to the sink. The latter decides the optimal erasure coding parameters (including the number of data
blocks and the needed redundancy) based on its current delivery probability. The FAD scheme employs the message fault tolerance, which indicates the importance of the messages. The decisions on message transmission and dropping are made based on fault tolerance for minimizing transmission overhead. The system parameters are carefully tuned according to thorough analyses to optimize network performance. Extensive simulations have been carried out for performance evaluation. Our results show that the proposed DFT-MSN data delivery schemes achieve high message delivery ratio with acceptable delay and transmission overhead.

Meanwhile, we observe that without end-to-end connections due to sparse network density and sensor node mobility, routing in DFT-MSN becomes localized and ties closely to medium access control, which naturally calls for merging Layer 3 and Layer 2 protocols in order to reduce overhead and improve network efficiency. While cross-layer design has been discussed extensively in the past several years, this work distinguishes itself from others by considering the unique characteristics of the sparsely-connected, delay-tolerant mobile sensor network. The goal of conventional sensor network protocols is to optimize energy consumption with a given delay or throughput requirement, while the sensor nodes usually enjoy stable connectivity and ample channel bandwidth. DFT-MSN, however, is fundamentally an opportunistic network, where the communication links exist only with certain probabilities and become the scarcest resource. At the same time, the sensor nodes in DFT-MSN have very limited battery power, like those in other sensor networks. Clearly, there is a tradeoff between link utilization and energy efficiency. None of the existing sensor network protocols have considered such a unique network environment and performance tradeoff. Our goal is to make efficient use of the transmission opportunities whenever available, while keeping the energy consumption at the lowest possible level. To this end, we develop a cross-layer data delivery protocol for DFT-MSN, which includes two phases, i.e., the asynchronous
phase and the synchronous phase. In the first phase, the sender contacts its neighbors to identify a set of appropriate receivers. Since no central control exists, the communication in the first phase is contention-based. In the second phase, the sender gains channel control and multicasts its data message to the receivers. Furthermore, several optimization issues in these two phases are identified, with solutions provided to reduce the collision probability and to balance between link utilization and energy efficiency.

In order to better understand the queuing characteristics of DFT-MSN, we then introduce a generic queuing analytic model, which aims at providing an insight of the performance for DFT-MSN with various data delivery schemes and mobility patterns. The inputs of the model are the data delivery scheme employed and the nodal mobility pattern, while the outputs are the queuing characteristics of the network. Based on our analysis of the message arrival and service processes, we find that each individual sensor can be modeled as an M/M/1/K queue, and the whole network can be treated as a network of queues. Following Jackson network theory, major queuing characteristics of the network can thus be obtained. We also exemplify the generic analytic model with several representative data delivery schemes (including direct transmission, ZebraNet, and Replication-based Data Delivery) and nodal mobility patterns (such as uniform and power-law distributions). To validate our analytic models, we have carried out extensive simulations and observed a good match between analytic and simulation results. The proposed analytic models along with the numeric results provide a deep insight of the queuing characteristics for DFT-MSN with various data delivery schemes and mobility patterns.

Moreover, we also implement a DFT-MSN testbed by using Crossbow sensors, in order to further evaluate the performance of the proposed schemes. A small scale experiment has been carried out, which shows the effectiveness and efficiency of DFT-MSN.
The rest of the dissertation is organized as follows: Chapter 2 discusses background and related work. Chapter 3 presents an overview of DFT-MSN and several efficient data delivery schemes tailored specially for DFT-MSN. Chapter 4 introduces a cross-layer data delivery protocol based on the principles presented in Chapter 3. Chapter 5 introduces the generic queuing analytic model. The testbed design is discussed in Chapter 6. Finally, Chapter 7 concludes the dissertation.
Chapter 2
Background and Related Works

In this chapter we introduce the background of and motivation for our proposed research and related work.

2.1 Background

2.1.1 Pervasive Computing

With computing devices becoming progressively small and powerful, the trend of computing tries to move beyond the realm of the personal computer to the tiny devices deeply embedded in people’s daily life. The idea of pervasive computing was envisioned in Mark Weiser’s early work [7, 8, 9, 10] as a system that “activates the world, makes computer so imbedded, so fitting, so natural, that we use it without even thinking about it, and is invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere.” The essence of this vision was the seamless integrated environments saturated with computing and communications capability, yet gracefully adapted to human users [11, 12, 13].

Pervasive computing evolves along a line of work dating back to the 1970s [14]. One of the major challenges in turning its ultimate goal into reality is the development of distributed system architectures that can effectively bridge between the information world and the physical world. Two notable earlier milestones in this evolution are distributed systems and mobile computing [11, 14]. The former arose at the intersection of personal computers and local area networks, covering several foundational areas to pervasive computing, including remote communication, fault tolerance, high availability, remote information access, and security. The latter, which further enables the systems to support
mobile nodes, can be grouped into five broad areas, including mobile networking, mobile information access, support for adaptive applications, system-level energy saving techniques, and locations sensitivity. Current research on pervasive computing further extends the previous work by incorporating several additional research challenges, e.g., effective use of smart spaces, invisibility, localized scalability, and masking uneven conditioning [11].

Meanwhile, pervasive computing may have different objectives to different people. At its core, however, it deals with three interweaved issues, including how people use the computing devices to perform certain tasks, how the applications are created and deployed, and how the ubiquitous information can enhance the application performance [15].

Many pervasive computing research projects have emerged in major universities and companies. The goal of the Aura Project [16, 17] at CMU is to provide the users with an invisible, but always existing, environment of computing and information services, which consists of various devices including wearable, handheld, desktop and infrastructure computers. It is characterized as “distraction free ubiquitous computing.” The Oxygen project [18] at MIT tries to understand what turns a dormant environment into an empowered one, where users can shift much of the burden of their tasks to the environment. It focuses both on technologies that enable the environment and on technologies that improve the user experience. The Portolano Project [19, 20] at the University of Washington addresses data-centric routing, which helps data migration among applications. The Easy Living project [21] at Microsoft targets at developing a prototype architecture and technologies for building intelligent environments.
2.1.2 Wireless Mobile System

The development of wireless mobile system evolved remarkably since the radio was first used to provide continuous contact with ships sailing in the English channel in 1897. The first mobile telephone service was introduced by AT&T in twenty-five major American cities in 1946 [22, 23]. The early mobile telephone system employed a single powerful transmitter, which used Frequency Modulation (FM) transmission in a half-duplex mode, and large tower to provide coverage of up to 50 miles or more from the base [22, 24]. While this approach achieved good coverage, it also implied difficulty on frequency reuse due to interference. Thus the available radio channels were locked up over a large area by only a small number of users. Faced with the fact that the allocated spectrum cannot follow the demand for mobile telephone services, it became necessary to restructure the radio telephone system so that high capacity and large coverage can be achieved simultaneously with limited radio spectrum. During the 1950s and 1960s, the AT&T Bell Labs, together with other companies, developed the theory and techniques of cellular radiotelephony, which replaces a single, high power transmitter (large cell) with many low power transmitters (small cells), with the available frequencies reused among these cells. Each of these low power transmitters provides services to only a small portion of the entire area. In the late 1970s, the first generation cellular system was standardized in the United States, called Advanced Mobile Phone System (AMPS) [25]. Later the Conference of European Posts and Telegraphs (CEPT) formed a study group to develop the Global System for Mobile communication [26, 27] (GSM). The phase I of the GSM specifications, which are based on Time Division Multiple Access (TDMA) [28, 29, 30], were published in 1990. The second generation system were also developed in American and Asian countries, such as IS-54 (based on TDMA), IS-95 (based on CDMA [31, 32]), and Japanese Digital Cellular (JDC) standards. The explosive growth of Internet has stimulated the interest in developing data services in existing networks, such as GPRS.
system [33, 34] (General Packet Radio Service). The third generation (3G) [35] systems are designed to offer flexible multimedia services to users on-demand anywhere, and at any time. The main standards of 3G systems, known collectively as IMT2000 [36], are a single family of compatible standards. The most important IMT2000 proposals are the Universal Mobile Telecommunications System (UMTS) [37] with Wideband-CDMA, CDMA2000, and Time Division Synchronous CDMA (TDSCDMA).

The widespread success of cellular system has led to the development of many new wireless systems. Mobile IP was developed to enable computers to maintain Internet connectivity while moving from one Internet attachment point to another [22, 38]. Each node maintains two address: the home address and the care-of address. The former is the address of the node in its home network, while the latter is its address registered with the foreign network where it is visiting. Tunneling is used to forward IP datagrams from a home address to a care-of address. The wireless local area networks (WLAN) [22] is now increasingly used to replace the wired local area networks. The smallest building block of a WLAN is a basic service set (BSS), which consists of a number of stations executing the same MAC protocol and competing for access to the same shared wireless medium. A BSS may be either isolated, or connected to a backbone distribution system (DS) through an access point (AP). Several basic service sets can be further organized into an extended service set (ESS) by a distribution system.

Recently, Mobile Ad-hoc Networks (MANET) [39, 40, 41, 42, 43, 44, 45, 46, 47] have been developed for the need of the rapid deployment of independent mobile users, e.g. establishing survivable, efficient, dynamic communication for emergency/rescue operations, disaster relief efforts, and military networks. Such network scenarios cannot rely on centralized and organized connectivity, resulting in a decentralized network, where all network activities including routing discovery must be performed by the nodes themselves. Each host is potentially a router and it is possible to dynamically establish
routes by connecting a sequence of neighboring hosts from a source to a destination in the ad-hoc network. The main challenges in the design and operation of ad-hoc networks stem from the possibility of rapid change of the network topology as well as the lack of a centralized control and management entity as in cellular system. There are essentially two different strategies for ad-hoc routing. Proactive protocols (e.g., [41, 44]) which precompute routes can avoid the long latency of route setup but suffer from poor scalability because of the need to maintain routing entries for all other nodes. An alternative way is to use on-demand approaches (e.g., [42, 45, 46]), where routes are not computed until there is data which needs to be sent. The on-demand approaches, on the contrary, provide better scalability with the cost of long latency. Hierarchy is also introduced to alleviate the scalability problem. One successful example is Bluetooth [48]. Originally developed by Ericsson, Bluetooth is a specification for low-cost, low-power, short-range radio links, and serves as replacement for the cables connecting all kinds of peripherals and portable devices.

2.1.3 Sensor Networks

The advancement in integrated, low-power, wireless communication devices and sensors makes it possible to deploy deeply embedded wireless sensor networks (WSN) for various purposes, such as surveillance detection, wildlife tracking, virus monitoring, military usages, etc. Similar to the development of other wireless personal communications systems, the development of wireless sensor networks goes back at least to the DARPA-sponsored Distributed Sensor Nets Workshop at Carnegie-Mellon University in 1978 [49, 50]. The early development of sensor networks was mainly driven by the interest of military usage [51, 52, 53]. The famous SensIT project [53] was launched in 1998 by DARPA to support research on wireless, ad-hoc networks for large distributed military sensor systems. Totally 29 projects from 25 institutions were funded under this
project [50]. With the maturity of the related technologies, the sensor networks have also been proved promising in many domains other than military usage, such as industrial control and monitoring, home automation, supply chain management, environmental sensing, health monitoring, etc. The representative works include the Wireless Integrated Network Sensors (WINS) project [54, 55, 56, 57] at the University of California at Los Angeles, the PicoRadio program [58, 59] and the Smart Dust project [60, 61] at the University of California at Berkeley, and µAMPS program [62] at the Massachusetts Institute of Technology.

A typical sensor network is a self-organized network which consists of a large number of sensor nodes that are densely deployed in the area of interest, as shown in Fig. 2.1. Each sensor node is equipped with various sensing units, which can capture events of interest, and wireless transceiver, which transmits the captured events back to some special central processing nodes, namely sink nodes [63]. Located close to or inside the sensor network, the sink nodes are normally equipped with more powerful processing, communication, and storage units so that they are capable of doing more complicated operations on collected data. Since the deployment (e.g., position information) of each individual sensor node is not required to be pre-determined, the sensors can be randomly scattered around the phenomenon which may be inaccessible or unfriendly to human beings, such as chemistry experimental field, tropical forest, etc. Depending on the self-organizing capability, the sensors can then form a well-connected mesh network for data collection.
The communication architecture of the sensor nodes normally consists of physical layer, data link layer, network layer, transport layer and application layer [63, 50]. In addition, due to their energy-constrained nature, most sensor network systems also have cross-layer power management schemes. Moreover, some systems may have mobile nodes. Thus mobility supporting are also embedded in the entire communication architecture.

The sensor networks can be classified into two broad categories, the event-driven sensor networks and the time-driven sensor networks, according to how they acquire and transmit data. In a time-driven sensor network, the sensors are activated periodically to acquire data and transmit them to the sink. Since data acquisition and transmission follow a predetermined schedule, the data rate is usually a constant. A typical example is the environmental monitoring sensor network that periodically collects data of temperature, humidity, wind speed, etc. In an event-driven sensor network, nodes don’t acquire and transmit significant data (with a high data rate) unless the target events (such as fire or intrusion) are detected, despite that they may also produce routine data periodically and infrequently. Data acquisition and transmission in the event-driven sensor network is usually unpredictable.

A Typical sensor node consists of four important components, i.e., sensing component, communication component, processing component, and power supply component [63]. In some special applications, the sensor node may also have additional parts, such as location detection and nodal mobility controller, as shown in Fig. 2.2. The sensing units usually consist of two components, i.e., sensors and ADC. The sensors capture the interested events, which are then converted from analog signal into digital signal using ADC. The processing unit usually includes a microprocessor and memory. The former does all the computation and management work, and the latter is always used to hold the application and temporarily buffer a small amount of user data. The communication unit has a transceiver which transmits and receives data messages. Finally, the power supply unit
provides power for all the other units. In some special system, the power supply unit also provides interfaces for the applications to monitor the power consumption. There are some unique characteristics of these networked sensors. First, they always have small physical size, which makes them suitable to be deeply embedded into the ubiquitous system. Secondly, these networked sensors, which are always equipped with a limited power source, should be very power efficient. We expect them to work as long as possible. In some application scenarios, however, replenishment of power resources might be impossible. Therefore, power efficiency is one of the key design issues of networked sensor. Third, the wireless transceiver equipped with the sensor nodes normally has limited transmission range, which places a challenging constraint for the design of data delivery scheme. Thus, the sensor network normally employs the multi-hop-based protocol for data delivery, instead of direct transmission from the sensors to the sink nodes.

Today, the sensors can be constructed using commercial components in small physical size and power consumption [64]. One of the most popular platforms is Berkeley Mote sensor node. The original mote has a 8-bit MCU (ATMEL 90LS8535), which is Harvard architecture with 16-bit addresses. It also has 32 8-bit general registers and runs at 4 MHZ.
and 3.0V. The memory is very constrained: 8KB of flash as the program memory and 512 bytes of SRAM as data memory. The radio is the most important component, which consists of an RF transceiver (TR1000)[65] working at 916.50 MHz, an antenna, and some other components for signal strength and sensitivity configuration. It has a speed up to 19.2 Kbps using OOK modulation. The sensing unit is a temperature sensor (AD7418), which has internal A/D converters.

2.2 Related Work

2.2.1 Routing Protocols in Traditional Sensor Networks

With the energy constrained nature of wireless sensors, it is utmost important to make efficient use of battery power in order to increase their lifetimes. In particular, since most energy of a sensor is spent in data transmission, which includes transmitting data generated by the node itself and the data relayed for other sensors [66, 67], finding an optimal approach for data transmission is particularly important.

The simplest scheme is flooding, where each node broadcasts its messages to all the neighbors. It does not require costly topology maintenance and complex route discovery algorithm. However, the deficiencies are also very obvious: implosion, overlap, and resource blindness [68]. A family of adaptive protocols called Sensor Protocols for Information via Negotiation (SPIN) [68] is designed to address the deficiencies of flooding. They are based on the idea that it is more efficient for the sensors to broadcast the brief information of the obtained data instead of the data itself. Gossiping [69] is a derivation of flooding, where nodes only send the incoming packets to a randomly selected neighbor, instead broadcasting them to all the neighbors. It eliminates the implosion problem with the cost of long delivery delay.

Several simple schemes are discussed in [63] for prolonging the sensor network’s lifetime. For example, the maximum available power approach chooses the path that has
the highest total available energy. The minimum hop approach uses the conventional shortest path. In the maximum minimum nodal energy approach, multiple paths to the sink are considered. In each path, the node with the minimal residual energy is identified and compared with other paths. The path with the largest minimal energy is chosen. [70] introduces an energy-based traffic spreading approach. It is an improvement over the shortest path approach, where the path tries to avoid any node with residual energy below a threshold value.

In [71], five power-aware metrics, namely Minimize Energy Consumed per packet, Maximize Time to Network Partition, Minimize Variance in Node Power Levels, Minimize Cost per Packet, and Minimize Maximum Node Cost, are employed for determining routes in wireless ad-hoc networks. Using these metrics in a shortest-cost routing algorithm is more power-efficient than conventional shortest-hop routing.

[72] aims to maximize the network lifetime until the first battery drains out. It reduces the problem to a maximum flow model in which the performance objective is to maximize the network lifetime. [73] formulates the same problem as anycast routing. It presents iterative algorithms for obtaining the optimal solution and derives an upper bound of the network lifetime for a given topology.

[74] proposes to setup the minimum-cost path through a modified flooding approach. In order to reduce overhead, a node defers rebroadcasting of the received message for a backoff interval proportional to its minimum cost. Multiple broadcasts may be eliminated by employing large backoff intervals at the cost of an increased setup time.

Network survivability is introduced in [75] as the metric for routing. Based on this metric, consistent use of the lowest energy paths may not be optimal for the network lifetime and long-term connectivity. Therefore, each node maintains and uses multiple routes simultaneously.
[76] introduces a new scheme for routing in wireless sensor networks. Each sensor node in the network has a vector of its routing direction and weight. A route is determined by solving a set of partial differential equations similar to the Maxwell’s equations.

[77] discusses the energy efficient data collection for sensor networks. It exploits the natural tradeoff between application quality and energy consumption of the sensors. Several sensor models (Always-Active, Active-Listening, Active-Sleep, and Active-Listening-Sleep) and the corresponding data collection protocols are studied for achieving high energy-efficiency. The simulation results indicate that the Active-Sleeping model consumes the least amount of sensor energy.

Since a sensor network is usually deployed as an integral entity to retrieve data in the area of interest, the lifetime of the whole sensor network is more important than those of individual nodes. [78] proposes a balance-based energy-efficient data delivery scheme by balancing the traffic between the nodes far away from the sink and the nodes closer to the sink. [79] defines the network lifetime to be the time that the sensor network lasts, until fewer than a fraction of ξ sensor nodes remains alive. It also proposes several energy-efficient communication schemes based on power control and load balancing. More specifically, an Integer Linear Programming (ILP) algorithm is first introduced to estimate the upper bound of the network lifetime. The authors then propose a Distributed Energy Efficient Routing (DEER) protocol for the time-driven sensor networks and a Residual Energy-based Traffic Splitting (RETS) protocol for the event-driven sensor network, respectively. The simulation results show that the proposed approaches can effectively distribute energy consumption evenly among the sensor nodes to prolong the network lifetime.

The protocol proposed in [80] computes an energy-efficient subnetwork, namely the minimum energy communication network (MECN), for a given communication network.
[81] extends it to Small MECN (SMECN), which can construct a smaller subnetwork than MECN, by using minimum energy property.

Another category of routing protocols for wireless networks is cluster-based. In particular, [82] studies the energy efficiency problem in non-homogeneous wireless networks. The intra-cluster communication uses low transmission power, while a high transmission power level is employed only for inter-cluster communication. In [83], Heinzelman et al. propose Low Energy Adaptive Communication Hierarchy (LEACH), a clustering-based protocol that utilizes randomized rotation of local cluster heads to evenly distribute the energy load among the sensors in the network. It is based on the assumption that the computation cost is much lower than the communication cost. Thus, local computation is employed in each cluster to reduce data transmission and the number of nodes involved in long distance transmission. [84] proposes a deterministic approach for cluster-head election. The new approach proves to be more energy-efficient than the one employed in [83]. [85] proposes a distributed algorithm for organizing sensors into a hierarchy of clusters with the objective of minimizing the total energy cost in the system. It also introduces an algorithm for deriving the optimal number of cluster heads at each level.

Sensor network is also a data-centric network, where interest dissemination is performed to assign the sensing tasks to the sensor nodes. Generally there are two approaches for interest dissemination [63]. In the first approach, the sink nodes propagate the interest to the sensor nodes, while in the second approach, the sensor nodes advertise the available data and wait for the request from the interested nodes. The authors in [86] also propose an attribute-based naming scheme to help identify the information of interest. In that system, the users can simply query by using the attributes of the interested event, instead specific nodes.
The directed diffusion is proposed in [87] for data dissemination. The sink first broadcast the task description to the nearby sensor nodes. The task information is then stored and propagated throughout the whole network. During the process, the gradients from the source back to the sink are set up. Thus, when a sensor node has data for transmission, it sends data along the gradient path.

In data centric routing, aggregation (or data fusion) is normally used to alleviate the implosion and overlap problems. Data coming from multiple sensor nodes are aggregated before being transmitted to the sink nodes individually. [88] studies how to reduce aggregation costs in large-scale sensor networks through distributed data compression and traffic aggregation. An optimal greedy solution is proposed for distributed compression subject to aggregation costs. It also discusses how to scale the approach to large sensor networks by investigating a hierarchical model for sensors, compressors, and sinks. The simulation results show that significant energy can be saved by jointly optimizing compression and aggregation structures.

In addition to routing, energy-aware medium access control (MAC) protocols are also explored in the literature, as can be found in [89, 90, 91, 92, 93, 94, 95].

### 2.2.2 Data Delivery in Delay Tolerant Networks

The Delay-Tolerant Network (DTN) is an occasionally connected network that may suffer from frequent partitions and that may be composed of more than one divergent set of protocol families [5]. DTN originally aimed to provide communication for the *Interplanetary Internet*, which focused primarily on the deep space communication in high-delay environments and the inter-operability between different networks deployed in extreme environments lacking continuous connectivity [5, 6]. An overall architecture of DTN has been proposed in [6], and it operates as an overlay above the transport layer to provide services such as in-network data storage and retransmission, interoperable
naming, authenticated forwarding, and a coarse-grained class of service. In [96], Burleigh et al. identify several fundamental principles that would underlie a DTN architecture and propose a new end-to-end overlay network protocol called Bundling. In [97], Fall et al. investigate the custody transfer mechanism to ensure reliable hop-by-hop data transmission, thus enhancing the reliability of DTN. That work also extends the DTN architecture with the concept of transaction abort.

DTN technology has been recently introduced into wireless sensor networks. Its pertinent work can be classified into the following three categories, according to their differences in nodal mobility.

(1) **Network with Static Sensors.** The first type of DTN-based sensor networks are static. Due to a limited transmission range and battery power, the sensors are loosely connected to each other and may be isolated from the network frequently. For example, the Ad hoc Seismic Array developed at the Center for Embedded Networked Sensing (CENS) employs seismic stations (i.e., sensors) with large storage space and enables storing and forwarding of bundles with custody transfer between intermediate hops [98]. In [99], wireless sensor networks are deployed for habitat monitoring, where the sensor network is accessible and controllable by the users through the Internet. The SeNDT (Sensor Networking with Delay Tolerance) project targets at developing a proof-of-concept sensor network for lake water quality monitoring, where the radio connecting sensors are mostly turned off to save power, thus forming a loosely connected DTN network [100]. DTN/SN focuses on the deployment of sensor networks that are inter-operable with the Internet protocols [101]. [102] proposes to employ the DTN architecture to mitigate communication interruptions and provide reliable data communication across heterogeneous, failure-prone networks.

(2) **Network with Managed Mobile Nodes.** In the second category, mobility is introduced to a few special nodes to improve network connectivity. For example, the Data
Mule approach is proposed in [103] to collect sensor data in sparse sensor networks, where a mobile entity called data mule receives data from the nearby sensors, temporarily store them, and drops off the data to the access points. This approach can substantially save the energy consumption of the sensors as they only transmit over a short range, and at the same time enhance the serving range of the sensor network.

(3) **Network with Mobile Sensors.** While all of the above delay-tolerant sensor networks center at static sensor nodes, ZebraNet [104] employs the mobile sensors to support wildlife tracking for biology research. The ZebraNet project targets at building a position-aware and power-aware wireless communication system. A history-based approach is proposed for routing, where the routing decision is made according to the node’s past success rate of transmitting data packets to the base station directly. The pioneering work of ZebraNet has motivated our research on mobile sensor networks. When a sensor meets another sensor, the former transmits data packets to the latter if the latter has a higher success rate. This simple approach, however, doesn’t guarantee any desired data delivery ratio. The Shared Wireless Info-Station (SWIM) system is proposed in [105, 106] for gathering biological information of radio-tagged whales. It is assumed in SWIM that the sensor nodes move randomly and thus every node has the same chance to meet the sink. A sensor node distributes a number of copies of a data packet to other nodes so as to reach the desired data delivery probability. In many practical applications, however, different nodes may have different probabilities to reach the sink, and thus SWIM may not work efficiently. Worst yet, some nodes may never meet the sink, resulting in failure of data delivery in SWIM. The pioneering work of ZebraNet and SWIM has motivated our research on mobile sensor networks. At the same time, we observe that the data transmission schemes employed in ZebraNet and SWIM are based on direct contact probability between sensor and sink, and thus inefficient. In addition, an erasure coding based data forwarding algorithm is proposed for opportunistic networks in [107].
simulation results show that this algorithm provides the best worst-case delay performance with a fixed amount of overhead. However, it neither explains how to determine the optimal value of replication overhead nor discusses the distribution scheme for the coded messages.

DTN technology has also been employed in mobile ad-hoc networks. Vahdat and Becker [108] propose an epidemic routing protocol for intermittently connected networks. When a message arrives at an intermediate node, the node broadcasts the message to all its neighbors so that the message can be propagated out quickly. Whenever two nodes move into the transmission range of each other, they exchange pair-wise messages that the peer node has not seen yet. In [109] the authors propose a 2-hop forwarding approach in a theoretical framework where nodes have infinite buffer. The message can be forwarded to one intermediate node before it reaches the destination. A Context-Aware Routing (CAR) algorithm is proposed in [110] to provide asynchronous communication in partially-connected mobile ad-hoc networks. In [111], the authors consider highly mobile nodes that are interconnected via wireless links. Such a network can be used as a transit network to connect other disjoint ad-hoc networks. Five opportunistic forwarding schemes are studied and compared therein. [112] proposes a Message Ferrying (MF) approach for sparse mobile ad-hoc networks, where network partitions can last for a significant period. The basic idea is to introduce deterministic nodal movement and exploit such non-randomness to help data delivery. In PROPHET [113], each node maintains a delivery predictability vector, which indicates its likelihood to meet other nodes. The messages can then be forwarded from the low-predictability nodes to the high-predictability nodes. This simple approach may result in high overhead due to the maintenance of delivery predictability vector and the excessive message copies generated during forwarding. [114] studies the human mobility patterns. It reveals that some nodes are more likely to meet with each other so that the network may be better described by a community model. [115]
propose the Mobile Relay Protocol (MRP) which integrates traditional routing and message storage in the network. If the source node cannot find the route to the destination, it broadcast the message to its neighbors, which will delivery the message, if there is any route to the destination, or store it in its buffer. The authors in [116] present an experimental study on feasibility of using user mobility and opportunistic direct contact to form an opportunistic network. The results show that user mobility can potentially be used to form a network. [117] proposes four types of message dropping strategies for deciding which bundles to exchange when two nodes meet. [118] studies the sociological movement pattern of mobile users and proposes a series of sociological orbit based routing protocols.
Chapter 3
Data Delivery in DFT-MSN

This chapter focuses on developing effective and efficient data delivery schemes for DFT-MSN. We first present an overview of DFT-MSN. Then we study two basic approaches, namely, direct transmission and flooding, by analyzing their performance using queuing theory and statistics. Based on the analytic results that show the tradeoff between data delivery delay/ratio and transmission overhead, we introduce an optimized flooding scheme that minimizes transmission overhead in flooding. Finally, we propose two simple and effective DFT-MSN data delivery schemes, namely Replication-Based Efficient Data Delivery Scheme (RED) and Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD).

3.1 An Overview of DFT-MSN

The proposed DFT-MSN has a two-layer hierarchical architecture. The higher layer is a backbone network, which can be either a wireless LAN or wired network with wireless access interface. The lower layer consists of two types of nodes, the wearable sensor nodes and the high-end sink nodes. The former are attached to mobile objects, gathering target information and forming a loosely connected mobile sensor network for information delivery (see Fig. 3.1 for mobile sensors $S_1$ to $S_{10}$ scattered in the field, where only $S_2$ and $S_3$, $S_4$ and $S_5$, and $S_6$ and $HES_2$ can communicate with each other at this moment). Since the transmission range of a sensor is usually short, it cannot deliver the collected data to the destination (e.g., a data server) directly. As a result, a number of high-end nodes, either mobile nodes (e.g., mobile phones or personal digital assistants with sensor interfaces) or static nodes (e.g., personal computers), are deployed at strategic locations with high visiting probability or carried by a subset of people, serving as the
FIGURE 3.1. An overview of DFT-MSN architecture. $S_1$-$S_{10}$: sensors; $HES_1$-$HES_3$: high end sensors (sinks); $AP_1$-$AP_8$: access points of backbone network.

$sinks$ to receive data from wearable sensors (see $HES_1$ $HES_2$ and $HES_3$ in Fig. 3.1). The high end sink nodes are assumed to have sufficient power compared with the sensor nodes, and thus capable of doing further processing on the collected data, such as aggregation, fusing, filtering, etc. Meanwhile, the high end nodes are equipped with more powerful wireless transceiver so that they can forward the data to the end user through the backbone network directly. With its self-organizing ability, DFT-MSN is established on an ad hoc basis without pre-configuration.

Although it is composed of similar hardware components, DFT-MSN distinguishes itself from conventional sensor networks by the following unique characteristics.
• **Nodal mobility**: The sensors and the sinks are attached to people with various types of mobility. Thus the network topology is dynamic (similar to the mobile ad-hoc network).

• **Sparse connectivity**: The connectivity of DFT-MSN is very low, forming a sparse sensor network where a sensor is connected to other sensors only occasionally.

• **Delay tolerability**: Data delivery delay in DFT-MSN is high, due to the loose connectivity among sensors. Such delay, however, is usually tolerable by the applications that aim at pervasive information gathering from a statistic perspective.

• **Fault tolerability**: Redundancy (e.g., multiple copies of a data message) may exist in DFT-MSN during data acquisition and delivery. Thus, a data message may be dropped without degrading the performance of information gathering. Similar idea has also been found in [119] to deal with the trade-off between transport reliability and energy consumption in sensor networks.

• **Limited buffer**: Similar to other sensor networks, DFT-MSN consists of sensor nodes with limited buffer space. This constraint, however, has a higher impact on DFT-MSN, because the sensor needs to store data messages in its queue for a much longer time before sending them to another sensor or the sink, exhibiting challenges in queue management.

In addition, DFT-MSN exhibits critical needs in data filtering and aggregation due to the likely existence of multiple copies of a data message, especially at sinks. It also shares characteristics of other sensor networks such as a short radio transmission range, low computing capability, and limited battery power. The major differences between DFT-MSN and traditional sensor networks are summarized in Table 3.1.

These characteristics make the mainstream approaches used for data delivery in traditional wireless sensor networks ineffective and inefficient in DFT-MSN, since there is
TABLE 3.1. Summary of differences between DFT-MSN and conventional WSN

<table>
<thead>
<tr>
<th></th>
<th>Nodal mobility</th>
<th>Network topology</th>
<th>Node density</th>
<th>Network connectivity</th>
<th>Delay tolerance</th>
<th>Fault tolerance</th>
<th>Impact of queue size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFT-MSN</td>
<td>Various nodal speeds</td>
<td>Highly dynamic</td>
<td>Sparse and unbalanced</td>
<td>Connectivity is low and unbalanced</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Conventional WSN</td>
<td>Static</td>
<td>Stable</td>
<td>Dense</td>
<td>Fully connected</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

no end-to-end connection between the sensor nodes and the sink nodes at any specific moment, and the topology of the network can be changing consistently. Therefore, designing an effective data delivery scheme, which accommodates these unique features of DFT-MSN, becomes challenging.

DFT-MSN is fundamentally an opportunistic network, where communication links exist with certain probabilities. In such a network, replication is necessary for data delivery in order to achieve certain success ratio [120]. Clearly, replication also increases the transmission overhead. Thus it is a key issue for a DFT-MSN data delivery scheme to properly deal with the trade-off between data delivery ratio/delay and overhead. In the following sections, we first study two basic data delivery approaches, i.e., the direct transmission and flooding. Then we present our proposed data delivery schemes.

### 3.2 Studies of Two Basic Approaches

We first study two basic approaches and analyze their performance. Without loss of generality, we consider a network that consists of $N$ sensors and $n$ sink nodes uniformly distributed in an area of $1 \times 1$. We assume that a sensor or a sink has a fixed radio transmission range, forming a radio coverage area denoted by $a$ ($a \ll 1$). We define the *service area* of a sink node to be its radio coverage area (i.e., $a$). The total service area of all sink nodes in the network is denoted by $A$ ($A < 1$). Clearly, $A = 1 - (1 - a)^n$. Given the
very short radio transmission range and the small number of sinks, the probability that two or more sinks share an overlapped service area is low. Thus $A = 1 - (1 - a)^n \approx na$.

### 3.2.1 Basic Approach I: Direct Transmission

The Basic Approach I is a direct transmission scheme, where a sensor transmits directly to the sink nodes only. More specifically, assume that the generated data message is inserted into a first come first serve (FCFS) queue. Whenever the sensor meets a sink, it transmits the data messages in its queue to the sink. A sensor does not receive or transmit any data messages of other sensors.

The sensors are usually activated and deactivated periodically. For analytic tractability, we assume the sensor’s activation period to be an exponentially distributed random variable with a mean of $T$. The sensor performs sensing and generates one data message upon waking up in each period. In addition, we assume the length of the message is equal to a constant of $L$. Since the activation period is exponentially distributed, the message arrival is a Poisson process with an average arrival rate of $\lambda = 1/T$. The service rate, $\mu$, depends on the available bandwidth ($w$) between a sensor and a sink and the probability ($p$) that a sensor is able to communicate with the sink. To facilitate our illustration, we first assume the bandwidth to be a constant. Possible bandwidth variation due to channel contention will be considered later in this section. Since the sensors and the sink nodes are uniformly distributed, the probability that a sensor is within the coverage of at least one sink node is determined by the total service area of all sink nodes, i.e.,

$$p = A = 1 - (1 - a)^n \approx na.$$ We now prove that the service time is a random variable with Pascal distribution.

**Lemma 1.** Given a constant message length of $L$, a fixed channel bandwidth of $w$ (per time slot), and a service probability of $p$, the service time of the message is a random variable with Pascal distribution.
Proof. Denote a random variable $X$ to be the service time. Let $s$ be the number of time slots required to transmit a message if the node is within the service area. With constant message length $L$ and fixed bandwidth $w$, we have $s = {L \over w}$. In each time slot, a node has the probability of $p$ to be within the service area. Thus, the distribution function of $X$, i.e., the probability that the message can be transmitted within no more than $x$ time slots, is

\[
F_X(x) = \sum_{i=0}^{x-s} \binom{s+i-1}{s-1} p^s (1-p)^i.
\]  

This is the Pascal distribution, with mean value of $s \over p$ and variation of $s \over p^2$. \(\square\)

1) Infinite Buffer Space

We first assume that the sensor has infinite buffer space. With a Poisson arrival rate and a Pascal service time, data generation and transmission can be modelled as an M/G/1 queue, with $\lambda = {1 \over T}$ and $\mu = p \over s = Aw \over L$. In order to arrive at the steady state, we have $\lambda < \mu$, leading to the minimum service area,

\[
A > {L \over T \times w}.
\]  

In other words, the queue will be built up to infinite length if the service area is less than $L \over T \times w$.

For given message arrival rate $\lambda$ and service rate $\mu$, we can derive the average number of messages (including the one currently being served) at a sensor,

\[
q = \rho + \rho^2 + \lambda^2 \times \rho^2 \times {1 \over 2 \times (1-\rho)},
\]  

where $\rho = \lambda \over \mu$, and the average message delivery delay of,

\[
\omega = {q \over \lambda}.
\]
FIGURE 3.2. Performance of direct transmission with \textit{infinite} buffer space under $N = 100$, $n = 10$, $T = 50$, $w = 150$, $a = 0.0314$.

Assume each sensor consumes $J$ Joule to transmit a message and ignore the data processing power. The average power consumption to deliver a message to the sink is,

$$E = J.$$ \hspace{1cm} (3.5)

2) Finite Buffer Space

With finite buffer space (e.g., by assuming each sensor able to keep maximum $K$ messages in its queue), the data generation and transmission can be modelled as an
FIGURE 3.3. Performance of direct transmission with finite buffer space under $N = 100$, $n = 10$, $K = 20$, $T = 20$, $w = 20$, $a = 0.0314$. 
M/G/1/K queue. The message arrival rate ($\lambda$) and the service rate ($\mu$) are calculated in the same way as discussed for the case of infinite buffer space. Now we derive the steady state probabilities of this M/G/1/K queue. Let $k_n$ denote the probability of $n$ arrivals during the period for serving a message. According to the Poisson distribution of message arrival, we have

$$k_n = \sum_{t=s}^{\infty} \frac{e^{-\lambda t} (\lambda t)^n}{n!} \times \binom{t-1}{s-1} p^s (1-p)^{t-s}.$$  

(3.6)

Let $\pi_i$ denote the probability that the system size (i.e., the remaining number of messages right after the current message being served) is $i$. Then, the stationary equations are

$$\pi_i = \begin{cases} 
\pi_0 k_i + \sum_{j=1}^{i+1} \pi_j k_{i-j+1}, & (i = 0, 1, \cdots, K-2) \\
1 - \sum_{j=0}^{K-2} \pi_j, & (i = K-1). 
\end{cases}$$  

(3.7)

Plugging Equation (3.6) into Equation (3.7), we obtain $K$ equations with $K$ unknowns. Solving them, we arrive at $\{\pi_i \mid 0 \leq i \leq K-1\}$. Thus, the average number of messages (including the one currently being served) at a sensor is

$$q = \sum_{i=0}^{K-1} i \pi_i.$$  

(3.8)

Note that since the buffer space is limited, a fraction of messages are dropped upon arrival. Denote $q'_i$ to be the probability that an arriving message finds a system with $i$ messages. Then $q'_K$ is the message dropping probability,

$$q'_K = \frac{\rho - 1 + \frac{\pi_0}{\pi_0 + \rho}}{\rho},$$  

(3.9)

where $\rho = \frac{\lambda}{\mu}$. Since dropped messages do not join the queue, the effective message arrival rate is

$$\lambda_e = \lambda (1 - q'_K).$$  

(3.10)

Thus, the average message delivery delay equals

$$\omega = \frac{q}{\lambda_e}.$$  

(3.11)
3) Further Discussion

If the service area of a sink (i.e., $a$) is large, multiple nearby sensors may transmit at the same time. Thus the channel bandwidth, $w$, is not a constant. As a result, this is no longer a Markov process. If we consider the average service time only, however, we may still use the queuing models discussed in Secs. 3.2.1 to obtain approximate results.

Assume total available bandwidth $W$ is shared by all sensors that are in the service area of a sink. The average data transmission rate of a sensor is $w = \frac{W}{L} \times \frac{1}{1+(N-1)a_{\mu}^{\lambda}}$, where $\frac{\lambda}{\mu}$ is the probability that a sensor has data messages in its queue and accordingly $1 + (N - 1)a_{\mu}^{\lambda}$ is the average number of active sensors that transmit to the sink. Therefore,

$$\mu = \frac{wp}{L} = \frac{pW}{L} \times \frac{1}{1 + (N - 1)a_{\mu}^{\lambda}}, \quad (3.12)$$
i.e.,

$$\mu = \frac{pW}{L} - (N - 1)a_{\mu}^{\lambda}. \quad (3.13)$$

The validity of above analytic models will be discussed next.

4) Numeric Results

We have carried out simulations to validate our analytic models. The network is deployed in an area of $100 \times 100$ m$^2$, and the transmission range of each node is 9 m. For simplicity, the sink nodes are placed far away from each other so that there is no overlap among their service areas. The sensor nodes and the sink nodes are all moving randomly. Other simulation parameters are shown in the captions of Figs. 3.2 and 3.3.

Fig. 3.2 depicts the results for the network with infinite buffer space. As can be seen, the analytic results match the simulation results very well. With an increase in message length, the traffic load increases, thus resulting in a longer average system size (i.e., the total number of messages that are currently being served or waiting in the queue) and a longer average message delivery delay.
For the network with finite buffer space, we also observe a good match between simulation and analytic results (see Fig. 3.3). Since the buffer size is limited, a fraction of arrival traffic is dropped when the queue is full. As a result, the average system size is smaller compared with the case of infinite buffer space. The message dropping rate increases with the message length.

3.2.2 Basic Approach II: Flooding

The second basic approach is flooding. We first discuss the simple flooding scheme and then introduce an optimized flooding scheme.

1) Simple Flooding

In the simple flooding scheme, a sensor always broadcasts the data messages in its queue to nearby sensors, which receive the data messages, keep them in queue, and rebroadcast them. Intuitively, this approach achieves a lower data delivery delay at the cost of more traffic overhead and energy consumption.

Similar queuing models as discussed in Sec. 3.2.1 can be employed for analyzing this flooding approach. Compared with Basic Approach I where message arrival depends on message generation only, a sensor in the flooding approach not only generates its own data messages but also receives messages from other sensors, resulting in a higher $\lambda$. On the other hand, since a sensor may transmit to other sensors in addition to the sinks, the service rate is also higher. The queue length and queuing delay can be derived accordingly.

In the Basic Approach I, the queuing delay is the same as the data message delivery delay because a sensor transmits to the sink nodes only. In the flooding approach however, they are different, due to the duplicate messages at multiple sensor nodes. To analyze the message delivery delay, we consider a data message generated by a sensor with infinite buffer space. For simplicity, we assume the sensor’s activation period to be a constant $T$,
within which the sensor can transmit its messages to its neighbors that are activated at the same time. We assume the bandwidth is high enough such that the sensor can always transmit its data messages when it meets other active sensors or the sink nodes. We also assume the mobility is high enough and the network is large enough such that the sensor always meets different neighbors when it wakes up. We study a sequence of activation periods after the message is generated. \( p \) is the probability that a sensor can communicate with at least one sink node when it is activated. As we have discussed in Sec. 3.2.1, \( p = A = 1 - (1 - a)^n \approx na \). Denote \( p_j \) to be the probability that the message is not delivered to the sink nodes in the first \( j - 1 \) periods and at least one copy of the message is delivered to the sink in the \( j^{th} \) period. Let \( N_j \) denote the number of sensors that have a copy of the message in the \( j^{th} \) period if the message has not been delivered to the sink. \( N_j \) is calculated as follows:

\[
N_j = \begin{cases} 
(N - 1)a + 1, & j = 1 \\
(N - N_{j-1})(1 - (1 - a)^{N_{j-1}}) + N_{j-1}, & j > 1.
\end{cases}
\]

(3.14)

Consequently, \( p_j \) is derived below,

\[
p_j = \begin{cases} 
p, & j = 1 \\
(1 - (1 - p)^{N_{j-1}})(1 - \sum_{i=1}^{j-1} p_i), & j > 1.
\end{cases}
\]

(3.15)

Thereupon, the average delay of delivering the data message is expressed by

\[
\omega = T \sum_{j=1}^{\infty} j \times p_j.
\]

(3.16)

Note that, when \( N_1 = N_2 = \ldots = 1 \), the above analysis turns into an alternative model for the Basic Approach I, where the sensor transmits its data messages to the sink directly, and thus there is only a single copy of a message in the network.

Since many copies of a given message exist in the network and a sensor is not aware whether the sink has received it or not, the message is eventually received and transmitted
FIGURE 3.4. Performance of flooding schemes.

once by every sensor node, resulting in a total of $N$ copies. Accordingly, the average power consumption per message is proportional to the network size, i.e.,

$$E = O(J \times N).$$  \hspace{1cm} (3.17)

2) Optimized Flooding

In the simple flooding scheme, each sensor aggressively propagates its data messages to any neighboring nodes, resulting in the lowest delivery delay. At the same time, however, it also incurs very high overhead (i.e., the number of message copies) and energy
consumption. Here we introduce an optimized flooding scheme that may significantly reduce flooding overhead and energy consumption.

The basic idea of the optimized flooding scheme is to estimate the message delivery probability and stop further propagation of a message if its delivery probability is already high enough in order to reduce transmission overhead. Similar to our discussion on simple flooding, we consider a sequence of activation periods. Assume the message’s propagation is terminated after period \( d \) (i.e., the sensor that has a copy of the message does not transmit it to any other nodes except the sinks after the \( d^{th} \) period). Our objective is to minimize \( d \) such that the message delivery probability in total \( D (D \geq d) \) periods is higher than a given threshold, i.e., \( p_{D} \geq \gamma \).

Since the sensors stop broadcasting the message after \( d \) periods, \( N_j \) is given by,

\[
N_j = \begin{cases} 
    (N - 1)a + 1, & j = 1 \\
    (N - N_{j-1})(1 - (1 - a)^{N_{j-1}}) + N_{j-1}, & d \geq j > 1 \\
    N_d, & j > d.
\end{cases}
\]  

(3.18)

Similar to the analysis for simple flooding, \( p_j = [1 - (1 - p)^{N_{j-1}}](1 - \sum_{i=1}^{j-1} p_i) \) with \( p_1 = p \).

For a given threshold \( \gamma \), one can derive the minimum \( d \) such that \( p_{D} \geq \gamma \). Accordingly, the average delay is \( \omega = T \sum_{j=1}^{\infty} j \times p_j \).

After determining the optimal value of \( d \), we can estimate the average number of message copies made during the \( d \) periods, \( M_d \). Note that \( N_j \) is the number of copies in the \( j^{th} \) period, given that the message has not been delivered to the sink in the first \( j - 1 \) periods. Thus, \( N_d \) is not equivalent to \( M_d \). Since the message is not propagated any more after the \( d^{th} \) period, the number of copies reaches its maximum at the \( d^{th} \) period. Let \( U_j \) denote the number of nodes which have a copy of the message but have not transmitted to the sink nodes yet at the \( j^{th} \) period, and \( V_j \) denote the number of copies that have been...
sent to the sinks. We have

\[
U_j = \begin{cases} 
  (N - 1)(1 - (1 - a)^{1-p}) + 1 - p, & j = 1 \\
  (1 - p)U_{j-1} + (N - U_{j-1} - V_{j-1}) \\
  \times (1 - (1 - a)^{(1-p)U_{j-1}}), & d \geq j > 1 
\end{cases}
\]  

(3.19)

and

\[
V_j = \begin{cases} 
  p, & j = 1 \\
  V_{j-1} + p \times U_{j-1}, & d \geq j > 1. 
\end{cases}
\]  

(3.20)

Therefore, the average number of message copies made during the \(d\) periods is

\[
M_d = U_d + V_d. 
\]  

(3.21)

3) Numeric Results

We have simulated and compared the two flooding approaches discussed above. The network is deployed in an area of \(100 \times 100 \, m^2\) with 3 sink nodes, and the transmission range of each node is 9 \(m\). For simplicity, the sensor nodes and the sink nodes are all randomly moving, and the message buffer of each sensor is large enough so that no message is dropped. \(\gamma = 0.7\) and \(D = 5\).

Fig. 3.4 compares analytic and simulation results of both approaches. As can be seen, the simulation results and the analytic results match well. As shown in Fig. 3.4(a), the message delivery delay of both approaches decreases slightly with increase in network density. This is somewhat expected because under higher network density, the message is broadcasted to more neighbors and thus is propagated faster. We notice that the message delay of the optimized flooding is slightly higher than that of the simple flooding approach because the sensors stop forwarding the message after \(d\) periods. At the same time, the optimized flooding scheme introduces much fewer duplicate messages compared with its
simple flooding counterpart (see Fig. 3.4(b)) and thus significantly reduces energy consumption. The increase of network density leads to a linear increase in the number of duplicate messages when simple flooding is employed. In contrast, the number of duplicate messages of the optimized flooding approach increases only marginally, because \(d\) is optimized to lower flooding overhead.

### 3.2.3 Observation from The Two Basic Approaches

We have studied two basic approaches so far. The direct transmission approach minimizes transmission overhead (i.e., the number of message copies) and energy consumption, at the expense of a long message delivery delay (with large buffer space) or a low message delivery ratio due to a high message dropping rate (with small buffer space). In contrast, the flooding approach minimizes the message delivery delay. At the same time however, it results in very high transmission overhead and energy consumption. Note that, although the optimized flooding scheme may significantly reduce the number of message copies, it is based on the assumptions of unlimited buffer space and globally synchronized activation periods. Those assumptions usually don’t hold in practical DFT-MSNs.

DFT-MSN is fundamentally an opportunistic network, where replication is necessary for data delivery, though at the expense of increased transmission overhead, in order to achieve certain success ratio. An efficient DFT-MSN data delivery scheme will take into consideration the tradeoff between delivery delay/ratio and transmission overhead/energy. In particular, the following three key issues need to be addressed.

- **When to transmit data messages?** When a sensor moves into the communication range of another sensor, it needs to decide whether to transmit its data messages or not, in order to achieve a high message delivery ratio, and at the same time, minimize transmission overhead.
• *Which messages to transmit?* The data messages generated by the sensor itself or received from other sensors are put into the sensor’s data queue. After deciding to initiate data transmission, the sensor needs to determine which messages to transmit if there are multiple messages with different degrees of importance in its queue.

• *Which messages to drop?* A data queue has a limited size. When it becomes full (or due to other reasons as to be discussed later), some messages have to be dropped. The sensor needs to decide which messages to drop according to their importance in order to minimize data transmission failure.

With the above issues taken into consideration, we will propose two data delivery schemes for DFT-MSN in the next two sections, namely Replication-Based Efficient Data Delivery Scheme (RED) [121] and Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD) [1]. Both schemes aim to minimize the replication overhead, while achieving the required data delivery probability. In the former scheme, the replication is done by the source node via erasure coding. In the latter scheme, a message is replicated dynamically according to its fault tolerance by the source and the intermediate nodes.

### 3.3 Replication-Based Efficient Data Delivery Scheme (RED)

The proposed replication-based efficient data delivery scheme (RED) consists of two key components for data delivery and message management, elaborated below.

#### 3.3.1 Data Delivery

1) *Nodal Delivery Probability*

The decision on data transmission is made based on *delivery probability*, which indicates the likelihood that a sensor can deliver data messages to the sink. Note that the delivery probability is *not* simply the probability that a node meets the sinks.
Let $\xi_i$ denote the delivery probability of a sensor $i$. $\xi_i$ is initialized with zero and updated upon an event of either message transmission or timer expiration. More specifically, the sensor maintains a timer. If there is no message transmission within an interval of $\Delta$, the timer expires, generating a timeout event. The timer expiration indicates that the sensor couldn’t transmit any data messages during $\Delta$, and thus its delivery probability should be reduced. Whenever sensor $i$ transmits a data message to another node $k$, $\xi_i$ should be updated to reflect its current ability in delivering data messages to the sinks. Note that since end-to-end acknowledgement is not employed in DFT-MSN due to its low connectivity, sensor $i$ doesn’t know whether the message transmitted to node $k$ will eventually reach the sink or not. Therefore, it estimates the probability of delivering the message to the sink by the delivery probability of node $k$, i.e., $\xi_k$. More specifically, $\xi_i$ is updated as follows,

$$
\xi_i = \begin{cases} 
(1 - \alpha)\xi_i + \alpha \xi_k, & \text{Transmission} \\
(1 - \alpha)\xi_i, & \text{Timeout}, 
\end{cases}
$$

where $[\xi_i]$ is the delivery probability of sensor $i$ before it is updated, and $0 \leq \alpha \leq 1$ is a constant employed to keep partial memory of historic status. If $k$ is the sink, $\xi_k = 1$, because the message is already delivered to the sink successfully. Otherwise, $\xi_k < 1$. Clearly, $\xi_i$ is always between 0 and 1.

2) Data Transmission

The data messages are maintained in a first-in-first-out queue. The transmission is simple. Without loss of generality, we consider a sensor $i$, which has a message at the top of its data queue ready for transmission and is moving into the communication range of a set of sensors. Sensor $i$ first learns their delivery probabilities and available buffer spaces
via simple handshaking messages. Then, sensor $i$ transmits its message to the neighbor $j$, who has the highest delivery probability ($\xi_j > \xi_i$) and available buffers.

3) Further Discussion

During our experiments, we have found two potential inefficiencies in data delivery, stemming from the approach for updating nodal delivery probability. The first problem is mutual reference. Assume a node $j$ with a slightly higher delivery probability than that of a node $i$. When nodes $i$ and $j$ are within transmission range, node $i$ transmits a message to node $j$, based on the data delivery scheme discussed above. Then $\xi_i$ increases due to a successful transmission. In the worst case, node $j$’s delivery probability decreases because of timeout, and thus $\xi_j$ becomes lower than $\xi_i$. Consequently, node $j$ may transmit messages back to node $i$ in the next transmission period, incurring fluctuation and unnecessary transmission overhead. To address this problem, we establish Lemma 2 as follows.

**Lemma 2.** For two nodes $i$ and $j$ with delivery probability $\xi_i$ and $\xi_j$ (assume $\xi_j \geq \xi_i$), $\xi_i < \frac{2-2\alpha \xi_j}{2-\alpha}$ is the necessary and sufficient condition to avoid the mutual reference problem.

**Proof.** We prove the necessary condition first. Assume there exists a $\delta$ so that mutual reference problem can be avoided if $\xi_j \geq \xi_i + \delta$. Thus after node $i$ transmits a message to node $j$, the following inequation must hold:

$$\xi_j' - \xi_i' > -\delta,$$

where $\xi_i'$ and $\xi_j'$ are the delivery probability of nodes $j$ and $i$ updated according to Equation 3.22 after the transmission. In the worst case, $\xi_i' = (1-\alpha)\xi_i + \alpha \xi_j$ and $\xi_j' = (1-\alpha)\xi_j$. Plugging $\xi_j'$ and $\xi_i'$ into Equation (3.23), we have

$$(1-\alpha)\xi_j - (1-\alpha)\xi_i - \alpha \xi_j > -\delta,$$

42
and thus arrive at
\[ \delta > \frac{\alpha}{2 - \alpha} \xi_j. \]  
(3.25)

Therefore, we obtain the necessary condition: to avoid mutual reference problem, node \( i \) should transmit to node \( j \) only if
\[ \xi_i < \frac{2 - 2\alpha}{2 - \alpha} \xi_j. \]  
(3.26)

Similarly, we can show the sufficient condition: if \( \xi_i < \frac{2 - 2\alpha}{2 - \alpha} \xi_j, \xi_j' - \xi_i' > -\delta \) holds and thus the mutual reference problem is avoided.

Another problem is unnecessary propagation. When a node meets a neighbor with higher delivery probability, it always sends messages to this neighbor according to the data transmission scheme discussed above, even when itself already has a large enough probability to reach the sink node directly. This results in extra transmission and energy consumption. To avoid unnecessary propagation, each node maintains an additional parameter called direct delivery probability denoted by \( \psi \), which indicates how likely this node can transmit the messages directly to the sink node. If \( \psi \) is larger than a predefined threshold, the node only transmits messages to the sink node directly.

### 3.3.2 Message Management

As we have discussed in Sec. 3.2.3, replication is usually employed to improve data delivery ratio and/or reduce data delivery delay in opportunistic networks. In this research, we propose an erasure-coding approach tailored for DFT-MSN, which efficiently addresses the tradeoff between delivery ratio/delay and overhead.

In the erasure coding approach [120], a message is first split into \( b \) blocks with equal size. Erasure coding is then applied to these \( b \) blocks, producing \( S \times b \) small messages (which are referred as block messages), where \( S \) is the replication overhead. The gain of erasure coding stems from its ability of recovering the original message based on any \( b \)
FIGURE 3.5. Determining optimal $b$ to minimize $S$ in erasure coding.
block messages. Assume that each block message has a constant delivery probability of $p$, then the delivery probability of the original message is

$$P = \sum_{j=b}^{Sb} \binom{Sb}{j} p^j (1-p)^{Sb-j}. \quad (3.27)$$

Clearly, the erasure coding approach is reduced to simple whole message replication when $b = 1$.

Our objective is to determine the optimal erasure coding parameters (i.e., $b$ and $S$) with given inputs (i.e., $p$), in order to meet the desired message delivery probability (denoted by $H$) while minimizing the transmission overhead.

According to Equation (3.27), we can find the minimum $S$ for given $b$ and $p$, so that $P$ is no less than $H$, i.e.,

$$S(p, b) = \min \{ S \mid \sum_{j=b}^{Sb} \binom{Sb}{j} p^j (1-p)^{Sb-j} \geq H \}. \quad (3.28)$$

In theory $b$ may vary from 1 to the length of the message. But large $b$ results in many small blocks, which also increases the processing overhead and decreases the bandwidth utilization. Thus a minimum block size of $m$ is adopted in our proposed approach. Let $M$ denote the maximum message size. Thus the maximum value of $b$ is: $b_{max} = \frac{M}{m}$. Fig. 3.5 shows $S(p, b)$ with $p$ varying from 0 to 1 and $b$ varying from 1 to $b_{max} = 50$. As we can see, the minimum $S$ generally increases with the decrease of $p$. When $p$ is very small, a large value of $S$ is necessary for achieving the desired delivery ratio, i.e., $H$. This, however, may degrade the network performance due to the overwhelming overhead. Therefore, an upper bound of $S$ (denoted by $S_{max}$) is enforced in our implementation. This approach is particularly useful in situations where nodes temporarily have very low delivery ratio (for example, at the beginning of deployment).

The solid curve on the surface of Fig. 3.5 indicates the minimum $S$ for each given $p$, while the dashed curve, which is the projection of the solid curve on the horizontal plane, for simplicity, we assume the optimal erasure coding is used.
indicates the optimal $b$ for each $p$ in order to minimize $S$. We notice that when $p$ is very low (less than a $\beta_1$, which is around 0.2) or very high (larger than a $\beta_2$ whose value depends on $H$), the optimal value of $b$ is always 1, which means that the whole message replication is preferable. $\beta_2$ usually increases with $H$. For example, as can be observed by the comparison of Fig. 3.5(a) and Fig. 3.5(b), $\beta_2$ increases from 0.7 to 0.9 when $H$ increases from 0.9 to 0.99. When $p$ is between $\beta_1$ and $\beta_2$, the optimal $b$ varies widely and non-monotonously between 1 and $b_{max}$, depending on the value of $p$. In our implementation, a list of $p$ values and their corresponding optimal $b$ values are kept in a table. Whenever a node $i$ generates a data message, it checks its current delivery probability, $\xi_i$. Let $p = \xi_i$, node $i$ looks up the table to determine the optimal $b$ and minimum $S$, and accordingly encodes the data messages into $S \times b$ data blocks, which are then put to the queue independently for future transmission.

### 3.4 Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD)

The RED data delivery scheme proposed in Sec. 3.3 employs the erasure coding to improve delivery ratio. Its advantage is simplified message manipulation and queue management at intermediate nodes, since all computation is done by the source node. At the same time, however, the optimization of erasure coding parameters is usually inaccurate because they are calculated according to the current data delivery probability of source node, especially when the source is very far away from the sinks. In addition, propagating many small messages in the network may incur further processing overhead and inefficiency of bandwidth utilization. In this section, we propose a Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD), in order to avoid the above problems, at the expense of increased complexity in message and queue management.
The proposed FAD data delivery scheme depends on two important parameters, namely, the nodal delivery probability and the message fault tolerance. The former has been discussed in Sec. 3.3.1 for the RED scheme. The latter indicates the amount of redundancy and the importance of a message. We first introduce the definition and updating algorithm of message fault tolerance. Then, the FAD data delivery scheme is elaborated.

3.4.1 FAD Parameter: Message Fault Tolerance

Unlike the RED data delivery scheme discussed in Sec. 3.3 (and most other typical data transmission schemes) where the packets are deleted from the buffer after they are transmitted to the next hop successfully, a sensor employing FAD may still keep a copy of the message after its transmission to other sensors. Therefore, multiple copies of the message may be created and maintained by different sensors in the network, resulting in redundancy. The fault tolerance degree (FTD) is introduced to represent the amount of redundancy and to indicate the importance of a given message. We assume that each message copy carries a field that keeps its FTD. Let \( F_j^i \) denote the FTD of message \( j \) in the queue of sensor \( i \). The FTD of a message copy is defined to be the probability that at least one copy of the message is delivered to the sink by other sensors in the network. When a message is generated, its FTD is initialized to be zero. Let’s consider a sensor \( i \), which is multicasting a data message \( j \) to \( Z \) nearby sensors, denoted by \( \Xi = \{\psi_z \mid 1 \leq z \leq Z\} \). The multicast transmission essentially creates totally \( Z + 1 \) copies. An appropriate FTD value needs to be assigned to each of them. More specifically, the message transmitted to sensor \( \psi_z \) is associated with a FTD of \( F_{\psi_z}^j \),

\[
F_{\psi_z}^j = 1 - (1 - [F_j^i])(1 - \xi_z) \prod_{m=1, m \neq z}^{Z}(1 - \xi_{\psi_m}),
\]

and the FTD of the message at sensor \( i \) is updated as

\[
F_i^j = 1 - (1 - [F_i^j]) \prod_{m=1}^{Z}(1 - \xi_{\psi_m}),
\]
where \( F_j^i \) is the FTD of message \( j \) at sensor \( i \) before multicasting. The above process repeats at each time when message \( j \) is transmitted to another sensor node. In general, the more times a message has been forwarded, the more copies of the message are created, thus increasing its delivery probability. As a result, it is associated with a larger FTD.

### 3.4.2 FAD Data Delivery Scheme

The proposed FAD data delivery scheme consists of two components for queue management and data transmission, discussed below.

1) **Queue Management**

   Compared to the simple first-in-first-out queue in RED, the data queue management in FAD is more complicated. Each sensor has a data queue that contains data messages ready for transmission. The data messages of a sensor come from three sources. (1) After the sensor acquires data from its sensing unit, it creates a data message, which is inserted into its data queue; (2) When the sensor receives a data message from other sensors, it inserts the message into its data queue; (3) After the sensor sends out a data message to a non-sink sensor node, it may also insert the message into its own data queue again, because the message is not guaranteed to be delivered to the sink. The queue management is to appropriately sort the data messages in the queue, to determine which data message to be sent when the sensor meets another sensor, and to determine which data message to be dropped when the queue is full.

   Our proposed queue management scheme is based on the fault tolerance, which signifies how important the messages are. The message with smaller FTD is more important and should be transmitted with a higher priority. This is done by sorting the messages in the queue with an increasing order of their FTD. Message with the smallest FTD is always at the top of the queue and transmitted first. A message is dropped at the
following two occasions. First, if the queue is full when a message arrives, its FTD is compared with the message at the end of the queue. If the new message has a larger FTD, it is dropped. Otherwise, the message at the end of the queue is dropped, and the new message is inserted into the queue at appropriate position according to its FTD. Second, if the FTD of a message is larger than a threshold (e.g., $\mathcal{R}$), the message is dropped, even if the queue is not full. This is to reduce transmission overhead, given that the message will be delivered to the sinks with a high probability by other sensors in the network. A special example is the message which has been transmitted to the sink. It will be dropped immediately because it has the highest FTD of 1.

With the above queue management scheme, a sensor can determine the available buffer space in its queue for future arrival messages with a given FTD. Assume a sensor has a total queue space for at most $K$ messages. Let $k_i^m$ denote the number of messages with a FTD of $m$ in the queue of sensor $i$ (where $0 \leq m \leq 1$). Then, the available buffer space at sensor $i$ for new messages with FTD $x$ is $B_i(x) = K - \sum_{m=0}^{x} k_i^m$. If $B_i(x) = 0$, any arrival message with a FTD of $x$ or higher will be dropped. Note that, however, even when the queue is filled by $K$ messages and becomes full, $B_i(x)$ may still be larger than 0, for a small $x$ (i.e., for messages with a low FTD). Buffer space information is important to make decision on data transmission, as discussed next.

2) Data Transmission

A data transmission decision is made based on the delivery probability. Without loss of generality, we consider a sensor $i$, which has a message $j$ at the top of its data queue ready for transmission and is moving into the communication range of a set of $Z'$ sensors. Sensor $i$ first learns their delivery probabilities and available buffer spaces via simple handshaking messages. Let $\Xi' = \{\psi_z \mid 1 \leq z \leq Z'\}$ designate the $Z'$ sensors, sorted by a decreasing order of their delivery probabilities. Sensor $i$ multicasts its message $j$ to a
subset of the $Z'$ sensors, denoted by $\Phi$, which is determined by the Algorithm 1, where $\mathcal{R}$ is a threshold, $F_{ij}^j$ is the FTD of the message $j$ at sensor $i$, and $B_{\psi_z}(F_{ij}^j)$ is the number of available buffer slots at Node $\psi_z$ for messages with FTD $F_{ij}^j$.

**Algorithm 1** Identification of receiving sensors.

$\Phi = \emptyset$.

for $z = 1:Z'$ do

if $\xi_i < \xi_{\psi_z}$ AND $B_{\psi_z}(F_{ij}^j) > 0$ then

$\Phi = \Phi \cup \psi_z$.

end if

if $1 - (1 - F_{ij}^j) \prod_{m \in \Phi} (1 - \xi_m) > \mathcal{R}$ then

Break.

end if

end for

By following Algorithm 1, sensor $i$ sends message $j$ to a set of neighbors with higher delivery probabilities (i.e., $\xi_i < \xi_{\psi_z}$), and at the same time, controls the total delivery probability of message $j$ (i.e., $1 - (1 - F_{ij}^j) \prod_{m \in \Phi} (1 - \xi_m)$) just enough to reach $\mathcal{R}$ in order to reduce unnecessary transmission overhead. In order to avoid unnecessary message drops due to buffer overflow at the receiver, sensor $i$ checks the available buffer space of its neighboring nodes for message $j$ (i.e., $B_{\psi_z}(F_{ij}^j)$) before data transmission.

Clearly, this message transmission scheme is equivalent to direct transmission when the network is just deployed, because the delivery probability is initialized with zero and thus the sensors transmit to the sink nodes only. As the delivery probability is gradually updated with non-zero values, multihop relaying will take place.

### 3.5 Simulation Results

Extensive simulation has been carried out to evaluate the performance of the proposed DFT-MSN data delivery schemes. In our simulation, 3 sink nodes and 100 sensor nodes are randomly deployed in an area of $200 \times 200 \ m^2$. The whole area is divided into 25 non-overlapped zones, each with an area of $40 \times 40 \ m^2$. A sensor node is initially resided
in its home zone. It moves with a speed randomly chosen between 0 and 5 m/s. Whenever a node reaches the boundary of its zone, it moves out with a probability of 20%, and bounces back with a probability of 80%. After entering a new zone, the sensor repeats the above process. However, if it reaches the boundary to its home zone, it returns to its home zone with a probability of 100%. The message size is 200 bits. Each sensor has a maximum transmission range of 10 m and a maximum queue size of 120 whole messages (or a total of 120 × 200 bits). The data generation of each sensor follows a Poisson process with an average arrival interval of 100 s. The channel bandwidth is 10 kbps. The fault tolerance threshold used in FAD scheme is set to be $\Re = 0.9$, while the delivery threshold used in RED scheme is set to be $H = 0.9$. In erasure coding, the maximum replication overhead ($S_{max}$) is set to 3 and the maximum number of blocks ($b_{max}$) is 20. The above default simulation parameters are summarized in Table 3.2.

The sensor node transmits its data messages according to our proposed DFT-MSN data delivery schemes. We first study the effectiveness of delivery probability updating scheme.

<table>
<thead>
<tr>
<th>TABLE 3.2. Default simulation parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum sensor transmission range</td>
</tr>
<tr>
<td>Number of sensor nodes</td>
</tr>
<tr>
<td>Number of sink nodes</td>
</tr>
<tr>
<td>Size of network area</td>
</tr>
<tr>
<td>Size of a zone</td>
</tr>
<tr>
<td>Probability to move out of a zone</td>
</tr>
<tr>
<td>Probability to move back to home zone</td>
</tr>
<tr>
<td>Maximum queue length</td>
</tr>
<tr>
<td>Message generation rate</td>
</tr>
<tr>
<td>Whole message length</td>
</tr>
<tr>
<td>Bandwidth</td>
</tr>
<tr>
<td>Nodal moving speed</td>
</tr>
<tr>
<td>$\Re$</td>
</tr>
<tr>
<td>$H$</td>
</tr>
<tr>
<td>$S_{max}$</td>
</tr>
<tr>
<td>$b_{max}$</td>
</tr>
</tbody>
</table>
For clarity, we set the delivery probability of the sensor into 5 discrete levels. Level $i$ ($1 \leq i \leq 5$) represents the successful delivery probability between $(i - 1) \times 0.2$ and $i \times 0.2$.

Fig. 3.6(a) shows DFT-MSN at the initial stage, where each sensor node has a delivery probability of 0. With the proposed protocol running, each node updates its delivery probability. The results after 1000 seconds are illustrated in Fig. 3.6(b), where the nodes closer to the sinks usually have higher delivery probabilities, as expected.

We vary several parameters to observe their impacts on the performance. Fig. 3.7 compares the performance of the proposed two data delivery schemes (i.e., RED and
FAD) with the simple flooding approach and the direct transmission approach, by varying
the number of sink nodes in DFT-MSN. As shown in Fig. 3.7(a), the delivery ratio
increases with more sink nodes being deployed for all approaches. This is reasonable
because the sensors then have high probabilities to reach the sink nodes, thus resulting in
high delivery ratio. The proposed RED and FAD schemes always have higher delivery
ratios than other approaches, especially when a small number of sinks are deployed. As
expected, the flooding approach has a much lower delivery ratio than other approaches
because it generates too many message copies, which leads to excessive buffer overflow
and message dropping. Fig. 3.7(b) demonstrates that the average delay of every approach
decreases quickly with more sink nodes deployed in the network. Although the flooding
approach has the smallest message delivery delay, its delivery ratio is very low. In
addition, since the RED scheme on average transmits more copies for each message than
the FAD scheme, it has a slightly lower delay than the FAD scheme. Clearly, the direct
transmission approach suffers from the longest delay since messages can then be delivered
only when the source node meets the sink.

Energy consumption of the sensor is due mainly to data transmission. Thus, the more
duplicated copies generated, the higher the energy consumption. As depicted in
Fig. 3.7(c), the number of message copies in direct transmission is always 1, since a
sensor always transmits data messages to the sink directly. The results of flooding are not
shown here, because it generates excessive copies which are several orders of magnitude
higher than those of other approaches. The duplicated message number decreases in both
RED and FAD schemes with the increase of sink node density. This is because more sink
nodes may shorten the message transmission path (in terms of number of hops) from the
sensors to the sinks, and accordingly reduce the transmission overhead. Additionally, the
replication parameter $S$ in RED becomes smaller when nodal delivery probability ($p$)
increases according to our discussion in Sec. 3.3.2. This may further compress the
overhead of message transmission in RED scheme. Meanwhile, we have also noticed that the FAD scheme always has smaller overhead than the RED scheme. This is reasonable because the message replication in FAD is done dynamically according to the delivery probabilities of the intermediate nodes and thus more accurate, compared with the RED scheme where replication is performed by the source node only.

We also vary the maximum queue length of each sensor in our simulations, with results presented in Fig. 3.8. With an increase in maximum queue length, the delivery ratio increases for all approaches, as expected (see Fig. 3.8(a)). As shown in Fig. 3.8(b), the queue length doesn’t have a significant impact on the delay of the simple flooding approach, RED scheme and FAD scheme. The delay of the direct transmission approach, however, increases with the longer queue length, because more data messages will then reside in the queue for a longer time before being delivered. It is also noticed that the FAD approach can well control its transmission overhead (i.e., the number of copies generated) even when the available queue size is large. On the other hand, more duplicated copies are generated under the RED approach (see Fig. 3.8(c)) as a result of increasing the maximum queue size.

Fig. 3.9 depicts the impact of nodal moving speed. As the speed increases, the delivery ratios of all approaches but the simple flooding rise, while the delivery delays of all approaches decrease. This is because the node with a higher speed has a better opportunity to meet other nodes and also with higher probability to reach the sink nodes. Thus, the messages have a better chance to be delivered before they are dropped. It is also noticed that the transmission overhead of the proposed FAD and RED schemes decreases slightly with the increase of nodal speed (as shown in Fig. 3.9(c)), making them most suitable for the network with varying nodal speeds.

Fig. 3.10 illustrates the impact of node density by varying the total number of sensor nodes in the network. As shown in Fig. 3.10(a), the RED scheme is sensitive to the change
FIGURE 3.7. Impact of the number of sink nodes.
FIGURE 3.8. Impact of maximum queue length.
of node density. Its delivery ratio first rises and then decreases sharply. This can be explained as follows. At first, the delivery ratio increases because more nodes participate in message relaying so that a message has better chance to reach the sink node. When the node density becomes too high, however, excessive message propagation incurs much more buffer overflow due to limited queue size and limited communication bandwidth, especially for the nodes close to the sinks. As a result, the RED scheme is more suitable for sparse network. In contrast, the FAD scheme has very steady performance with the increase of node density, exhibiting the perfect scalability. The node density doesn’t have significant impact on average overhead in both RED and FAD schemes, as shown in Fig 3.10(c). It is also noticed that data messages can be propagated faster in a network with higher nodal density, thus decreasing average delay in both RED and FAD schemes (if the message is delivered to the sink successfully), as shown in Fig 3.10(b).
FIGURE 3.9. Impact of nodal speed \((m/s)\).
(a) Average delivery ratio.

(b) Average delay.

(c) Average overhead.

FIGURE 3.10. Impact of node density.
Chapter 4
Protocol Design and Optimization for DFT-MSN

Besides the basic data delivery schemes proposed in Chapter 3, we also observe that without end-to-end connections, routing in DFT-MSN becomes localized and ties closely to Layer 2 protocols, which naturally calls for merging Layer 3 and Layer 2 in order to reduce overhead and improve network efficiency [122]. Note that, while cross-layer design has been discussed extensively in the past several years, this work distinguishes itself from others by considering the unique characteristics of the sparsely-connected, delay-tolerant mobile sensor network. The goal of conventional sensor network protocols (e.g., [123, 124, 125]) is to optimize energy consumption with a given delay or throughput requirement, with the sensor nodes usually enjoying stable connectivity and ample channel bandwidth. DFT-MSN, however, is fundamentally an opportunistic network, where the communication links exist only with certain probabilities and are deemed the scarcest resource. A naive approach is to let sensors work aggressively in order to catch every possible opportunity for data transmission. Under such an approach, however, the sensors are likely to drain off their battery quickly, resulting in poor performance of the overall network. Clearly, there is a tradeoff between link utilization and energy efficiency. None of the existing sensor network protocols have considered such a unique network environment and performance tradeoff. Our goal is to make efficient use of the transmission opportunities whenever they are available, while keeping the energy consumption at the lowest possible level.

In this chapter, we develop a cross-layer data delivery protocol for DFT-MSN. To address the tradeoff between data delivery ratio/delay and overhead/energy, we design the data delivery protocol with two phases, i.e., the asynchronous phase and the synchronous...
phase. In the first phase, a sender contacts its neighbors to identify a set of appropriate receivers. Since no central control exists, the communication in the first phase is contention-based. In the second phase, the sender gains channel control and multicasts its data message to the receivers. Furthermore, several optimization issues in these two phases are identified, with possible solutions proposed to reduce the collision probability and improve energy efficiency. Extensive simulations are carried out for performance evaluation. Our results show that the proposed cross-layer data delivery protocol for DFT-MSN achieves a high message delivery ratio with low energy consumption and an acceptable delay.

4.1 Cross-Layer Data Delivery Protocol for DFT-MSN

We propose the data delivery protocol for DFT-MSN by taking a cross-layer approach, aiming to strike the balance between link utilization and energy efficiency. This work is mainly based on the principles of the FAD data delivery scheme proposed in Sec. 3.4. In the proposed data delivery protocol, each sensor has a working cycle that consists of two modes, the sleep mode and the work mode. The length of the working cycle is dynamic, as will be discussed in Sec. 4.2. Without loss of generality, we consider a sensor $i$ with a message $M$ at the top of its data queue. If it does not serve as either a sender or a receiver in the past $L$ transmissions as discussed below, sensor $i$ enters sleep mode for a period of $T_i$. The optimal value of $T_i$ will be discussed later in Sec. 4.2. Upon waking up, Sensor $i$ goes through two phases for possible data transmission to nearby sensors, namely, asynchronous phase and synchronous phase, as elaborated below.

4.1.1 Asynchronous Phase

The asynchronous phase starts after the node wakes up from sleeping. Since no central control exists during this phase, all communications are contention-based. More
FIGURE 4.1. Proposed cross-layer data delivery protocol: (a) Sender; (b) and (c) Qualified neighbors; (d) Unqualified neighbors; (e) Possible new arrivals.

specifically, sensor $i$ turns on its radio and listens for a period of $\tau_i$ (see Fig. 4.1). If the channel is idle, it transmits a preamble to occupy the channel and inform its neighbors to prepare for receiving its RTS (Request-To-Send) packet. If collision happens (i.e., receiving preamble from other nodes), it gives up its attempt for further transmission and restarts the asynchronous phase again.

After the preamble, sensor $i$ sends an RTS packet that contains its nodal delivery probability (i.e., $\xi_i$), the FTD of message $M$ (i.e., $f_i^M$), and the length of the contention window (denoted by $W$). The contention window is the period to allow the neighboring sensors to reply.

Once a neighbor node receives the RTS packet, it checks if it can serve as a qualified receiver. A qualified receiver of node $i$ must satisfy two conditions. First, it must have higher delivery probability than node $i$. Second, it should have available buffer space for messages with FTD of $f_i^M$. Each of the qualified receivers then sends back a CTS
(Clear-To-Send) packet, which includes its nodal delivery probability and available buffer space, to node \( i \) at a random time point during the period of \( W \). Obviously, since there is no central control, the transmission of CTS packet is also contention-based. If more than one CTS arrive at node \( i \) at the same time, all of them are lost.

Note that, although the above RTS/CTS handshaking mechanism appears to be similar to that of the IEEE 802.11 protocol (which is the reason we use the notion of RTS/CTS), information delivered through RTS/CTS in these two protocols are different. In DFT-MSN, the two control packets exchange nodal delivery probability and available buffer space between a sender and its potential receivers, and they are crucial for the nodes to make efficient decisions on data transmission.

Node \( i \) keeps listening on the channel and collects the CTS packets, which are used to construct the neighbor table. After that, node \( i \) can easily make central arrangement for next data transmissions. Hereafter, the communication enters the synchronous phase. In addition, similar to IEEE 802.11 DCF function, the network allocation vector (NAV) mechanism can be employed to minimize overhearing and address the hidden station problem. Thus, the neighboring nodes of each receiver update their NAVs upon receiving the CTS. The details of updating NAV is omitted here.

In the above asynchronous phase, contention mainly happens in two situations. First, multiple nodes may try to grasp the channel by transmitting a preamble simultaneously. Second, multiple qualified neighbor nodes may reply with CTS packets simultaneously. For both situations, we have designed simple and effective schemes to address the tradeoff between the collision probability and bandwidth utilization, as to be discussed in Sec. 4.2.

### 4.1.2 Synchronous Phase

In this phase, all the data transmissions are synchronized, and thus contention-free. After obtaining information from the qualified receivers, node \( i \) decides which of them are to be
selected for data forwarding, according to the FTD of the outgoing message $M$ and the receivers’ delivery probabilities. In order to reduce unnecessary transmission overhead, only a subset $\Phi$ of the qualified receivers are selected for transmission so that the total delivery probability of message $M$ is just enough to reach a predefined threshold $\mathcal{R}$.

Algorithm 1 in Sec. 3.4.2 is employed to determine $\Phi$.

Node $i$ then constructs and sends out a SCHEDULE packet which includes the list of IDs of the selected receivers and the corresponding FTD of the message copy to be received by each receiver (which is calculated according to Eq.(3.29)). The ID list also implicitly determines the order of ACK packets to be transmitted by the receivers.

Following that, node $i$ sends out the data message and waits for the acknowledgements from the receivers.

If a node $j$ finds its ID in the SCHEDULE packet, it accepts the following message $M$, which is then inserted into its data queue and attached with the FTD indicated in the SCHEDULE packet. Then, node $j$ replies with an ACK packet at a specific time slot according to the receiver list in the SCHEDULE packet. For example, if node $j$ is the $k$-th receiver in the list, it transmits its ACK at $k \times t_{ack}$ after receiving the message, where $t_{ack}$ is a constant period for the ACK transmission.

If any ACK packet is lost, node $i$ assumes that the corresponding neighbor is invalid and removes it from $\Phi$. After receiving the ACK packets, node $i$ recalculates the FTD of its local copy of message $M$ by using Eq. (3.30) and updates the data queue, as discussed in Sec. 3.4.1.

Each sensor repeats the above two-phase process as long as it is in the work mode.

4.2 Protocol Optimizations

Communication link and battery power are the two scarcest resources in DFT-MSN, directly dictating its data delivery performance. Given their very low nodal density, the
mobile sensors are intermittently connected, thus calling for the needs of making the utmost use of the temporarily available communication links. A naive approach is to let sensors stay in the work mode continuously, so as to catch every possible opportunity for data transmission. Under such an approach, however, the sensors may drain off their battery quickly, resulting in poor performance of the overall network.

Clearly, there is a tradeoff between link utilization and energy efficiency. In this section, we address this tradeoff from the following three perspectives.

4.2.1 Periodic Sleeping

The radio transceiver can be in one of the four possible states: transmitting, receiving, listening, and sleeping, which correspond to four power levels. For short range wireless communication as in sensor networks, power consumed is about the same for listening to idle channels as for reception or transmission of useful data. Worse yet, due to the inherent nature of sparse connectivity of DFT-MSN, the sensor nodes are mostly in the idle listening state if they do not turn off their radios. Hence periodic sleeping is clearly necessary for prolonging the lifetime of individual sensors and accordingly the entire DFT-MSN. On the other hand, it is obvious that sleeping will degrade the link utilization, thus lowering the protocol performance in terms of the delivery ratio and delay.

In order to deal with this tradeoff, a simple and effective scheme is proposed to determine when and how long a node should go to sleep. As we have discussed in Sec. 4.1, a node $i$ turns off its radio for a period of $T_i$ if it does not act as either a sender or a receiver in the past $L$ transmissions. The sleeping period $T_i$ is determined by two factors. The first one is how likely the node can perform transmissions if the radio is turned on (i.e., it meets another node with a higher delivery probability). To facilitate our discussion,
we introduce a parameter $\rho_i$ defined as

$$
\rho_i = \begin{cases} 
  s_i/S, & s_i \neq 0 \\
  1/S, & s_i = 0,
\end{cases}
$$

(4.1)

where $s_i$ is the number of working cycles in which node $i$ has done transmission successfully in the past $S$ consecutive cycles. If $\rho_i$ is large, the sleeping period $T_i$ ought to be shortened to enable more transmissions. On the other hand, if $\rho_i$ is small, the node may employ a large $T_i$ so as to reduce unnecessary power consumption.

The second one is related to node’s available message buffer. When the message buffer is likely to be full, a short sleeping period should be used since the sensor needs to transmit whenever there is a chance in order to avoid dropping the important messages. To this end, we define

$$
\alpha_i = K_{Fi}^i/K,
$$

(4.2)

where $K_{Fi}^i$ is the number of messages with FTD smaller than $F$, while $K$ is total buffer space in terms of the number of messages.

Based on $\rho_i$ and $\alpha_i$, the sleeping period of node $i$, $T_i$, can then be calculated as follows:

$$
T_i = \text{Max}(T_{\text{min}}, T_{\text{min}} \times \left\lceil \frac{1}{\rho_i} \times \frac{1}{(1 - H + \alpha_i)} \right\rceil),
$$

(4.3)

where $T_{\text{min}}$ is the minimum sleeping period and $H$ is a predetermined threshold. If $\alpha_i \geq H$, the sleeping period is shortened; otherwise, a larger sleeping period is employed. Assume the power consumption of turning on/off the radio is $P_{\text{change}}$, while power consumption for the radio in the idle state and the sleep state equals $P_{\text{idle}}$ and $P_{\text{sleep}}$, respectively. To ensure a net power saving, $T_{\text{min}}$ should be

$$
T_{\text{min}} \geq \frac{2P_{\text{change}}}{P_{\text{idle}} - P_{\text{sleep}}},
$$

(4.4)

In addition, since the minimum value of $\rho_i$ is $\frac{1}{S}$, the maximum sleeping period, i.e., $T_{\text{max}}$, is

$$
T_{\text{max}} = \frac{S}{1 - H} \times T_{\text{min}}.
$$

(4.5)
4.2.2 Collision Avoidance During RTS Transmission

Collision happens when multiple neighboring nodes try to transmit simultaneously. As a result, the colliding messages are lost, and both energy and bandwidth consumed for the communication are then wasted. In the proposed protocol, collisions mainly happen in the asynchronous phase, where nodes contend for channel access, due to two reasons. First, multiple nodes may try to grasp the channel by transmitting their preambles simultaneously. Second, if a single preamble is successfully transmitted followed by an RTS, multiple qualified neighbor nodes may reply with CTS simultaneously, which again results in collisions. In this subsection, we discuss the approach for minimizing the collision probability of transmitting preambles and RTS, while the problem involved in CTS transmission will be discussed in the next subsection.

As discussed in Sec. 4.1, when a node \( i \) wakes up, it listens to the channel for a period of \( \tau_i \) before initiating any data transmission. A carefully designed collision avoidance scheme is needed here, in order to reduce the overall collision probability, while allowing the node with a lower delivery probability to have a better chance to grasp the channel. This is important for achieving high channel efficiency, because, once winning channel contention, the node with lower delivery probability is more likely to identify receivers, given a node always transmits data to other nodes with higher delivery probabilities.

Intuitively, we can assign a short listening period to the nodes with a lower delivery probability, so that they can grasp the channel easily. This naive approach, however, may leads to the serious unfairness problem. Since each node starts listening almost simultaneously after the previous transmission, the node with the smallest listening period will always occupy the channel first. As a result, before the node with the smallest delivery probability finishes transmitting all of its messages, no other nodes have a chance to transmit.
Here, a simple and effective scheme is proposed for a node to select its listening period adaptively. Let $\tau_{\text{max}}$ denote the maximum listening period. Each time when a node $i$ starts listening to the channel, it dynamically selects a listening period, which is uniformly distributed between 1 and $\sigma_i$. $\sigma_i$ is determined by the following equation,

$$\sigma_i = \xi_i \times \tau_{\text{max}}.$$  \hspace{1cm} (4.6)

Next, the probability for a node to grasp the channel is estimated. For simplicity, we consider an independent cell, where all nodes inside can communicate with each other, but none of them can contact any node outside the cell. Since we are considering a sparse network, the interference from the nodes outside the cell can be ignored. Assume there are $m$ nodes inside the cell and every node knows the delivery probability of others by referring to its neighbor table. The probability that an arbitrary node $i$ can grasp the channel is approximated as

$$P_i = \sum_{\tau_i=1}^{\sigma_i} \frac{1}{\sigma_i} \times \prod_{j=1}^{m} \frac{\theta_{ij}}{\sigma_j},$$

where

$$\theta_{ij} = \begin{cases} \sigma_j - \tau_i, & \sigma_j > \tau_i, \\ 0, & \sigma_j \leq \tau_i. \end{cases}$$  \hspace{1cm} (4.8)

Thus, the probability that no one can grasp the channel (probability of collisions), i.e., $\gamma$, is

$$\gamma = 1 - \sum_{i=1}^{m} P_i.$$  \hspace{1cm} (4.9)

Obviously, the larger $\tau_{\text{max}}$ is, the less likely the collision will happen. However, with a large $\tau_{\text{max}}$, the sensor nodes need to listen for a long time, resulting in low channel efficiency and high energy consumption. Our goal is to find a minimum $\tau_{\text{max}}$ which keeps the collision probability under a predefined threshold $H$, i.e.,

$$\min (\tau_{\text{max}} | \gamma \leq H).$$  \hspace{1cm} (4.10)

This can be done by individual sensors using typical optimization schemes.
4.2.3 Collision Avoidance During CTS Transmission

Each RTS packet contains a field of contention window to indicate the period during which the sender waits for CTS packets. All qualified neighbors should reply with CTS packets within this period. If multiple nodes answer simultaneously, collision happens and all colliding CTS packets are lost.

In order to reduce the collision probability, the contention window is split into a number of small time slots. Each slot equals the time to transmit a CTS packet by the receiver, plus the time for the sender to process the CTS packet. While it is desirable to have each qualified neighbor reply CTS in a unique time slot, this, however, is not realistic since the nodes are not synchronized at this moment. An intuitive approach is to let a qualified receiver randomly select a time slot. In this simple approach, the length of the contention window, i.e., $W$, is a crucial parameter. More qualified neighboring nodes need a larger contention window.

In the following discussion, we propose two simple schemes to determine the optimal value of contention window $W$. The former tries to minimize the overall collision probability, while the latter aims to minimize the collision probability for those nodes with high nodal delivery probabilities (thus better qualifying to serve as receivers).

1) Minimizing Overall Collision Probability

In this scheme, we intend to minimize the probability of any CTS packet experiencing collisions. Assume node $i$ has sent RTS successfully to a set of its neighbors, denoted by $\phi_i$. Let $|\phi_i|$ be the number of nodes in $\phi_i$.

Let every qualified neighbor node randomly select a time slot between 1 and $W$ for sending its CTS packet. Then, the probability that every CTS is transmitted in a collision-free unique time slot is $\left(\frac{W}{|\phi_i|}\right) \times |\phi_i|! \times \left(\frac{1}{W}\right)^{|\phi_i|}$, and accordingly the overall
collision probability equals
\[
\gamma_o = 1 - \left( \frac{W}{|\Phi_i|} \right) \times |\Phi_i|! \times \left( \frac{1}{W} \right)^{|\Phi_i|}.
\] (4.11)

A minimum \( W \) can thus be chosen to ensure \( \gamma_o \) lower than an expected value similar to what we have discussed in the previous subsection.

2) Minimizing Collision Probability for Nodes with High Delivery Probabilities

Since the nodes with high delivery probabilities normally are better candidates for data relaying, it is reasonable to make their collision probabilities as small as possible. With this consideration in mind, we design a special collision-avoiding scheme, as elaborated below.

When a node \( i \) transmits RTS, the following information is embedded in the packet: contention window (\( W \)), the maximum delivery probability of neighboring nodes (\( \xi_{\max(i)} \)), and the current delivery probability of node \( i \) (\( \xi_i \)). Note that \( \xi_{\max(i)} \) is an estimation based on historical information that node \( i \) has, and it may or may not be accurate. If a qualified neighboring node \( j \) receives this RTS, it calculates the time slot used to transmit its CTS, which is a random variable between 0 and \( t_j \). Obviously, the larger \( t_j \) is, the less likelihood that node \( j \) will conflict with other nodes. \( t_j \) is calculated according to the following equation:

\[
t_j = \left( \frac{\xi_j - \xi_i}{\xi_{\max(i)} - \xi_j} \right) \times W.
\] (4.12)

Thus, if node \( j \) has an equal or larger delivery probability than \( \xi_{\max(i)} \), it can select an arbitrary time slot within the whole contention window. Otherwise, if \( \xi_j \) is small, it may have few choices.

In this scheme, \( W \) is also crucial to the collision probability. However, it is not very meaningful to discuss the overall collision probability. Instead, we are more interested in the collision probability for those nodes with high delivery probabilities. For a specific
neighbor $j$ with high delivery probability $\xi_j$, its collision probability is

$$\gamma_j = \sum_{s=1}^{t_j} \left( \frac{1}{t_j} \times \left( 1 - \prod_{k \in \Phi_i, k \neq j} \delta_k(s) \right) \right),$$

(4.13)

where $\prod_{k \in \Phi_i, k \neq j} \delta_k(s)$ is the probability that no other node selects slot $s$ except node $j$, while $\delta_k(s)$ denotes the probability that node $k$ doesn’t choose slot $s$,

$$\delta_k(s) = \begin{cases} \frac{t_k - 1}{t_k}, & t_k \geq s, \\ 1, & \text{otherwise}. \end{cases}$$

(4.14)

Obviously, for neighbors with delivery probabilities greater than $\xi_j$, their collision probabilities are smaller than $\gamma_j$.

Finally, the optimal value of $W$ can be calculated by node $i$ before it sends out its RTS packet, in order to keep $\gamma_j$ lower than a threshold.

### 4.3 Simulation Results

Extensive simulations have been carried out to evaluate the performance of the proposed cross-layer data delivery protocol for DFT-MSN. In the default simulation setup, 3 sink nodes and 100 sensor nodes are randomly scattered in an area of $200 \times 200$ m$^2$. The whole area is divided into 25 non-overlapped zones, each with an area of $40 \times 40$ m$^2$. A sensor node is initially resided in its home zone. It moves with a speed randomly chosen between 0 and 5 m/s. Whenever a node reaches the boundary of its zone, it moves out with a probability of 20%, and bounces back with a probability of 80%. After entering a new zone, the sensor repeats the above process. However, if it reaches the boundary to its home zone, it returns to its home zone with a probability of 100%. Each sensor has a maximum transmission range of 10 m and a maximum queue size of 200 messages. The data generation of each sensor follows a Poisson process with an average arrival interval of 120 s. Each data message has 1000 bits, while each control packet has 50 bits. The channel bandwidth is 10 kbps. The power consumption is estimated according to the transceiver
used in Berkeley mote, which consumes 13.5mW, 24.75mW and 15µW, in receiving, transmitting and sleep, respectively [90]. The power consumption of idle listening is the same as that of receiving, while the power consumption of turning on/off the radio is four times of listening power consumption. Each simulation lasts 25000s. For a given simulation setup, we run the simulation multiple times and average the collected results.

In order to evaluate the performance of the proposed data delivery scheme and its optimizations, we have implemented four protocols with different optimization levels, dubbed \textit{OPT}, \textit{NOOPT}, \textit{NOSLEEP}, and \textit{ZBR}, respectively. \textit{OPT} is the proposed protocol that employs all optimization schemes (discussed in Sec. 4.2 for the protocol parameters $\tau_{\text{max}}$, $W$, and $T_i$). \textit{NOOPT} adopts the basic protocol proposed in Sec. 4.1, but without optimization. Instead, fixed protocol parameters are employed. \textit{NOSLEEP} is similar to \textit{OPT}, except that the nodes do not perform periodic sleeping. Finally, \textit{ZBR} differs from \textit{OPT} only in the message transmission scheme, by replacing the message fault tolerance-based multicast scheme with the ZebraNet’s history-based scheme [104], which is the most comparable scheme in the literature. Note that the data delivery scheme proposed in SWIM [105] is not simulated here, because it is designed for the network with uniform nodal mobility, and thus work ineffectively in our simulation scenarios where different sensor nodes have different delivery probabilities. Other protocols developed for mobile ad-hoc networks (such as IEEE 802.11 and its enhanced versions) or conventional sensor networks (e.g., [123, 124, 125]) are not considered either, because their performance level are expected to be far lower than the proposed approach in DFT-MSN, although each of them may have been optimized for their target application scenarios.

We vary several parameters to observe their impacts on performance. Fig. 4.2 presents the protocol performance by varying the number of sink nodes in DFT-MSN. As shown in Fig. 4.2(a), with more sink nodes present, the sensors exhibit a better opportunity to reach the sink nodes, thus resulting in a higher delivery ratio. \textit{OPT} and \textit{NOSLEEP} outperform
NOOPT slightly since they both employ optimized parameters to reduce the collision probability. As expected, ZBR has the lowest delivery ratio, especially when there are only a few sink nodes, because it employs the direct contact probability to decide message transmission. For the nodes that never directly meet the sink nodes, the transmission becomes random, and thus less efficient.

Fig. 4.2(b) compares the average nodal energy consumption rate of different implementations. Obviously, with more sink nodes existing, the message can be transmitted with fewer hops, reducing energy consumption. As expected, OPT conserves energy effectively compared to the other approaches. Since NOOPT does not optimize the protocol parameters, we observe many collisions during RTS/CTS transmissions, which accordingly waste battery power. Energy consumed by NOSLEEP is very high (about eight times of the energy consumption of OPT), due to its idle listening. On the other hand, because of its inefficient transmission control, ZBR has higher overhead than OPT, although the same optimized parameters are employed.

The average message delay is compared in Fig. 4.2(c). The delay decreases sharply with an increase in the number of sink nodes, since the messages can then be delivered to the sinks with fewer hops. NOSLEEP has a shorter message delay than OPT and NOOPT, because the nodes do not sleep at all so that they can capture every possible transmission chance, resulting in faster delivery. Surprisingly, ZBR also has a very low delay shown in the figure. This delay, however, is for successfully delivered data messages only. We have observed that in ZBR, most of the messages delivered successfully are those generated by nodes near the sinks, thus resulting in less transmission hops and accordingly a shorter message delivery delay. Clearly, it is not meaningful to compare it with other approaches in terms of the delay metric.

Fig. 4.3 depicts the impact of the number of sensor nodes. With an increase in the network size, the delivery ratio decreases sharply for all protocols, as can be seen in Fig.
4.3(a). When more nodes exist in the field, the nodes near the sinks will carry much more messages and become the bottlenecks. Due to the limited bandwidth and buffer size, many messages are to be dropped, resulting in lower delivery ratio. Meanwhile, all nodes experience more contention and collisions, further degrading performance. Due to the lack of proper transmission control, ZBR exhibits significantly increased overhead in the network with more sensor nodes; that is in sharp contrast to other protocols, as shown in Fig. 4.3(b). Meanwhile, the message delivery delay decreases with an increase in the network size for all protocols, since each node then has a better chance to reach other nodes with higher delivery probabilities, resulting in faster delivery (see Fig. 4.3(c)).

We also vary the maximum queue size of each sensor, with results presented in Fig. 4.4. As the maximum queue size grows, the delivery ratio improves under all approaches, as expected (see Fig. 4.4(a)). However, the queue size has a limited impact on mean nodal power consumption, as shown in Fig. 4.4(b). This is reasonable because the queue size does not significantly affect the number of data transmissions. On the other hand, the message delivery delay increases slightly with a larger queue size, because more data messages can then reside in the queue for a longer time before being delivered.

Fig. 4.5 depicts the impact of the nodal moving speed. As the speed increases, the delivery ratios of all approaches rise, while the delivery delays of all approaches decrease. This is because the node with a higher speed has a better opportunity to meet other nodes and to reach the sink nodes. Thus, the messages enjoy a better chance to be delivered before they are dropped. It is also noticed that the transmission overhead of OPT decreases with the increase of the nodal speed (as shown in Fig. 4.5(b)), making it adaptable for networks with various nodal speeds.

In short, our simulation results demonstrate the effectiveness of the proposed cross-layer data delivery protocol with the optimization schemes, which achieves the
highest data delivery ratio and the lowest energy consumption, with only marginally increased delay overhead, when compared with other approaches.
FIGURE 4.2. Impact of number of sink nodes.
FIGURE 4.3. Impact of number of sensor nodes.
FIGURE 4.4. Impact of maximum queue size.
FIGURE 4.5. Impact of average nodal speed.
Chapter 5

Analytic Study for DFT-MSN

Analytic study is a proven approach for studying system performance, revealing underlying characteristics, and evaluating communication protocols. There are two major steps towards the analytic study on DFT-MSN [126]. First, the specific application scenario needs to be modeled properly. Some simplifications and assumptions are normally employed in this step to make the model tractable. Then in the second step, mathematical approaches are applied, in order to derive the desired results or demonstrate the underlying insights. Particularly, we are most interested in several network performance metrics, such as average delivery ratio, average message delivery delay, protocol overhead, average queue length, etc. In the rest of this chapter, we introduce a generic queuing analytic model [127] for DFT-MSN, which aims at providing an insight of the queuing characteristics of DFT-MSN.

5.1 Generic Analytic Model for DFT-MSN

We first discuss a few assumptions used in this work and then analyze the message arrival and service times, based on which we present the general analytic results. The symbols and notations used in the analysis are summarized in Table 5.1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Size of the network area</td>
</tr>
<tr>
<td>$A_D$</td>
<td>Circular area with a radius of $r$ centered at the point $D$</td>
</tr>
<tr>
<td>$A_s$</td>
<td>Circular area with a radius of $r$ centered at the current location of sink node $s$</td>
</tr>
<tr>
<td>$A_\Phi$</td>
<td>Area covered by all sink nodes in the network</td>
</tr>
<tr>
<td>$f_i$</td>
<td>Average message generating rate at sensor $i$</td>
</tr>
<tr>
<td>$f_{i'}$</td>
<td>Average message arrival rate from other nodes to node $i$</td>
</tr>
<tr>
<td>$f_{i''}$</td>
<td>Average message arrival rate by which node $i$ reinserts the messages back to its queue after transmission</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$l_1$</td>
<td>The inner radius of the ring-shaped area in power-law distributed mobility pattern</td>
</tr>
<tr>
<td>$l_2$</td>
<td>The outer radius of the ring-shaped area in power-law distributed mobility pattern</td>
</tr>
<tr>
<td>$L$</td>
<td>Average system length</td>
</tr>
<tr>
<td>$L_i$</td>
<td>Average queue length of sensor $i$</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of sensor nodes in the network</td>
</tr>
<tr>
<td>$N_{\Theta_i}$</td>
<td>Number of nodes in $\Theta_i$</td>
</tr>
<tr>
<td>$N_{\Psi_i}$</td>
<td>Number of nodes in $\Psi_i$</td>
</tr>
<tr>
<td>$N_{\Omega_i}$</td>
<td>Number of sets in $\Omega_i$</td>
</tr>
<tr>
<td>$N_{\Omega_i^k}$</td>
<td>Number of sets in $\Omega_i^k$</td>
</tr>
<tr>
<td>$p_i(D)$</td>
<td>Probability that sensor $i$ visits a specific location $D$</td>
</tr>
<tr>
<td>$P_i(D)$</td>
<td>Probability that sensor $i$ is within the area $A_D$</td>
</tr>
<tr>
<td>$P_i(s)$</td>
<td>Probability that sensor $i$ is within the area $A_s$</td>
</tr>
<tr>
<td>$Q_{ji}$</td>
<td>Probability that sensor $j$ can transmit to sensor $i$</td>
</tr>
<tr>
<td>$S$</td>
<td>Number of sink nodes in the network</td>
</tr>
<tr>
<td>$S_i$</td>
<td>Overall service rate of node $i$</td>
</tr>
<tr>
<td>$S_i(\Phi)$</td>
<td>Service rate of node $i$ due to transmission to sink nodes directly</td>
</tr>
<tr>
<td>$S_i(\Sigma)$</td>
<td>Service rate of node $i$ due to transmission to other nodes</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>Average message delivery delay</td>
</tr>
<tr>
<td>$\Delta_i$</td>
<td>Average waiting time of a message at node $i$</td>
</tr>
<tr>
<td>$\Theta_i$</td>
<td>The set of nodes with lower delivery probability than node $i$</td>
</tr>
<tr>
<td>$\Theta_{ij}$</td>
<td>The set of nodes in $\Theta_i$, but with node $j$ excluded</td>
</tr>
<tr>
<td>$\Upsilon$</td>
<td>Average message delivery ratio of the network</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>The set of sink nodes</td>
</tr>
<tr>
<td>$\Psi_i$</td>
<td>The set of nodes with higher delivery probability than sensor $i$</td>
</tr>
<tr>
<td>$\Omega^k_i(h)$</td>
<td>A set of $k$ nodes out of $N - 1$ nodes (excluding node $i$), where at least one of them have higher delivery probability than node $i$.</td>
</tr>
<tr>
<td>$\Omega^k_{ij}(h)$</td>
<td>A set of $k$ nodes out of the $N_{\Theta_{ij}}$ nodes in $\Theta_{ij}$ (in ZebraNet’s History-based Scheme), or a set of $k$ nodes out of the $N - 2$ nodes with node $i$ and node $j$ excluded (in the Simple Replication-based Scheme).</td>
</tr>
<tr>
<td>$\alpha^k_i(s)$</td>
<td>Probability that there are exact $k$ nodes, except node $i$, within $A_s$.</td>
</tr>
<tr>
<td>$\alpha^k_i(D)$</td>
<td>Probability that there are exact $k$ other nodes, except node $i$, in $A_D$ and at least one of them has higher delivery probability than sensor $i$.</td>
</tr>
<tr>
<td>$\alpha^k_{ij}(D)$</td>
<td>Probability that exact $k$ other sensors in $\Theta_{ij}$ are in $A_D$ (in ZebraNet’s History-based Scheme), or the probability that exact $k$ other sensors (excluding node $i$ and node $j$) are in $A_D$ (in the Simple Replication-based Scheme).</td>
</tr>
<tr>
<td>$\beta$</td>
<td>The exponent parameter of the pow-law distribution</td>
</tr>
<tr>
<td>$\epsilon_{ji}$</td>
<td>Overall probability that a message is transmitted from $j$ to $i$</td>
</tr>
<tr>
<td>$\epsilon_{j\Phi}$</td>
<td>Overall probability that a message is transmitted from $j$ to any sink node directly</td>
</tr>
<tr>
<td>$\rho_i$</td>
<td>Traffic load at sensor $i$, i.e., $\lambda_i/\mu_i$</td>
</tr>
<tr>
<td>$\zeta_i$</td>
<td>Probability that sensor $i$ can deliver a message to the sink nodes either directly or via other sensors</td>
</tr>
<tr>
<td>$\lambda_i$</td>
<td>Total average message arrival rate of node $i$</td>
</tr>
<tr>
<td>$\mu_i$</td>
<td>Service rate of node $i$ after approximating the service time with exponential distribution</td>
</tr>
<tr>
<td>$q^i_r$</td>
<td>Probability that there are exactly $q \leq K$ data messages in the queue of sensor $i$</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$\bar{R}$</td>
<td>Average message dropping ratio of the network</td>
</tr>
</tbody>
</table>

### 5.1.1 General Assumptions

Initially $N$ sensors and $S$ sink nodes are uniformly distributed in the area of interest. Each sensor collects information periodically, and tries to deliver the messages to the sink nodes using a short-range radio with a transmission range of $r$. For analytic tractability, we make the following assumptions.

- **Mobility Pattern:** After initial deployment, each sensor moves around with a known mobility pattern. The probability that a sensor $i$ visits a specific location $D$ is denoted by $p_i(D)$. In addition, to facilitate our discussion, we denote $P_i(D)$ as the probability that node $i$ is within the area $A_D$, which is the circular area centered at $D$ and with a radius of $r$. Clearly,

  \[ P_i(D) = \int \int_{A_D} p_i(D) dD, \quad (5.1) \]

- **Time Slot:** Without loss of generality, we assume the time is split into small slots, each of which equals the transmission time of one message between two neighboring nodes. Thus at most one message can be delivered in a time slot, even multiple nodes are within the transmission range of each other.

- **Data Generation:** Each sensor node generates data messages periodically. The data generation follows a Poisson process with mean value of $f_i$ for sensor $i$.

- **Sink Node Locations:** The sink nodes are assumed to be placed far away from each other so that a sensor node is not located within the transmission range of more than one sink nodes.
• **Queuing Management Scheme:** We assume each node maintains a First-Come-First-Served (FCFS) queue with a size of $K$ messages. When a new message arrives while the queue is full, the new message is dropped.

• **Data Transmission:** The data transmission between two nodes may occur when they are in the transmission range of each other. We assume one and only one transmission can take place during one time slot within a given neighborhood area (e.g., in $A_D$). If there are multiple nodes, each of them has the same probability to transmit and possible collision is ignored.

In the following subsections, we first analyze the behaviors of the message service and arrival processes at each individual sensor. Then, based on Jackson network theory, each individual sensor is approximated as an M/M/1/K queue. Consequently, the DFT-MSN can be modeled as a network of queues. Finally, we derive the queuing characteristics of the network following Jackson’s theorem.

### 5.1.2 Service Time

Assume that the data processing time is negligible. Then the service time of a sensor equals the time to transmit a data message. A message transmission may occur in two situations, depending on the data delivery scheme employed.

First, the sensor can send a data message directly to a sink if it is within the service area of a sink node $s$ (i.e., the circular area centered at the current location of $s$ and with a radius of $r$), denoted by $A_s$. The probability that sensor $i$ moves into $A_s$ is denoted by $P_i(s)$, and at the same time, the probability that there are exact $k$ other nodes within $A_s$ is

$$
\alpha^k_i(s) = \sum_{h=1}^{N-1} \left[ \prod_{m \in \Omega^k_i(h)} P_m(s) \right] \prod_{n \neq i, n \notin \Omega^k_i(h)} (1 - P_n(s)),
$$

where $\Omega^k_i(h)$ denotes a set of $k$ nodes out of the total $N - 1$ nodes (with node $i$ excluded) and $h$ is an index, ranging from 1 to $\binom{N-1}{k}$. Since only one of these $k + 1$ nodes can
transmit in each time slot, the probability of node $i$ transmits to sink $s$, given that there are $k$ other nodes around $s$, is $\frac{1}{k+1}$. Thus, the average probability that node $i$ can transmit to any sink node is

$$S_i(\Phi) = \sum_{s \in \Phi} \left[ P_i(s) \times \sum_{k=0}^{N-1} \left( \frac{1}{k+1} \times \alpha^k_i(s) \right) \right],$$

(5.3)

where $\Phi$ denotes the set of sink nodes. To simplify the computation, one can also use an approximation as below. When a node $i$ moves into $A_s$, the average number of nodes located within $A_s$, i.e., $N_i(s)$ is

$$N_i(s) = 1 + \sum_{j=1, j \neq i}^{N} P_j(s).$$

(5.4)

Since only one transmission happens during each time slot, and the sender is randomly selected from the $N_i(s)$ sensors, the probability that sensor $i$ can transmit to any one of the sink nodes is then estimated as

$$S_i(\Phi) = \frac{\sum_{s \in \Phi} P_i(s)}{N_i(s)}.$$

(5.5)

Second, depending on the data delivery scheme employed, a sensor may also send data messages to nearby sensor nodes, which can then relay the messages towards the sink nodes in a multi-hop manner. Let $S_i(\Xi)$ denote the probability that a sensor $i$ transmits a message to other sensor nodes, which obviously depends on the individual data delivery scheme. Thus, the average probability that sensor $i$ can transmit a data message (denoted by $S_i$) is the probability that it can transmit directly to the sink nodes (i.e., $S_i(\Phi)$), plus the probability that it transmits to other sensor nodes (i.e., $S_i(\Xi)$),

$$S_i = S_i(\Phi) + S_i(\Xi).$$

(5.6)

Let $X$ denote a random variable representing the number of slots required for sensor $i$ to successfully send out a data message. $X$ thus is geometrically distributed with parameter $S_i$. Its mass distribution function is

$$f_X(x = n) = (1 - S_i)^{n-1} S_i.$$

(5.7)
Note that the geometric distributed random variable can be approximated by an exponential random variable, i.e.,

\[ F_X(x \leq n) = 1 - e^{-\mu n}. \]  

(5.8)

Therefore, we arrive at an exponential service time with a mean service rate of \( \mu = -\ln(1 - S_i) \).

### 5.1.3 Message Arrival

The messages processed at a sensor node \( i \) come from three sources, depending on the data delivery scheme employed. First, node \( i \) itself generates data messages periodically. As mentioned earlier, this self-message generating process follows Poisson distribution with mean value of \( f_i \). Secondly, other nodes may send data messages to node \( i \), with the average data rate denoted by \( f'_i \). Last, after sending a message, node \( i \) may reinsert this message back to its queue. This leads to an additional message arrival rate of \( f''_i \). Since the service time is exponential, we may apply the Kleinrock’s approximation, and thus the combined message arrival rate at sensor \( i \) can be approximated as Poisson distribution with an average arrival rate of

\[ \lambda_i = f_i + f'_i + f''_i, \]  

(5.9)

where \( f_i \) is given while \( f'_i \) and \( f''_i \) are to be determined under specific data delivery schemes.

### 5.1.4 Queuing Characteristics

If the maximum queue size \( K \) is large compared with average queue length, each individual queue in the network can be approximated as an M/M/1/K queue according to Jackson network theory. We can readily derive the queuing characteristics of each sensor node, such as the average queue length, average dropping rate, etc., with standard approach. According to our assumption of FCFS queue, the newly arriving message is
dropped when the queue is full. In the equilibrium state, the probability that there are exactly \( q \leq K \) data messages in the queue of sensor \( i \) is

\[
\mathcal{q}_i^q = \rho_i^q \times \frac{1 - \rho_i}{1 - \rho_i^K + 1},
\]

(5.10)

where \( \rho_i \) is the traffic intensity of node \( i \), with value of \( \frac{\lambda_i}{\mu_i} \). Thus the average number of data messages in the queue of sensor \( i \) is

\[
L_i = \frac{\rho_i}{1 - \rho_i} - \frac{\rho_i \times (K \times \rho_i^K + 1)}{1 - \rho_i^K + 1}.
\]

(5.11)

Note that, if \( \lambda_i > \mu_i \), which is possible in some scenarios, the data queue of node \( i \) is always full, i.e., \( L_i = K \).

To estimate the average time a message spending at node \( i \), we need to calculate the effective data arrival rate that actually enter queue \( i \). Note that when queue \( i \) is already full, it no longer accepts any data messages. Hence, the effective message arrival rate (of the messages that actually enter queue \( i \)) is \( \lambda_i(1 - \mathcal{q}_i^K) \). Therefore the average waiting time of a message at node \( i \) is

\[
\Delta_i = \frac{L_i}{\lambda_i(1 - \mathcal{q}_i^K)}.
\]

(5.12)

By treating our system as a network of queues, we arrive at the average number of data messages in the system, i.e.,

\[
L = \sum_{i=1}^{N} L_i.
\]

(5.13)

If there are multiple copies existing in the network for each message, it is difficult to directly derive the average message delivery delay. However, for those simple delivery schemes which only keep at most one copy for each message at any time, the average message delay can be estimated by applying Little’s formula on the whole network. Since the effective message arrival rate of the entire network can be estimated as \( \sum_{i=1}^{N} f_i(1 - \mathcal{q}_i^K) \), the average delay that a data message stays in the system is

\[
\Delta = \frac{L}{\sum_{i=1}^{N} f_i(1 - \mathcal{q}_i^K)}.
\]

(5.14)
Meanwhile, if there is only one copy exists for each message in the network, the system
delivery ratio can be estimated by the complement of the system dropping ratio, i.e., $\mathcal{R}$,
which is calculated as the total message dropping rate versus total message arrival rate of
the network, as shown below,
$$
\mathcal{R} = \frac{\sum_{i=1}^{N} \omega_i^K \times \lambda_i}{\sum_{i=1}^{N} f_i}.
$$
(5.15)
Thus the average data delivery ratio of the system is
$$
\Upsilon = 1 - \mathcal{R},
$$
(5.16)
which signifies the overall efficiency of DFT-MSN and is usually the most interest of our
performance evaluation.

5.2 Applying The Generic Model in Specific Scenarios

In this section, we elaborate how to apply the generic queuing analytic model to evaluate
the performance of a network with specific data delivery scheme and nodal mobility
pattern. We first examine the direct delivery scheme with two different nodal mobility
patterns, i.e., the uniform distribution pattern and the power-law distribution pattern. Then
we examine the ZebraNet’s history-based data delivery scheme and a simple
replication-based scheme in networks with nodal mobility pattern following power-law
distribution, respectively.

5.2.1 Direct Delivery Scheme

Under the direct delivery scheme, the sensor nodes do not communicate to each other. A
sensor only delivers its messages to the sink nodes directly.

1) Uniform Distribution Pattern

We first consider the scenario that the sensors follow a uniform mobility pattern. More
specifically, the sensors move randomly inside the interested area $A$ following uniform
distribution. Thus, the density function of the random variable that a sensor $i$ is in an arbitrary location $D$ is

$$p_i(D) = \frac{1}{A},$$

(5.17)

where $A$ is the size of the whole area. Then $P_i(s) = \frac{A_s}{A}$. Plugging $p_i(D)$ and $P_i(s)$ into Equations 5.2-5.3 we have

$$S_i(\Phi) = \frac{S \times A_s}{A + (N - 1)A_s}.$$  

(5.18)

For the simple direct delivery scheme, it is obvious that $S_i(\Xi) = 0$, $f_i' = 0$, and $f_i'' = 0$. Thus we have

$$\begin{aligned}
\mu_i &= -\ln(1 - S_i(\Phi)), \\
\lambda_i &= f_i.
\end{aligned}$$

(5.19)

The average system size $L$ can thus be easily derived using the results in Sec. 5.1.4. Since there is only one copy for each message in the network, the average delay $\Delta$ and the system delivery ratio $\Upsilon$ can be estimated by using Equation 5.14 and Equation 5.16, respectively.

2) Power Law Distribution

In this subsection we consider a DFT-MSN with power law distributed mobility pattern. More specifically, every node has a initial location (or home location). The probability that a sensor $i$ visits any location with a distance $l$ from its initial location is

$$F_i(l) = k_i \left(\frac{1}{l}\right)^\beta,$$

(5.20)

where $k_i$ and $\beta$ are the constant and exponent of the power-law distribution, respectively. Accordingly, the probability that sensor $i$ visits a specific point $D$, which has a distance of $l$ to node $i$’s initial location, is

$$p_i(D) = \frac{F_i(l)}{2\pi l}.$$  

(5.21)
We assume each sensor only moves around its initial location within a ring-shaped area, with inner and outer radius of \( l_1 \) are \( l_2 \). Considering that the probability of sensor \( i \) to be located in this ring-shaped area is 1, i.e.,

\[
\int_{l_1}^{l_2} F_i(l)dl = 1, \quad (5.22)
\]

we have

\[
k_i = \frac{1}{\int_{l_1}^{l_2} \left( \frac{1}{l} \right)^\beta \beta l \ d\beta}, \quad (5.23)
\]

where

\[
\int_{l_1}^{l_2} \left( \frac{1}{l} \right)^\beta \ d\beta = \begin{cases} 
\ln \frac{l_2}{l_1}, & \beta = 1, \\
\frac{1}{1-\beta} \left( l_2^{1-\beta} - l_1^{1-\beta} \right), & \beta \neq 1.
\end{cases} \quad (5.24)
\]

Plugging above results into Equation 5.1-5.5, we can calculate \( S_i(\Phi) \). As discussed in previous subsection, here \( S_i(\Xi) = 0 \), \( f_i' = 0 \), and \( f_i'' = 0 \). We can then calculate the average queue size, average delivery delay, and average delivery ratio, of the network directly by using the results derived in Sec. 5.1.4.

### 5.2.2 ZebraNet’s History-based Scheme

In this subsection, we study the history-based data delivery scheme proposed in ZebraNet [104] by using our generic analytic model. The nodal mobility pattern is assumed to follow power-law distribution. In this history-based scheme, each node maintains a delivery probability, which is the probability that node \( i \) can transmit to the sink node directly, i.e., \( S_i(\Phi) \). When a node meets some other sensor nodes with higher delivery probabilities, it sends a message to the neighboring node with the highest delivery probability, and then delete this message from its own message queue.

Next we first derive the service time and the message arrival rate of a specific node \( i \). Let \( \Omega^k_i(h) \) denote the sets of \( k \) nodes out of \( N - 1 \) nodes (excluding node \( i \)), where at least
one node has higher delivery probability than node $i$. $h$ is the index varying from 1 to $N_{\Omega^h_i}$,

$$N_{\Omega^h_i} = \sum_{j=0}^{k-1} \binom{N-1-N_{\Psi_i}}{j} \times \binom{N_{\Psi_i}}{k-j}, \quad (5.25)$$

where $\Psi_i$ is the set of nodes with higher delivery probability than node $i$, and $N_{\Psi_i}$ is the number of nodes in $\Psi_i$.

When node $i$ moves to a location $D$, the probability that there are exact $k$ other nodes in the area $A_D$, and at least one of them has higher delivery probability than $i$ is

$$\alpha^k_i(D) = \sum_{h=1}^{N_{\Psi_i}} \prod_{m \in \Omega^h_i(h)} P_m(D) \prod_{n \neq i, n \notin \Omega^h_i(h)} (1 - P_n(D)). \quad (5.26)$$

Since sensor $i$ won’t transmit to other sensor nodes if there is any sink node around, the probability that sensor $i$ can transmit to other sensors with higher delivery probabilities is

$$S_i(\Xi) = \int \int_{A - A_{\Phi}} p_i(D) \times \sum_{k=0}^{N_{\Psi_i} - 1} \alpha^k_i(D) \frac{1}{k+1} dD, \quad (5.27)$$

where $A$ is the entire network area and $A_{\Phi}$ is the area covered by the transmission range of all the sink nodes.

At the same time, sensor $i$ may receive data message from a node $j$, if it has the highest delivery probability among the nodes within node $j$’s neighborhood, resulting additional message arrival rate. Let $\Theta_i$ denote the set of nodes with lower delivery probability than node $i$, while $\Theta_{ij}$ denote the set of nodes in $\Theta_i$, but with node $j$ excluded. $N_{\Theta_{ij}}$ is the number of nodes in $\Theta_{ij}$. Meanwhile, let $\Omega^k_{ij}(h)$ denote the sets of $k$ nodes out of the $N_{\Theta_{ij}}$ nodes in $\Theta_{ij}$, where $h$ is an index varying from 1 to $N_{\Omega^k_{ij}} = \binom{N_{\Theta_{ij}}}{k}$.

When node $j \in \Theta_i$ moves to a location $D$, the probability that node $i$ is in the area $A_D$ is $P_i(D)$. Meanwhile, the probability that exact $k$ other sensors in $\Theta_{ij}$ are in the area $A_D$ is

$$\alpha^k_{ij}(D) = \sum_{h=1}^{N_{\Omega^k_{ij}}} \prod_{m \in \Omega^k_{ij}(h)} P_m(D) \times \prod_{n \notin \Omega^k_{ij}(h), n \neq i, n \neq j} (1 - P_n(D)). \quad (5.28)$$
Since node $j$ won’t transmit to node $i$ when it is in the transmission range of any sink node, the probability that sensor $j$ can transmit to sensor $i$ is

$$Q_{ji} = \int_{A-A_{\Phi}} \int_{A-A_{\Phi}} p_j(D) \times P_i(D) \times \sum_{k=0}^{Na(i,j)} \frac{a_k^j(D)}{k+2} dD.$$  \hspace{1cm} (5.29)

Meanwhile, the probability that node $j$ has a nonempty queue is $1 - \wp_j^0$. Therefore, the total data rate that other nodes transmit to node $i$ is

$$f'_i = \sum_{j \in \Theta_i} (1 - \wp_j^0) \times Q_{ji}. \hspace{1cm} (5.30)$$

Note that since $\wp_j^0$ depends on the message arrival rate of node $j$, $f'_i$ also depends on $f'_j$. It is obvious that when $j \in \Theta_i$, $\Theta_j \subset \Theta_i$. Thus, $f'_i$ can be safely calculated in a recursive way. Moreover, if the network is heavily loaded, $\wp_j^0 \approx 0$. Thus $f'_i$ can be approximated as

$$f'_i \approx \sum_{j \in \Theta_i} Q_{ji}. \hspace{1cm} (5.31)$$

We can then calculate the nodal service rate and the nodal message arrival rate following Equations 5.6 & 5.9, respectively, i.e.,

$$\begin{align*}
\mu_i &= - \ln \left( 1 - S_i(\Phi) - S_i(\Xi) \right), \\
\lambda_i &= f_i + f'_i.
\end{align*} \hspace{1cm} (5.32)$$

Since only one copy exists in the network for each message, we can calculate the average queue size, the average delivery delay, and the average delivery ratio of the network directly by using the results derived in Sec. 5.1.4.

### 5.2.3 Simple Replication-based Data Delivery Scheme

In this subsection, we study a simple replication-based data delivery scheme using our generic analytic model. This scheme further extends ZebraNet’s history-based approach by allowing multiple copies to be generated in the network for each message. More specifically, each sensor maintains a delivery probability, which indicates the likelihood it
can deliver the messages to the sink nodes directly, i.e., \( S_i(\Phi) \). A sensor transmits its message to all of its neighbors with higher delivery probability using multicast, and then delete this message from its queue. When the queue is full, any newly arrived message is dropped according to the assumption of FCFS queue. Besides, the nodes also move around following power law distribution introduced in Sec. 5.2.1.

In this simple replication-based scheme, although sensor \( i \) may possibly transmit to multiple neighbors in one time slot, the service rate due to the transmission to other sensor nodes (excluding sink nodes), i.e., \( S(\Xi) \), is still the same as that of the ZebraNet’s history-based approach discussed in Sec. 5.2.2, since only one message is transmitted to the neighbors in one time slot through multicast. However, the number of copies sent out for each message is different. In ZebraNet’s history-based scheme, only one copy exists in the network for each message, while in the simple replication-based scheme, multiple copies can exist in the network for each message.

At the same time, sensor \( i \) may receive messages from other sensors, resulting in additional message arrival rate. Let \( \Theta_i \) denote the set of nodes with lower delivery probability than node \( i \). When node \( j \in \Theta_i \) moves to a location \( D \), the probability that node \( i \) is in the area \( A_D \) is \( P_i(D) \). Meanwhile, let \( \Omega_{ki}^j(h) \) denote a set of \( k \) nodes out of the \( N - 2 \) nodes (with node \( i \) and node \( j \) excluded), where \( h \) is an index varying from 1 to \( N_{\Omega_{ki}^j} = \binom{N - 2}{k} \).

The probability that exact \( k \) other sensors (except node \( i \) and node \( j \)) are in the area \( A_D \) is

\[
\alpha_{ij}^k(D) = \sum_{h=1}^{N_{\Omega_k^i}^j} \left[ \prod_{m \in \Omega_{kj}^i(h)} P_m(D) \right] \times \left[ \prod_{n \notin \Omega_k^i(h), n \neq i, n \neq j} (1 - P_n(D)) \right].
\] (5.33)

Note that, although the above equation looks similar to Equation 5.28, they have different meanings. In the ZebraNet’s history-based scheme, the transmission from node \( j \) to node \( i \) happens only when node \( i \) has the highest delivery probability among all nodes in \( A_D \). Thus we only consider the situation where all other nodes within \( A_D \) have lower delivery
probabilities than node \( i \) (i.e., the situation where all nodes in \( A_D \) except node \( i \) belong to \( \Theta_{ij} \)). While in the replication-based scheme, because node \( j \) transmits to node \( i \) whenever node \( i \) has higher delivery probability than node \( j \) (even though node \( i \) is not the sensor with the highest delivery probability among all nodes in \( A_D \)), we consider any \( k \) nodes other than node \( i \) and node \( j \).

Therefore, the probability that sensor \( j \) can transmit to sensor \( i \) is

\[
Q_{ji} = \int \int_{A \rightarrow A_S} p_j(D) \times P_i(D) \times \sum_{k=0}^{N-2} \frac{\alpha_{ij}^k(D)}{k+2} dD,
\]

and consequently

\[
f'_i = \sum_{j \in \Theta_i} (1 - \wp_j^0) \times Q_{ji},
\]

which can be calculated in a recursive way, or

\[
f'_i \approx \sum_{j \in \Theta_i} Q_{ji},
\]

if the network is heavily loaded with \( \wp_j^0 \approx 0 \).

We then calculate the the nodal service rate \( \mu_i \) and the nodal message arrival rate \( \lambda_i \) following Equations 5.6 and 5.9, respectively, which lead to the same results shown in Equation 5.32. Accordingly, the nodal average queue size \( L_i \) and the average queue size of the network \( L \) can be easily derived using results obtained in Sec. 5.1.4.

Since multiple copies may be generated for each message, Equation 5.14 and Equation 5.16 cannot be used to calculate the average delay \( \Delta \) and the system delivery ratio \( \Upsilon \). However, we can estimate the data delivery ratio by tracking the overall transmission process of a message at each node, as elaborated below.

When a message \( M \) is generated by node \( j \), it has a probability of \( \wp_j^K \) to be dropped, and \( 1 - \wp_j^K \) to be transmitted to other nodes, including sink nodes or other sensor nodes.

To facilitate our discussion, we first consider one time slot when node \( j \) moves to location \( D \). Following the similar approach used in Equation 5.33 and Equation 5.34, we calculate
the average probability that node $j$ can transmit message $M$ to a node $i$ ($i \in \Psi_j$ where $\Psi_j$ denote the set of nodes with higher delivery probability than node $j$) in one time slot, i.e., $Q_{ji}$.

Because we are considering FCFS queue, as long as a message is not dropped upon arrival, it will finally be sent out successfully, either to the sink nodes or other sensor nodes. Thus the probability that message $M$ is sent from node $j$ to node $i$ in the $k^{th}$ time slot after it is moved to the head of the queue, if it is neither dropped initially nor transmitted in the previous $k - 1$ time slots, is

$$Q_{ji} \times (1 - S_j)^{k-1},$$  \hspace{1cm} (5.37)

where $S_j$ is the total probability that sensor $j$ can transmit to any other nodes in one time slot (see Equation 5.6), including sink nodes and sensor nodes with higher delivery probability. Consequently, the overall probability that the message is transmitted from $j$ to $i$ is

$$\epsilon_{ji} = (1 - \rho_j^K) \times \sum_{k=1}^{\infty} Q_{ji} \times (1 - S_j)^{k-1}$$

$$= (1 - \rho_j^K) \times \frac{Q_{ji}}{S_j}. \hspace{1cm} (5.38)$$

Similarly, the overall probability that the message $M$ is transmitted from node $j$ to any sink node directly is

$$\epsilon_{j\Phi} = (1 - \rho_j^K) \times \frac{S_j(\Phi)}{S_j}, \hspace{1cm} (5.39)$$

where $S_j(\Phi)$ is the probability that sensor $j$ can transmit to the sink nodes directly in one time slot.

Let $\zeta_j$ denote the probability that sensor $i$ can deliver a message to the sink nodes either directly or via other sensors. Then,

$$\zeta_j = \epsilon_{j\Phi} + \sum_{i \in \Psi_j} \epsilon_{ji} \times \zeta_i. \hspace{1cm} (5.40)$$
It is obvious that $\Psi_i \subset \Psi_j$ if $i \in \Psi_j$. Thus, following Algorithm 2, $\zeta_j$ can be safely calculated in a recursive way. Consequently, the average system delivery ratio is

$$\Upsilon = \frac{\sum_{j=1}^{N} \zeta_j}{N}. \quad (5.41)$$

**Algorithm 2** *getDeliveryRatio*(nodeID)

1. $j = \text{nodeID}$
2. $\zeta_j = 0$
3. for all $i \in \Psi_j$ do
   1. calculate $\epsilon_{ji}$
4. end for
5. for all $i \in \Psi_j$ do
   1. $\zeta_i = \text{getDeliveryRatio}(i)$
   2. $\zeta_j = \zeta_j + \epsilon_{ji} \times \zeta_i$
6. end for
7. $\zeta_j = \zeta_j + \epsilon_{j\Phi}$
8. return $\zeta_j$

### 5.2.4 Further Discussion

If the message queue at each node is priority-based, instead of FCFS, we can still obtain partial results. When a new message is arriving while the queue is full, one messages, either the newly arriving message or one of the existing messages, must be dropped. Which message to drop depends on the priority scheme employed. However, this does not affect the nodal service rate, nodal message arrival rate, and the nodal effective message arrival rate. Thus, we arrive at the same nodal queuing characteristics, such as nodal average queuing length and nodal message dropping rate. The average message delay, however, can not be derived directly by using Little’s formula in this case, since it depends on the message dropping policy. Moreover, while the average queuing length of the network can be derived accordingly, the calculation of average delivery ratio becomes much more difficult.
Meanwhile, if a message is not dropped after transmission to other nodes, but reinserted back to the queue instead, the nodal message arrival rate becomes

\[ \lambda_i = f_i + f_i' + S_i(\Xi), \]

(5.42)

where \( S_i(\Xi) \) is the average service rate due to transmission to other sensor nodes. Thus, the average message arrival rate is very likely to be larger than the service rate, resulting the average queue length of node \( i \) to be close to \( K \). Consequently, the average queue length of the network is nearly \( K \times N \).

### 5.3 Simulations and Numeric Results

We have carried out extensive simulations to validate our proposed generic analytic model. Various data delivery schemes and nodal mobility patterns discussed in Sec. 5.2 are simulated and compared with analytic results. For each simulation setup, we run the simulation multiple times and average the collected results.

#### 5.3.1 Direct Delivery with Uniformly Distributed Mobility

We first simulate a DFT-MSN with direct delivery scheme and uniform mobility pattern. A varying number of sensors, together with 5 sink nodes, are initially deployed in an area of \( 100 \times 100 \text{ m}^2 \) according to uniform distribution. The transmission range of the sensor node is 9\( \text{ m} \). The nodal maximum queue size is 50, and the nodal message generating rate is 0.05 message per time slot. Both simulation and analytic results are presented in Fig. 5.1. As can be seen, the analytic results match very well with the simulation results. With more sensors being deployed in the field, the average delivery ratio decreases, while the average system queue length and the average message delay both increase, because more nodes compete for the limited bandwidth at the sink nodes, consequently resulting in lower service rate of individual sensor node.
<table>
<thead>
<tr>
<th>Node Number</th>
<th>Delivery Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node Number</th>
<th>Average System Queue Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node Number</th>
<th>Average Message Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

(a) Average delivery ratio.

(b) Average system queue length.

(c) Average message delay.

FIGURE 5.1. Direct delivery scheme with uniform distribution mobility pattern.
FIGURE 5.2. Direct delivery scheme with power-law distribution mobility pattern.
(a) Average delivery ratio.

(b) Average system queue length.

(c) Average message delay.

FIGURE 5.3. ZebraNet’s history-based scheme with power-law distribution mobility pattern.
5.3.2 Direct Delivery with Power-law Distributed Mobility

We also simulate the direct delivery scheme in a network with nodal mobility pattern following power-law distribution. A number of sensor nodes and 5 sink nodes are initially randomly deployed in an area of $60 \times 60 \text{ m}^2$. The sensor nodes then move around their initial locations following power-law distribution. The moving ranges, i.e., $l_1$ and $l_2$, are $0.1m$ and $20m$, respectively, while the exponent parameter $\beta = 2$. The nodal message generating rate is 0.005 message per time slot. Fig. 5.2 compares the simulation results and the analytic results, in terms of the average nodal delivery ratio, average nodal queue length, and average message delay. As can be seen, the analytic results again match very well with the simulation results. Similarly, with more sensor nodes being deployed, the delivery ratio decreases slightly, while the message delay and the system queue length increase, for the same reason mentioned before.

Note that, unlike those observed under uniform distributed mobility, the curves under power law distributed mobility are not smooth. This is because that under power law distributed mobility, the contact probability between each pair of nodes, and accordingly network performance, are highly related to the nodes’ initial locations, which are randomly generated in our simulations. Similar fluctuations are also observed on the results to be discussed in the following subsections due to the same reason.

5.3.3 ZebraNet’s History-based Scheme

The results of ZebraNet’s history-based scheme are presented in Fig. 5.3, under the same simulation setup discussed above in Sec. 5.3.2. As can be observed in Fig. 5.3, the simulation results and the analytic results are close. With more sensor nodes being deployed, the average delivery ratio of the network increases slightly and the message delivery delay decreases, which is in sharp contrast to the direct transmission scheme. This is reasonable since when more nodes are deployed, a sensor has better chance to
deliver its messages promptly to the neighboring nodes with higher delivery probabilities before the messages are dropped. Meanwhile, the average nodal queue length decreases slightly with node density, due to the increased service rate.

5.3.4 Simple Replication-based Scheme

The simulations of the simple replication-based scheme use the same set of parameters as those used for the ZebraNet’s history-based scheme. As shown in Figs. 5.4 & 5.5, the simulation results and the analytic results match well. We first evaluate the impact of the
FIGURE 5.5. Simple replication-based scheme with power-law distribution mobility pattern.
node density, with results presented in Fig. 5.4. When more sensor nodes are deployed, they have better chance to deliver their messages through the neighboring nodes, thus resulting in the slight increase of the average delivery ratio, as can be observed in Fig. 5.4(a). At the same time, the average queue length of the network increases with the increase of the number of sensor nodes (see Fig. 5.4(b)), because of the higher message arrival rate. Fig. 5.5 shows the impact of nodal maximum queue length. As can be seen, the system delivery ratio increases with larger buffer space allocated to the sensors, since each queue can then hold more messages, resulting in smaller dropping ratio. Meanwhile, the average queue size of the network increases since each sensor usually buffers more messages in its queue.
Chapter 6
Testbed Design

We have verified the basic ideas of the proposed DFT-MSN data delivery schemes via extensive simulations, as shown in previous chapters. In this chapter, we focus on testbed implementation and experiments based on the Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD) introduced in Sec. 3.4 of Chapter 3.

6.1 Testbed Implementation and Experiments

6.1.1 Testbed Implementation

We have built a testbed by using Xbow MICA2 sensors [128] for monitoring the noise level in the library [129]. A MICA2 sensor node has a 4-MHz, 8-bit Atmel microprocessor and 512 KB of non-volatile flash memory that can be used for data logging. Its radio bandwidth is 38.4 Kbaud. The testbed runs TinyOS 1.1.0 operating system. There are two types of nodes in our testbed, namely mobile sensors and sink node. Their functions and implementations are discussed below.

1) Mobile Sensor

A mobile sensor has three functions: collecting information and generating data messages, transmitting or relaying data messages, and recording necessary information for performance evaluation. In order to reduce energy consumption, a sensor only wakes up periodically for data acquisition and transmission. We employ two timers, namely sensing-timer and transmission-timer, to control sensor’s activities. The former controls the sensor’s sensing activity. When sensing-timer expires, the mobile sensor samples the current noise level and generates a data message. The format of the data message is
illustrated in Fig. 6.1. When a new message is generated, it is assigned a sequence number, which is used, along with source address, to uniquely identify the message. Each message also contains a counter field to record its delivery delay. Once a data message is generated, it is inserted into the data queue, sorted by the fault tolerance of the messages.

The transmission timer controls the data transmission of the sensor. When the transmission timer expires, the sensor contacts its neighboring nodes if there are any. More specifically, it initiates a beacon message, which contains its address, its current delivery probability, and the minimal fault tolerance of its messages, to its neighbor nodes. Upon receiving a beacon message, the neighboring node decides whether to reply or not, based on deliver probability and buffer status. Then a message may be transmitted (depending on the reply) according to Algorithm 1 in Chapter 3. After transmission, the data queues of the sender and receiver are updated accordingly.

For performance evaluation, the sensors also record some information, such as the total number of sensing-data messages, the total number of transferred messages, and the number of dropped messages due to buffer overflow. This information is stored in the EEPROM and refreshed periodically. We retrieve these data after experiment for calculating the delivery ratio and overhead.

2) Sink Node

The sink node consists of one MICA2 sensor connected to a laptop via UART. The MICA2 at the sink is very similar to other sensors. The only difference is that it has no data queue. Upon receiving a data message, it will record the message in a text file on the laptop.

6.1.2 Experiment Setup

We have carried out a small scale experiment with six MICA2 nodes attached to students who move in the Dupré Library of our university. As shown in Fig. 6.2, the mobile sensor
nodes are initially scattered in three different areas, i.e., the reading area, the bookshelf area, and the computer service area. Each area has two boundaries, namely movement boundary and communication boundary. The former limits the nodal mobility in each area. The latter indicates the maximum radio transmission range of sensors in each area. The communication boundaries of any two areas partially overlap with each other. Note that, the nodes within transmission range may not always be able to communicate with each other because of the lack of line-of-sight (due to the bookshelves, computers, walls, etc.). Generally, a node only moves within the movement boundary of the area where it is currently located, while periodically it may move out to another area with certain probability. The moving speed of the mobile nodes is around 0-3 m/s. Each sensor has a maximum data queue size of 50 messages. The sensing-timer is set to be 60,000 binary milliseconds (1 binary millisecond equals 1/1024 second), while the transmission-timer is set as 10,000 binary milliseconds. In addition, we set $\alpha = 0.02$ and $\Re = 0.9$ in this experiment.
6.2 Experimental Results

We run the experiment for a period of $T$. Note that, due to the long delay of data delivery, many newly generated data messages are still stored at the intermediate nodes by the end of $T$. Since the sink node doesn’t receive these data messages, we may falsely assume they are lost, resulting in a low delivery ratio. This problem will become negligible when $T \to \infty$. But a large $T$ is not practical for our experiment that relies on the student volunteers to carry sensors. Alternatively, we choose a small $T$ (e.g., $T = 20$ min). After $T$, we do not terminate the experiment immediately, but instead, continue it for another period $t$ (e.g., $t = 10$ min), during which the sensors do not generate new data messages.

The experimental results are summarized in Table 6.1, where “generated messages” refer to the number of new messages generated by the mobile sensors; “transmitted messages” refer to the total number of message copies transmitted (i.e., including duplicate messages); “received messages” refer to the number of unique message copies received by the sink (i.e., excluding duplicate messages). As we have observed from our experiments, the proposed DFT-MSN data delivery scheme is efficient, with a total
### TABLE 6.1. Experimental results

<table>
<thead>
<tr>
<th></th>
<th>During $T$</th>
<th>During $t$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated messages</td>
<td>120</td>
<td>0</td>
<td>120</td>
</tr>
<tr>
<td>Transmitted messages</td>
<td>347</td>
<td>288</td>
<td>635</td>
</tr>
<tr>
<td>Received messages</td>
<td>87</td>
<td>29</td>
<td>116</td>
</tr>
<tr>
<td>Delivery ratio (%)</td>
<td>/</td>
<td>/</td>
<td>96.67</td>
</tr>
<tr>
<td>Average delay (m)</td>
<td>/</td>
<td>/</td>
<td>5.8</td>
</tr>
</tbody>
</table>

delivery ratio higher than 96% and average delay around 5.8 minutes. These results have been verified by multiple similar experiments with running time from 30 minutes to 2 hours. The high overhead is reasonable, given the very low network connectivity and nodal delivery probability. In addition, we observe a few messages (including duplicate message copies) dropped during the experiments due to buffer overflow (e.g., about 10 messages dropped during a 30-minute period). Based on the small-scale testbed, a university-wide large-scale experiment will be carried out in the future.
Chapter 7
Summary

In this dissertation, we study the Delay/Fault-Tolerant Mobile Sensor Networks (DFT-MSN’s) for pervasive information gathering. The major contributions are summarized as follows.

- **Architecture**

  We have proposed a two-layer hierarchical architecture for DFT-MSN. The upper layer is a traditional backbone network with wireless access interface. The lower layer consists of two types of nodes, the wearable sensor nodes and the high-end sink nodes. The former are attached to mobile objects, gathering target information and forming a loosely connected mobile sensor network for information delivery. The latter are normally more powerful nodes (e.g., PDA, laptop, etc.), collecting data from the sensor nodes and forwarding them to the end user through the backbone network. We have also identified several unique characteristics of DFT-MSN, i.e., nodal mobility, sparse connectivity, delay tolerability, fault tolerability, and buffer limit, which distinguish DFT-MSN from the traditional sensor networks.

  Meanwhile, DFT-MSN is fundamentally an opportunistic network, where the communication links exist only with certain probabilities, making design of effective data delivery protocol challenging.

- **Data Delivery Schemes**

  Due to the unique characteristics of DFT-MSN, mainstream approaches used for data delivery in traditional sensor networks cannot work effectively in DFT-MSN.

  After studying two basic approaches (i.e., direct transmission and flooding) by using
queueing theory and statistics, we have introduced two simple and effective DFT-MSN data delivery schemes, namely Replication-Based Efficient Data Delivery Scheme (RED) and Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD). The RED scheme utilizes the erasure coding technology in order to achieve the desired data delivery ratio with minimum overhead. Whereas the FAD scheme employs the message fault tolerance, which indicates the importance of the messages, to help message forwarding and queue management. Our simulation results have shown that both schemes achieve high message delivery ratio with acceptable delay.

- **Protocol Design**

  We have observed that without end-to-end connections, routing in DFT-MSN becomes localized and ties closely to the medium access control, naturally calling for merging layer-3 and layer-2 protocols in order to reduce overhead and improve network efficiency. To this end, we have proposed a cross-layer data delivery protocol, which consists of two phases, i.e., the asynchronous phase and the synchronous phase. In the first phase, the sender contacts its neighbors to identify a set of appropriate receivers. Since no central control exists, the communication in the first phase is contention-based. In the second phase, the sender gains channel control and multicasts its data messages to the receivers. Furthermore, we have identified several optimization issues in these two phases, with solutions provided to reduce the collision probability, and to balance between link utilization and energy efficiency. Extensive simulations have been carried out for performance evaluation. Our results have demonstrated that the proposed cross-layer data delivery protocol for DFT-MSN achieves a high message delivery ratio with low energy consumption and an acceptable delay.
• **Analytic Study**

In order to better understand the characteristics of DFT-MSN, we have also established a generic queuing analytic model based on queuing theory and statistic analysis. The inputs of the analytic model are the data delivery scheme employed and the nodal mobility pattern, while the outputs are the queuing characteristics of the network. We have also exemplified the generic analytic model with several representative data delivery schemes (including direct transmission, ZebraNet, and Simple Replication-based Data Delivery) and nodal mobility patterns (such as uniform and power-law distributions). To validate our analytic models, we have carried out extensive simulations and observed a good match between analytic and simulation results.

• **Testbed Design and Implementation**

To gain significant experimental knowledge and experience, we have implemented the FAD data delivery scheme and established a DFT-MSN testbed by using Crossbow sensors. Small scale experiments have been carried out to further demonstrate the effectiveness and efficiency of DFT-MSN.

This dissertation provides the first step to study DFT-MSN for pervasive information gathering. We have focused on the effective data delivery by considering the unique characteristics of DFT-MSN. In our future work, we will enhance our current research by investigating the following issues.

• **Privacy Issue**

Since DFT-MSN is a multi-hop network, the message from a node may be transmitted to some intermediate nodes before it reaches the destination. How to keep the data confidential during its transmission is a big concern. Public-key based algorithm can be used for encrypting messages. However, it can be very challenging
to design an effective and scalable key assigning and management scheme for such networks with various nodal mobility, large number of nodes, and limited transmission and computation capabilities. I plan to employ both pre-distribution and dynamic assigning schemes to address this problem. Statistic analysis will also be used to evaluate the performance of the key management scheme.

**Proper Pricing Policy for Facilitating Data Delivery**

DFT-MSN is a multi-hop network, where data messages may be delivered to the destination through multiple intermediate nodes. However, data relaying consumes precious resources at these intermediate nodes. It is very common that an intermediate node rejects data forwarding for some other nodes for various reasons. Meanwhile, if a malicious or greedy node keeps injecting nonsense data into the network, the system performance can be degraded dramatically. Therefore, for a commercial pervasive data gathering application, it is crucial to incorporate a proper pricing policy into the data delivery scheme, in order to achieve a certain level of fairness and robustness.

**Prediction-based Approach**

In our current work, the history information is employed to help making decision on data transmission in opportunistic networks. In particular, if a node meets another node frequently in the past, it's very likely that these two nodes will meet again in the near future. This is true in many scenarios, such as human society, animal community, etc. On the other hand, there may also exist some predictable patterns in the system, which can also be employed to help data delivery. Such predictable patterns include nodal mobility patterns, nodal behavior patterns, network connectivity patterns, etc. One typical example is people’s daily routine behavior: we go to work at 9:00AM and leave for home at 5:00PM. In this example, the
connectivity among the employees becomes strong during daytime but gets weak after 5:00PM. When we design protocol for applications in this scenario, we may want the nodes to transmit data more aggressively during daytime, while lower their duty cycle in the night. In future research, I plan to incorporate both history and prediction information into the design of data delivery scheme.

- **Adjustable Transmission Power Level**

Power efficiency is also a very important issue in DFT-MSN. In our current protocol design, all the nodes in the network are assumed to have a fixed transmission power level. Due to the lack of central control and the rapid change of network topology, it is difficult to assign an optimal transmission power level to all the nodes in DFT-MSN. An alternative way is to let each individual node dynamically tune its transmission power level according to its local information (e.g., number of neighbors, delivery probability, buffer status, remaining energy, etc.). Thus, an algorithm is needed to dynamically estimate proper transmission power level.

- **Analytic Model**

Although various data delivery schemes have been proposed for delay/fault tolerant networks, the analytic studies are still not enough. In the future, I plan to enhance the current analytic model of DFT-MSN by considering more generic scenarios. More specifically, instead of the simplest FCFS queue, I may study other more realistic queue management schemes, such as priority-based schemes. In addition, other more complicated data delivery schemes are to be considered.
References


[60] http://robotics.eecs.berkeley.edu/ pister/SmartDust/.


ABSTRACT

This work focuses on the Delay/Fault-Tolerant Mobile Sensor Networks (DFT-MSN’s) for pervasive information gathering, which play a key role in many promising applications, such as air quality monitoring, ubiquitous healthcare, and biological research. Due to the unique characteristics of DFT-MSN, such as sensor mobility, loose connectivity, fault tolerability, delay tolerability, and buffer limit, designing an efficient data delivery scheme is challenging.

We first study two basic approaches, namely, direct transmission and flooding, by using queuing theory and statistics. Based on the analytic results that show the tradeoff between data delivery delay/ratio and transmission overhead, we propose two simple and effective DFT-MSN data delivery schemes, namely Replication-Based Efficient Data Delivery Scheme (RED) and Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD). The RED scheme utilizes the erasure coding technology in order to achieve the desired data delivery ratio with minimum overhead, while the FAD scheme employs the message fault tolerance, which indicates the importance of the messages, for the same purpose. Our results show that both schemes achieve high message delivery ratio with acceptable delay.

We also observe that without end-to-end connections due to sparse network density and sensor node mobility, routing in DFT-MSN becomes localized and ties closely to medium access control, which naturally calls for merging Layer 3 and Layer 2 protocols in order to
reduce overhead and improve network efficiency. To this end, we develop a cross-layer data delivery protocol for DFT-MSN, with various optimization approaches proposed to address the tradeoff between link utilization and energy efficiency.

In order to better understand the characteristics of DFT-MSN, we also introduce a generic queuing analytic model based on queuing theory and statistic analysis. To validate our analytic models, we have carried out extensive simulations and observed a good match between analytic and simulation results. Beyond theoretical work, we also implement the FAD data delivery scheme and establish a DFT-MSN testbed by using Crossbow sensors. A small scale experiment is carried out to further evaluate the effectiveness and efficiency of DFT-MSN.
BIOGRAPHICAL SKETCH

Yu Wang was born on April 6, 1978, in Mianyang, P. R. China. He finished his B.S. and M.S. degrees at the University of Electronic Science and Technology of China in 2000 and 2003, respectively. In January of 2004, he joined the University of Louisiana at Lafayette to pursue his Ph.D. in Computer Science. He is currently a candidate for the degree of Doctor of Philosophy in Computer Science, which is to be awarded in May 2007.