Modeling and Optimization of Wireless Routing

Chuan Han

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Yaling Yang, Chair
Y. Thomas Hou
Jung-Min Park
R. Michael Buehrer
Hanif D. Sherali

May 7, 2012
Blacksburg, Virginia

Keywords: Wireless Routing, Metric Compatibility, Route Repair, Node Placement,
Convergecast Delay

Modeling and Optimization of Wireless Routing

Chuan Han

ABSTRACT

Recently, many new types of wireless networks have emerged, such as mobile ad hoc networks (MANETs), cognitive radio networks (CRNs) and large scale wireless sensor networks. To get better performance in these wireless networks, various schemes, e.g., metrics, policies, algorithms, protocols, etc., have been proposed. Among them, optimal schemes that can achieve optimal performance are of great importance. On the theoretical side, they provide important design guidelines and performance benchmarks. On the practical side, they guarantee best communication performance with limited network resources. In this dissertation, we focus on the modeling and optimization of routing in wireless networks, including both broadcast routing, unicast routing, and convergecast routing. We study two aspects of routing: algorithm analysis and quality of service (QoS) analysis. In the algorithmic work, we focus on how to build optimal broadcast trees. We investigate the optimality compatibility between three tree-based broadcast routing algorithms and routing metrics. The QoS work includes three parts. First, we focus on how to optimally repair broken paths to minimize impact of path break in MANETs. We propose a provably optimal cached-based route repair policy for real-time traffic in MANETs. Second, we focus on the impact of secondary user (SU) node placement on SU traffic delay in CRNs. We design SU node placement schemes that can minimize the multi-hop delay in CRNs. Third, we analyze the convergecast delay of a large scale sensor network which coexists with WiFi nodes. We derive a closed form delay formula, which can be used to estimate sensor packet convergecast delay given the distance between a sensor node and the sink node together with other networking setting parameters. The main contributions of this dissertation are summarized as follows:

1. **Optimality compatibility study between tree-based broadcast routing algorithms and routing metrics** Broadcast routing is a critical component in the routing design. While there are plenty of routing metrics and broadcast routing schemes in current literature, arbitrary combination of broadcast routing metrics with broadcast tree construction (BTC) algorithms may not result in optimal broadcast trees. In this work, we study the requirement on the combination of routing metrics and BTC algorithms to ensure optimal broadcast tree construction. When a BTC algorithm fails to find the optimal broadcast tree, we define that the BTC algorithm and the metric are not optimality compatible. We show that different BTC algorithms have different requirements on the properties of broadcast routing metrics. The metric properties for BTC algorithms in both undirected network topologies and directed network topologies are developed and proved. They are successfully used to verify the optimality compatibility between broadcast routing metrics and BTC algorithms.

2. **Optimal cache-based route repair policy for real-time traffic in mobile ad hoc networks** Real-time applications in ad hoc networks require fast route repair mechanisms to minimize the interruptions to their communications. Cache-based route repair schemes are popular choices since they can quickly resume communications using cached backup paths after a route break. In this work, through thorough theoretical modeling of the cache-based route repair process, we derive a provably optimal cache-based route repair policy. This optimal policy considers both the overhead of the route repair schemes and the promptness of the repair action. The correctness and advantages of our optimal policy are validated by extensive simulations.

3. **Optimal secondary user node placement study in cognitive radio networks** Information propagation speed (IPS) in a multi-hop CRN is an important factor that affects the network's delay performance and needs to be considered in network planning. The impact of primary user (PU) activities on IPS makes the problem of analyzing IPS in multi-hop CRNs very challenging and hence unsolved in existing literature. In this work, we fill this technical void. We establish models of IPS in multi-hop CRNs and compute how to maximize IPS in two cases. The first case, named the maximum network IPS, maximizes IPS across a network topology over an infinite plane. The second case, named the maximum flow IPS, maximizes the IPS between a given pair of source and destination nodes separated by a fixed distance. We reveal that both maximum IPSs are determined by the PU activity level and the placement of SU relay nodes. We design optimal relay placement strategies in CRNs to maximize these two IPS under different PU activity levels. The correctness of our analytical results is validated by simulations and numerical experiments.

4. **Convergecast delay analysis of large scale sensor networks coexisting with WiFi networks**

   Due to the increasing popularity of wireless devices, such as WiFi (IEEE 802.11) and ZigBee (IEEE 802.15.4), the ISM bands have become more and more crowded. Since ZigBee is the de facto radio technology of sensor networks, coexistence of WiFi networks and sensor (ZigBee) networks is challenging because of the great heterogeneity between WiFi and ZigBee technologies. In the presence of interference from WiFi and other sensor nodes, the performance of sensor networks is not clearly understood. In this work, we study delay performance of a large scale sensor network which coexists with WiFi networks. Given the distance from the sensor node to the sink node, we are interested in the expected delay of sensor packets to reach the sink node in the presence of both WiFi and sensor interference. We formulate the delay analysis problem as a two priority M/G/1 preemptive repeat identical queueing system, and analyze the delay using queueing theory and probability theory. First, we use a path probabilistic approach to derive the expected delay. Second, we develop a simplified linear approximation model for delay analysis. The correctness of both models is validated by NS2 simulations.

This work was supported in part by the US National Science Foundation under grant CNS-0831865 and the Institute for Critical Technology and Applied Science (ICTAS) of Virginia Tech.

To my family

Acknowledgments

I would like to sincerely thank my Ph.d. advisor, Prof. Yaling Yang. She not only constantly supported my research work during my Ph.d. but also gave me useful guidance to my research and future career development. Her patience and confidence on me made this dissertation possible. I will never forget numerous discussions with her, kind help and advice from her on my presentation and how to plan my future career. Her persistence and dedication to research work is my role model. Her optimistic attitude and lifestyle also helped me a lot. Without her help, it would become more difficult to accomplish this Ph.D. dissertation, and also my Ph.D. life would become less joyful.

I am lucky to have exceptional committee members: Prof. Tom Hou, Prof. Jung-Min park, Prof. R. Michael Buehrer, and Prof. Hanif D. Sherali. I greatly appreciate their suggestions and comments on my Ph.D. work, and their willingness to bend their schedules to accommodate meetings and exams.

During my Ph.D. study, I also had fruitful cooperation and insightful discussion with Yujun Li, and Yongxiang Peng. It was a joyful experience to me. Their work and inputs have become part of this dissertation. I am very fortunate to have long time friendship with my old friends, Dr. Jun Wang, Dr. Yu Wen, Dr. Jie Chen, and Dr. Huan Song. Their encouragement and enlightenment on various aspects of my Ph.D. life helped me out of many difficult situations.

I also want to thank my fellow friends in my lab whom I have worked with: Zhenhua Feng, Cheewo Na, Kaigui Bian, Isha Khanna, Siyu Zhan, Han Tian, Xuyang Ding, Yi Tang, Ting Wang, Jingyao Zhang, Bo Gao, Hao Wu. It was great to have their company in my Ph.D. years. In addition, I would like to thank my other friends at Virginia Tech: Ray Wang, Dao Zhou, An He, Xuetao Chen, Xinhao Ye, Zheying Guo, Guanying Wang, Canming Jiang, Liguang Xie, Min Li, etc.. Their friendship made my Ph.D. life more memorable and joyful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Objectives

The optimal performance study of wireless networks is of great importance because of its theoretical and practical significance. In this dissertation, we focus on modeling and optimization of routing in wireless networks. Two aspects of routing are studied: algorithm analysis and QoS analysis. In the algorithmic aspect, the focus is on broadcast routing. We study the optimality compatibility between broadcast tree construction algorithms and broadcast routing metrics. Our analysis shows that arbitrary combination of broadcast tree construction algorithms with broadcast routing metrics may lead to inferior performance. This motivates us to identify the necessary and sufficient routing metric conditions for several well-known broadcast tree construction algorithms to find the optimal broadcast tree. After the algorithmic work, we move to QoS analysis of wireless routing. It includes three parts. In the first part, we focus on the unicast routing QoS in MANETs. We design a provably optimal cached-based route repair policy for real-time traffic in MANETs. Because of node movement and fading channels, in MANETs the path in use may break, and the routing protocol may either flood the whole network to get new paths or use cached old paths. The objective of this work is to design a provably optimal route repair policy to achieve the design trade-off between flooding and using cached paths. In the second part, we focus on the impact of node placement on unicast routing performance in CRNs. We study how to place SU nodes optimally so that the unicast multi-hop information propagation delay in random CRNs is minimized. We propose the IPS to capture the impact of node placement on multi-hop delay. The goal is to find the maximum IPS by optimally placing SU nodes. In the third part, we are interested in the convergecast delay of a large scale sensor network coexisting with WiFi nodes. In such a network, the convergecast traffic causes delay to sensor packets. At the same time, since sensor nodes and WiFi nodes work in the same ISM band and WiFi transmission power is much higher than sensor node transmission power, existence of WiFi traffic causes additional delay to sensor packets. The impact of these two

delay components is unclear in the current literature. In this work, we derive the expected delay of sensor convergecast traffic given the distance from a sensor node to the sensor sink node as well as other network setting parameters.

1.2 Summary of Contributions

1.2.1 Optimality compatibility study between tree-based broadcast routing algorithms and routing metrics

Broadcast routing is an important component for network management and administration. Without effective broadcast routing, network performance may degrade. The effectiveness of broadcast routing can be measured by different metrics in different application scenarios. In some situations, multiple performance factors may be jointly considered. We show that random combination of broadcast tree construction (BTC) algorithms with routing metrics may lead to inferior routing performance. The objective of this work is to provide important optimality compatibility check for the optimality compatibility and consistency of several well-known BTC algorithms under various routing metrics. The well-known BTC algorithms discussed in this work are Prim's algorithm [10, 29], the generic edge-adding algorithm [10, 29], and Edmonds' algorithm [15]. Our unique contribution is two-fold. First, using a unique algebra model of broadcast routing, we identify the necessary and sufficient routing metric properties for BTC algorithms to find optimal broadcast trees. Second, we identify the conditions for BTC algorithms to find consistent and unique broadcast trees and discuss how this affects distributed broadcast routing protocol design.

1.2.2 Optimal cache-based route repair policy design for real-time traffic in mobile ad hoc networks

In MANETs, constant node movement and fading channels may cause frequent path break, resulting in communication interruption. For any flooding-based routing protocol, after flooding there may be multiple paths discovered, some of which may be cached for future use. When the path in use is broken, the protocol may either reflood the whole network to get fresh new paths or use a cached path. Since the flooding has a high overhead and a cached path may also be broken, there is design trade-off between flooding the whole network and using a cached path. In this work, we provide a mathematical analysis on this trade-off point and derive a provably optimal cache policy that is the best strategy between flooding and cache-based repair. Our unique contributions are listed as follows: While using cached paths for route repair has been proposed in many existing literature, we are the first to formulate an analytical framework to identify the optimal trade-off point between cache-based route repair and flooding-based route discovery in MANETs. This

analytical framework accurately captures the factors that affect the performance of route repair strategies. These factors include path length, mobility of nodes and network size. Based on our analytical framework, an easy-to-implement optimal cache policy is proposed and proved for cache-based route repair schemes. This policy finds the optimal trade-off point between flooding and cache-based route repair. Its correctness and advantages are analytically proved and then validated by simulations.

1.2.3 Optimal secondary user node placement study in cognitive radio networks

While there have been many research efforts on CRNs, delay performance is still a less explored area. Existing work on delay in CRNs either focuses on analysis in one hop network or tries to optimize one hop delay. In this work, we try to analyze the multi-hop delay performance in multi-hop CRNs. Different from existing work which is only based on static PU traffic activity, our work is based on temporal variability of PU traffic. Our analysis is based on a metric, IPS, which can capture impact of SU node placement on multi-hop delay. First, we establish a model of IPS in CRNs and categorize IPS maximization problems into two cases. The first case, named the maximum network IPS problem, maximizes the IPS across a network topology over an infinite plane. The second case, named the maximum flow IPS problem, maximizes the IPS between a given pair of source and destination nodes separated by a fixed distance. Second, we reveal that both the maximum network IPS and the maximum flow IPS are determined by the PU activity level and the placement of SU relay nodes. We have designed numerical methods to compute the two maximum IPSs and built optimal relay placement strategies to realize these two maximum IPSs under different PU activity levels. Finally, the correctness of our analytical results is validated by extensive simulations and numerical experiments.

1.2.4 Convergecast delay analysis of large scale sensor networks coexisting with WiFi networks

Wireless sensor networks have been under intensive and extensive research during the past decades. However, performance of sensor networks coexisting with other networks is less explored. The past decade has witnessed increasing popularity of WiFi (IEEE 802.11) and ZigBee (IEEE 802.15.4) devices. Since ZigBee is the de facto radio technology of sensor networks, coexistence of WiFi networks and sensor (ZigBee) networks is challenging because of the great heterogeneity between WiFi and ZigBee technologies. In the presence of interference from WiFi and other sensor nodes, the performance of sensor networks is not clearly understood. In this work, we study delay performance of a large scale sensor network which coexists with WiFi networks. Given the distance from the sensor node to the sink node, we derive the expected delay of sensor packets to reach the sink node in the presence of both

WiFi and sensor interference. We formulate the delay analysis problem as a two priority M/G/1 preemptive repeat identical queueing system, and analyze the delay using queueing theory and probability theory. First, we use a path probabilistic approach to derive the expected delay. Second, we develop a simplified linear approximation model for delay analysis. The correctness of both models is validated by NS2 simulations.

1.3 Outline

The rest of this dissertation is organized as follows. In Chapter 2, we review some background knowledge for this dissertation work. In Chapter 3, we study the optimality compatibility between three well-known broadcast routing algorithms and various metrics. In Chapter 4, we design a provably optimal cached-based route repair policy for real-time traffic in MANETs. In Chapter 5, we investigate how to place SU nodes in a CRN, and derive the maximum IPS in CRNs. In Chapter 6, we analyzed the convergecast delay of a large scale sensor network coexisting with WiFi networks.

Chapter 2

Background

This chapter introduces the basic background knowledge required by this research. It will overview existing mobility models in mobile ad hoc networks, and the basic concepts and principles of cognitive radio networks.

2.1 Mobility Models in Mobile Ad hoc Networks

Mobility models are important to both analysis or protocol design of MANETs. An ideal mobility model should accurately capture realistic node mobility in a MANET. In the current literature [7], there are two types of models used to capture node mobility: trace model and synthetic model. In this Section, we only focus on the synthetic model. We introduce four synthetic mobility models in MANETs: random walk mobility model, random waypoint mobility model, random direction mobility model, and semi-Markov smooth mobility model.

In the random walk mobility model, each mobile node moves from its current position to a new position by randomly choosing a direction and speed at which it travels. The direction and speed were chosen from ranges, $[0, 2\pi)$ and $[v_{min}, v_{max}]$, respectively. Each movement lasts either a constant time interval t or a constant distance d. The new position is updated when the movement is finished. If a mobile node reaches the simulation boundary, it bounces off the simulation border with angle determined by the incoming direction, and it continues along the new path. The random walk mobility model is a memoryless mobility model. The current direction and speed are independent of the past direction and speed. Because of memorylessness, the mobile node may experience sudden stops, sudden acceleration and sharp turns.

In the random waypoint mobility model, there exist pause times between changes of direction and/or speed. A mobile node stays in a position for a while (pause time). Once this period expires, the mobile node chooses random destination in the simulation area, a speed v that

is uniformly distributed in $[v_{min}, v_{max}]$. The mobile node then travels to the new destination at the selected speed v. When it arrives, it stays there for a specified time before moving again. The random waypoint mobility model will generate density waves. A density wave is the clustering of nodes in one part of the simulation region.

In the random direction mobility model, a mobile node chooses a random direction and travels along this direction. When the node reaches the boundary, it pauses for a certain amount of time. Then, it chooses another direction and continues moving.

In the semi-Markov smooth mobility model, a mobile node will experience four phases: speed acceleration, stable speed, speed deceleration, and pause phase. In the speed acceleration phase, the mobile node randomly choose a random direction, speed and amount of time from $[0, 2\pi)$, $[v_{min}, v_{max}]$ and $[t_{min}, t_{max}]$, respectively. In this phase, the mobile node accelerates its speed from 0 to its target speed along the chosen direction. In the stable speed phase, the mobile node randomly chooses the phase length. The speed and direction change based on the previous speed and direction according to a memory level parameter. In the speed deceleration phase, the mobile node randomly chooses a direction and amount of time to come to a full stop. The node does not change direction during this phase. In the semi-Markov mobility model, node movement is dependent on moving history, and follows the physical law of a smooth motion. The nodal distribution is uniform during the movement.

2.2 Cognitive Radio Network

The last decade has witnessed wide and rapid deployment of wireless devices, which significantly increases the demand of spectrum resources. National regulator bodies like federal communications commission (FCC) assign spectrum licenses to spectrum holders on a long time basis in a large region. Recently, field experiment study shows that currently allocated spectrum bands are underutilized [1]. The conventional fixed spectrum assignment policy does not fit into the continuously growing demand of spectrum. This motivates the cognitive radio (CR) technology [1], which is able to use spectrum opportunistically and intelligently.

The motivating idea of a cognitive radio network is to use the spectrum more efficiently through opportunistic spectrum use. In a cognitive radio network, the spectrum license holders are primary users (PUs) who have a higher priority to access the spectrum. Their licensed spectrum bands may be underutilized either temporally or geographically. The spectrum utilization efficiency is increased by letting secondary users (SUs) to use these unused spectrum left by PUs. SUs have a lower priority to access the spectrum. When a PU is using a channel, the SU cannot use the same channel at the same time. A SU has to vacate its current channel, when a PU reoccupies the same time. The interaction between PUs and SUs in cognitive radio networks is based on cognitive radios which can adapt to the changing radio environment. These advanced radios are able to dynamically change transceiver parameters according to the observed radio environment, like traffic pattern,

available bandwidth, transmission power, modulation schemes, etc.

Chapter 3

Optimality Compatibility between Broadcast Routing Algorithms and Metrics

3.1 Introduction

Broadcast routing is a critical component of network administration and management. It delivers and updates various network control information. Without effective broadcast routing, the whole network may degrade into chaos. Effective broadcast routing is especially critical in wireless networks due to the high cost of broadcast, the unreliable wireless channels and the limited available resource in wireless networks. Therefore, designing effective broadcast routing protocols is an important part in routing research and design.

The definition of "effective" broadcast routing, however, varies according to the application scenarios. In an energy-critical sensor network, broadcast routing may need to minimize energy consumption [41, 65, 40] or maximize network lifetime [30, 17, 18]. In an information-sensitive military network, broadcast routing may need to guarantee timely [46] and secure message delivery [16]. In many other application scenarios, multiple performance factors, such as delay, packet loss rate, bandwidth, etc., may need to be jointly considered for effective broadcast routing [2]. All these different performance requirements for broadcast routing are usually reflected in the design of routing metrics [4, 42], which guide the broadcast tree calculation algorithms to prefer one broadcast tree over the others.

While there are a lot of possible ways to design broadcast routing metrics, existing algorithms may not find the optimal broadcast trees (OBTs) for some of these routing metrics. For example, a common metric capturing the total broadcast routing energy consumption [58] is

$$w(T) = \sum_{i \in N_t(T)} \max_{(i,j) \in E(T)} \varepsilon_{ij}, \tag{3.1}$$

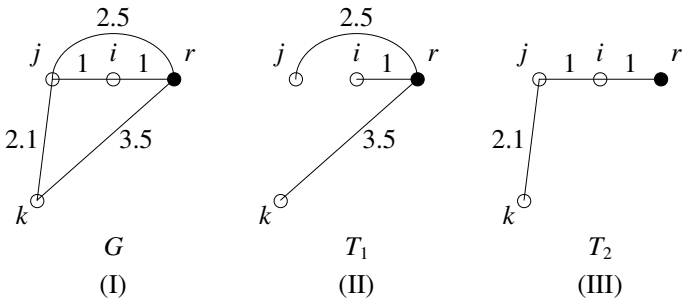


Figure 3.1: The network topology example

where $N_t(T)$ is the set of transmitting nodes in the broadcast tree T, $E(T)$ is the set of links in T, and ε_{ij} is the minimum energy consumption of transmitting a packet from node i to node j. While the metric definition in (3.1) is simple, finding the OBT based on this definition is very challenging. Consider the undirected network topology G consisting of four nodes r, i, j and k in Figure 3.1 (I), where the source node r of a broadcast session is marked as the black node, and the number associated with each edge is the ε_{ij} in (3.1). By definition (3.1), the minimum energy broadcast tree is $T_1 = \{ri, rj, rk\}$ in Figure 3.1 (II), whose total energy consumption is 3.5. However, following Prim's algorithm [10], which is a well-known algorithm for calculating the minimum spanning tree (MST) for undirected graphs, the broadcast tree is $T_2 = \{ri, ij, jk\}$ in Figure 3.1 (III), whose total energy consumption is 4.1. Hence, the combination of Prim's algorithm and the broadcast routing metric (3.1) fails to find the optimal broadcast routing tree.

The above example shows that a well-known broadcast tree construction (BTC) algorithms may not find the OBT in a network. This observation leads to two fundamental questions we seek to answer in this chapter. First, since well-known BTC algorithms may not be optimal for all routing metrics, what are the conditions for them to be optimal? Second, since distributed broadcast routing is desired in practical cases and a network generally has more than one OBTs, can these BTC algorithms create inconsistent routing decisions when they are implemented distributedly in a network? These questions are critical for the design of broadcast routing. Unfortunately, while similar questions have been answered for unicast routing [50, 51, 63, 37], the answers to these questions in broadcast routing are not clearly discussed.

In this chapter, we fill this critical void. The objective of our work is to provide important optimality compatibility check for the optimality compatibility and consistency of several well-known BTC algorithms under various routing metrics. The well-known BTC algorithms discussed in this chapter are Prim's algorithm [10, 29], the generic edge-adding algorithm [10, 29], and Edmonds' algorithm [15]. Our unique contribution is two-fold. First, using a unique algebra model of broadcast routing, we identify the necessary and sufficient routing metric properties for BTC algorithms to find OBTs. Second, we identify the conditions for BTC algorithms to find consistent and unique broadcast trees and discuss how this affects

Table 3.1: Notation

Symbol	Definition
$=$	Equivalent to
$A^-(n)$	The set of arcs that terminate at node n
$E(G)$	The edge set of graph G
$G = (N(G), E(G))$	The graph whose node set is $N(G)$ and edge set is $E(G)$
$G_1 \ominus G_2$	Remove the common edges of network topologies G_1 and G_2 from G_1
$G_1 \oplus G_2$	Merge of two network topologies G_1 and G_2
$N(G)$	The node set of graph G
$\partial(S)$	The edge cut of node set S
$\prec$	Lighter than
$\preceq$	Lighter than or equivalent to
$\succ$	Heavier than
a_{ij}	The arc emanating from node i and terminating at node j
$d^-(n)$	The indegree of node n
e_{ij}	The edge whose two end nodes are i and j
r	The root node of graph
$w(T)$	The weight of network topology T
$\|S\|$	The total number of elements in the set S

distributed broadcast routing protocol design.

The notation used in this chapter is summarized in Table 3.1. The remaining part of this chapter is organized as follows. In Section 3.2, the broadcast routing algebra is introduced. Sections 3.3 and 3.4 provide the necessary and sufficient metric properties required by broadcast routing for undirected network topologies and directed network topologies, respectively. A distributed broadcast routing protocol and some related properties are discussed in Section 3.5. The applications of the derived broadcast routing metric properties are presented in Section 3.6. Finally, Section 3.7 concludes this chapter.

3.2 Broadcast Routing Algebra

The analysis of the routing metric properties is based on our broadcast routing algebra that formulates the routing problem in an algebraic manner. This section introduces the broadcast routing algebra. Our algebra is different from Sobrinho's routing algebra [50]. Sobrinho's routing algebra can only operate over paths and hence can only capture unicast routing. Our broadcast routing algebra, on the other hand, can operate over arbitrary graphs and capture broadcast routing. Our routing algebra is also highly flexible. It not only can capture traditional routing metrics that define broadcast tree weight as linear link weight

aggregation, but also can capture more complex non-linear routing metrics.

3.2.1 Definition

Definition 1. *The broadcast routing algebra is defined as*

$$A = (\Sigma, \preceq, \oplus, \ominus, w(\cdot)), \tag{3.2}$$

where

- Σ is the set of signatures describing the characteristics of all the subgraphs of an original graph G. These characteristics may include each link's capacity, energy consumption, etc..
- The symbol $\preceq$ is the preference order operator, where $w(T_1) \preceq w(T_2)$ indicates that topology T_1 is better than or equivalent to topology T_2 under weight function $w(\cdot)$.
- The symbol $\oplus$ denotes the operator that joins two network topologies. For any edge $e_1 \in E(G_1)$ and node $n_1 \in N(G_1)$ in topology G_1, and any edge $e_2 \in E(G_2)$ and node $n_2 \in N(G_2)$ in topology G_2, we have e_1, e_2 in the edge set $E(G_1 \oplus G_2)$ and n_1, n_2 in the node set $N(G_1 \oplus G_2)$.
- The symbol $\ominus$ denotes the operator that removes the common edges of the left and the right operand topologies from the left operand topology. For any edge $e_1 \in E(G_1), e_1 \notin E(G_2)$, and node $n_1 \in N(G_1)$ in topology G_1, we have e_1 in the edge set $E(G_1 \ominus G_2)$ and n_1 in the node set $N(G_1 \ominus G_2)$.
- The symbol $w(\cdot)$ denotes the weight function over the signature of network topologies. With this routing algebra, any routing metric is mathematically represented by the weight function $w(\cdot)$.

We further define $w(a) \prec w(b)$ as $w(a) \preceq w(b)$ and $w(a) \neq w(b)$, and let $w(a) \succ w(b)$ mean $w(b) \prec w(a)$. Any BTC algorithm essentially builds a subgraph using the $\oplus$ and/or $\ominus$ operator. With the broadcast routing algebra, BTC algorithms can be expressed in an algebraic manner.

It is important to point out that, our broadcast routing algebra is different from Sobrinho's routing algebra. In Sobrinho's routing algebra, the $\oplus$ operation can only capture simple path concatenation as depicted in Figure 3.2 (I). In our broadcast routing algebra, the $\oplus$ operation is the more general graph union operation as depicted in Figure 3.2 (II), where given two broadcast trees T_1 and T_2 that share a common node l, the $\oplus$ operation joins the two trees and the resulting tree is $T = T_1 \oplus T_2$. Moreover, the broadcast routing algebra also has an unique $\ominus$ operation as shown by the example in Figure 3.2 (III), where $\ominus$ removes from its left operand topology the common edges between its left and right operand topologies.

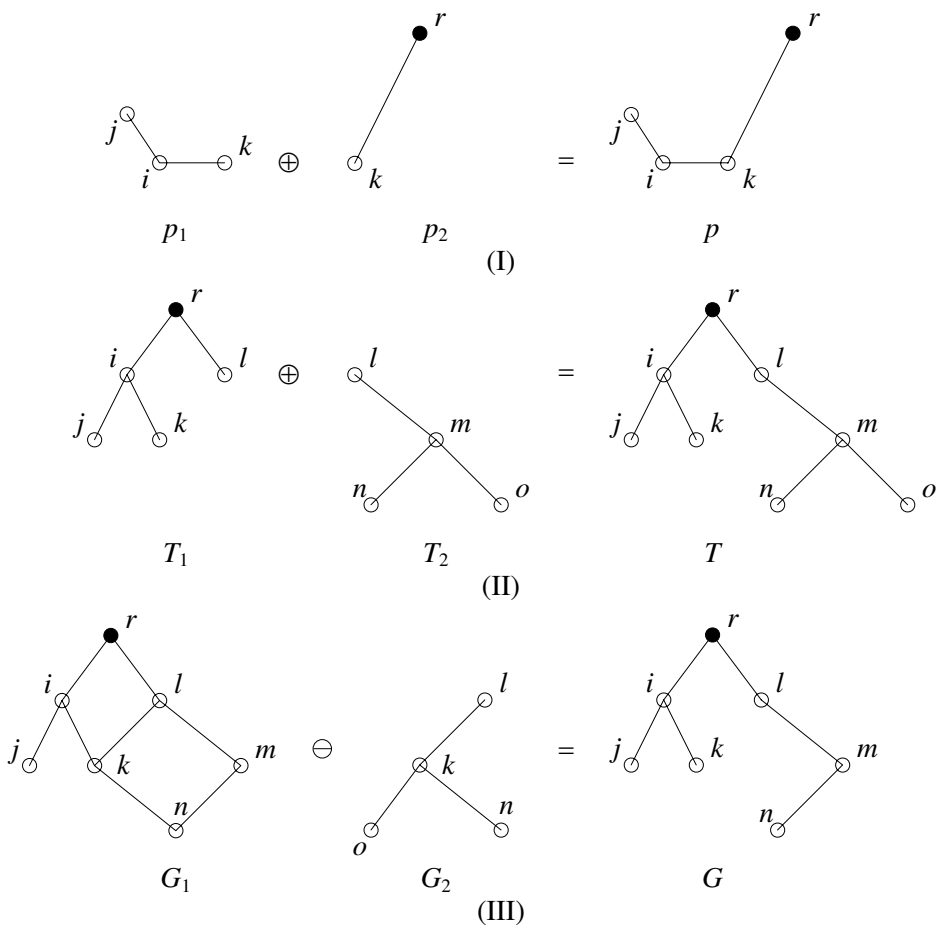


Figure 3.2: The broadcast routing algebra example

3.2.2 Properties

The broadcast routing algebra has the following properties:

Σ is closed under $\oplus$: $a \oplus b \in \Sigma$ for any $a, b \in \Sigma$;

Σ is closed under $\ominus$: $a \ominus b \in \Sigma$ for any $a, b \in \Sigma$;

$\preceq$ is complete: for any $a, b \in \Sigma$, either $w(a) \preceq w(b)$ or $w(b) \preceq w(a)$ (or both);

$\preceq$ is transitive: for $a, b, c \in \Sigma$, if $w(a) \preceq w(b), w(b) \preceq w(c)$, then $w(a) \preceq w(c)$;

$\oplus$ is idempotent: $a \oplus a = a$ for any $a \in \Sigma$;

$\oplus$ is commutative: $a \oplus b = b \oplus a$ for any $a, b \in \Sigma$;

$\oplus$ is associative: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ for any $a, b, c \in \Sigma$;

$\ominus$ is non-commutative: $a \ominus b \neq b \ominus a$ for some $a, b \in \Sigma$;

$\ominus$ is only left-associative but non-associative: $a \ominus b \ominus c = (a \ominus b) \ominus c$ for any $a, b, c \in \Sigma$, but $(a \ominus b) \ominus c \neq a \ominus (b \ominus c)$ for some $a, b, c \in \Sigma$.

3.3 Broadcast Routing in Undirected Network Topologies

In our analysis of BTC algorithms' requirements on routing metrics, there are two different models of the underlying networks: the directed graph model and the undirected graph model. The undirected graph model is appropriate if all the links in the network are bidirectional and the two directions have the same signatures (a.k.a. characteristics). The directed graph model is used to capture more complicated cases where there are asymmetric links. Different BTC algorithms need to be used for undirected and directed network topologies, and these algorithms have different requirements on routing metric design. In this section, we focus on BTC algorithms for undirected network topologies. In the next section, we study BTC algorithms for directed network topologies. Without loss of generality, we formulate the problem of optimal broadcast routing in undirected network topologies as the minimum spanning tree (MST) problem for undirected graphs [31]. In the remainder of this section, we develop and prove the necessary and sufficient metric properties for which Prim's algorithm and the generic edge-adding algorithm guarantee optimality compatibility. Prim's algorithm is based on subtrees, and the generic edge-adding algorithm is based on subforests. Many well-known MST algorithms, such as Kruskal's algorithm [10], Boruvka's algorithm [29], and the GHS algorithm [14], etc., are special cases of the generic edge-adding algorithm.

3.3.1 Prim's Algorithm

Algorithm Overview

Prim's algorithm starts by treating the root node r, which is the broadcast source node, as the initial partial spanning tree T_r. Then it progressively grows the partial spanning tree T_r by adding the best edge from the edge cut of the current T_r, until T_r spans the entire graph. The calculation of the best edge e^* can be generalized based on the binary operation $\oplus$ and the order relation $\preceq$ as follows:

$$e^* = \arg\min_{e \in \partial(T_r)} \{w(T_r \oplus e)\}, \tag{3.3}$$

where $\partial(T_r)$ is the edge cut of T_r [6]. Here, the edge cut $\partial(T_r)$ is the set of edges with one end in $N(T_r)$ and the other end in $N(G) - N(T_r)$. Note that in most textbooks, the original Prim's algorithm is only discussed with linear metrics based on linear link weight aggregation. The generalized form of Prim's algorithm with minimum weight edge in (3.3), however, can cover both linear and non-linear metric designs.

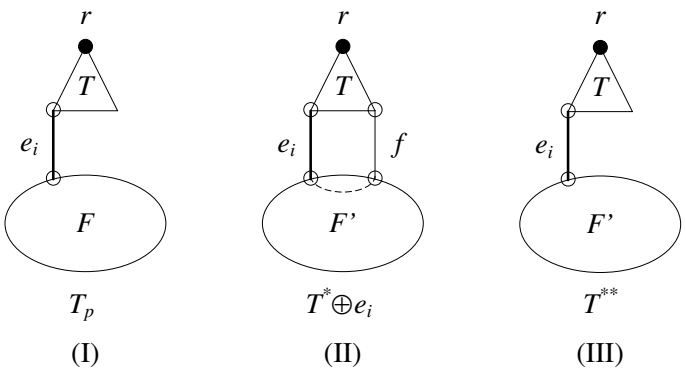


Figure 3.3: Sufficient proof illustration for Prim's algorithm

Metric Properties and Proof

The required routing metric property for the generalized Prim's algorithm can be expressed by the following definition.

Definition 2. Right $\oplus$-isotonicity for trees*: A weight function $w(\cdot)$ of trees is said to be right $\oplus$-isotonic for trees over a graph G if for any tree $T \subset G$ generated by Prim's algorithm, we have*

$$w(T \oplus e) \preceq w(T \oplus e') \Rightarrow w(T \oplus e \oplus F) \preceq w(T \oplus e' \oplus F), \tag{3.4}$$

for any edge $e, e' \in \partial(T)$ and forest F satisfying $E(F) \cap (E(T) \cup \partial(T)) = \emptyset$ such that both $T \oplus e \oplus F$ and $T \oplus e' \oplus F$ are still trees.

Essentially, the right $\oplus$-isotonicity for trees specifies that the preference order $\preceq$ between the two trees that are expanded from a tree generated by Prim's algorithm remains unchanged, when they are further expanded into larger trees through the same set of additional edges.

Theorem 1. *Given any connected and undirected network topology G whose root node is r, Prim's algorithm produces the MST, if and only if the broadcast routing metric $w(\cdot)$ is right $\oplus$-isotonic for trees.*

Proof. Sufficient condition: Let T_p be the Prim tree that is generated by Prim's algorithm. Denote $e_1, \ \ldots, \ e_{|N(G)|-1}$ as the order in which Prim's algorithm selects edges, where $|N(G)|$ is the total number of nodes in graph G. We next prove that if the broadcast routing metric satisfies the property in (3.4), Prim's algorithm produces a MST. This can be proved by contradiction.

Suppose T_p is not a MST. Following the order $e_1, \ \ldots, \ e_{|N(G)|-1}$, compare each edge in $E(T_p)$ with edges of a MST. Denote the MST that shares the largest number of consecutive common edges with T_p as T^* and the first edge that T_p differs with T^* as e_i. By Prim's

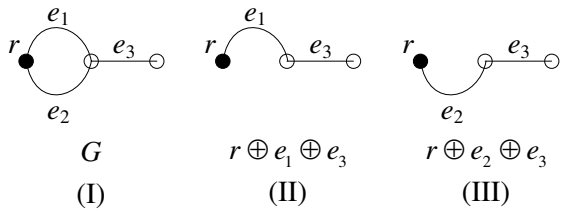


Figure 3.4: The necessary condition proof example for Prim's algorithm

algorithm, the edge set $\{e_1,\ e_2,\ \ldots,\ e_{i-1}\}$ is a tree. Denote this tree as T as shown in Figure 3.3 (I). Tree T is a subgraph for both T_p and T^*. Consider adding edge e_i to T^* as shown in Figure 3.3 (II). Then, there must exist a cycle containing e_i, and within the cycle there exists an edge $f \in \partial(T),\ f \in T^*,\ f \neq e_i$. By Prim's algorithm, it follows $w(T \oplus e_i) \preceq w(T \oplus f)$. Substituting f with e_i, the resulting subgraph $T^{**} = T^* \oplus e_i \ominus f$ in Figure 3.3 (III) is still a spanning tree. Since $w(T \oplus e_i) \preceq w(T \oplus f)$, by the property in (3.4), it follows that $w(T^{**}) = T^* \oplus e_i \ominus f = w(T \oplus e_i \oplus (T^* \ominus T \ominus f)) \preceq w(T \oplus f \oplus (T^* \ominus T \ominus f)) = w(T^* \oplus f \ominus f) = w(T^*)$. Since T^* is a MST, T^{**} is also a MST. The fact that T^{**} has one more common edge e_i with T_p than T^* contradicts the definition of T^*. Hence, T_p is also a MST.

Necessary condition: We need to prove that if Prim's algorithm produces the MST on any network topology, then the metric satisfies the property in (3.4). This can be proved by showing that its contrapositive is correct, i.e., if a metric does not satisfy the property in (3.4), then there is at least one network topology for which the algorithm does not guarantee optimality.

Consider the network topology G in Figure 3.4 (I). Suppose $w(r \oplus e_1) \preceq w(r \oplus e_2)$ and the metric does not satisfy the property in (3.4). Note that $r \oplus e_1 \oplus e_3$ and $r \oplus e_2 \oplus e_3$ are trees, as shown in Figure 3.4 (II) and (III), respectively. Since for the given metric the property in (3.4) does not hold, it is possible that $w(r \oplus e_1 \oplus e_3) \succ w(r \oplus e_2 \oplus e_3)$. Meanwhile, the tree produced by Prim's algorithm is $T_p = r \oplus e_1 \oplus e_3$ which is not the MST. Hence, for the given metric, Prim's algorithm does not produce the MST for this particular network topology. Therefore, for Prim's algorithm to guarantee optimality on any network topology, the metric must satisfy the property in (3.4). □

Note that the property in (3.4) is different from the consistency property in [20]. The consistency property in [20] is only the sufficient but not necessary condition to guarantee that Prim's algorithm produces the MST, while the property in (3.4) is both sufficient and necessary. This can be shown by the tree depth example in Section 3.6.1.

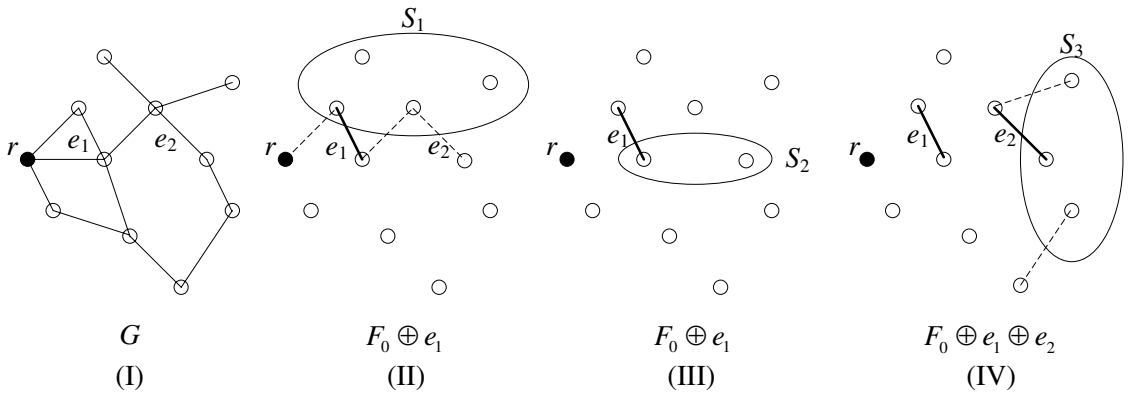


Figure 3.5: The generic edge-adding algorithm example

3.3.2 Generic Edge-adding Algorithm

There is a generic edge-adding algorithm on undirected graphs [10, 29]. It covers typical forest-based MST algorithms, e.g., Kruskal's algorithm, Boruvka's algorithm, and the GHS algorithm, etc.. These algorithms only differ in their edge-adding orders.

Algorithm Overview

The generic edge-adding algorithm starts by initializing forest F as $F = F_0 = (N(G), \emptyset)$, where $\emptyset$ is the empty set. In each of the following steps, it picks one edge e^* to add to F until F is a spanning tree of G. The edge e^* is determined by first finding a node set S such that F does not have any edge that belongs to the edge cut of S. Here, the edge cut of S is the set of edges in G with one end in S and the other end in $N(G) - S$. The edge e^* is then chosen from the edge cut of S [6], denoted as $\partial(S)$, as follows:

$$e^* = \arg\min_{e \in \partial(S)} \{w(F \oplus e)\}, \tag{3.5}$$

where $\partial(S) \cap E(F) = \emptyset$ and $S \subset N(G)$.

The edge-adding process can be illustrated by an example. Consider the undirected graph G in Figure 3.5 (I) and the initial forest $F_0 = (N(G), \emptyset)$. In the first step shown in Figure 3.5 (II), e_1 is chosen from $\partial(S_1)$, i.e., the edge cut of the node set S_1. This results in a forest $F = F_0 \oplus e_1$. Note that in the next step, the node set S_2 shown in Figure 3.5 (III) cannot be chosen since $\partial(S_2) \cap E(F_0 \oplus e_1) = \{e_1\} \neq \emptyset$. Instead, one can choose the node set S_3 and pick edge e_2 within $\partial(S_3)$ as shown in Figure 3.5 (IV). This process continues until the resulting forest is a spanning tree of G.

Metric Properties and Proof

The required routing metric property for the generic edge-adding algorithm can be expressed by the following definition.

Definition 3. Right $\oplus$-isotonicity for forests*: A weight function $w(\cdot)$ of forests is said to be right $\oplus$-isotonic for forests over a graph G if for any forest $F \subset G$ generated by the generic edge-adding algorithm, given any node set $S \subset N(G)$ satisfying $\partial(S) \cap E(F) = \emptyset$, we have*

$$w(F \oplus e) \preceq w(F \oplus e') \Rightarrow w(F \oplus e \oplus F') \preceq w(F \oplus e' \oplus F'), \quad (3.6)$$

for any edge e, $e' \in \partial(S)$ and forest $F' \subset G$ such that $F \oplus e \oplus F'$ and $F \oplus e' \oplus F'$ are forests.

Essentially, the right $\oplus$-isotonicity for forests specifies that the preference order $\preceq$ between the two forests that are expanded from a forest generated by the generic edge-adding algorithm remains unchanged, when they are further expanded into larger forests through adding a common set of edges.

Theorem 2. *Given any connected and undirected network topology G whose root node is r, the generic edge-adding algorithm produces the MST, if and only if the broadcast routing metric $w(\cdot)$ is right $\oplus$-isotonic for forests.*

Proof. Sufficient condition: Let T_a be the spanning tree generated by the generic edge-adding algorithm. Denote $e_1, e_2, ..., e_{|N(G)|-1}$ as the order of adding edges, where $|N(G)|$ is the total number of nodes in the graph G. Suppose T_a is not a MST. Denote T^* as the MST that shares the largest number of consecutive common edges with the edge sequence $e_1, e_2, ..., e_{|N(G)|-1}$ of T_a. Denote the first edge in the sequence that differs from T^* as e_i. Let F be the forest consisting of edges $e_1, e_2, ..., e_{i-1}$, i.e., $F = \{e_1, e_2, ..., e_{i-1}\}$. Consider adding edge e_i into the spanning tree T^*. Then, there must exist a cycle containing e_i. By the generic edge-adding algorithm and the tree property of T^*, there must exist an edge $e_i' \in E(T^*)$ in the cycle, such that $w(F \oplus e_i) \preceq w(F \oplus e_i'), e_i, e_i' \in \partial(S), \partial(S) \cap E(F) = \emptyset, S \subset N(G)$, where S is the selected node set before adding edge e_i when running the algorithm. Deleting e_i' from $T^* \oplus e_i$ generates another spanning tree. By the property in (3.6), it follows that $w(T^* \oplus e_i \ominus e_i') = w(F \oplus e_i \oplus (T^* \ominus F \ominus e_i')) \preceq w(F \oplus e_i' \oplus (T^* \ominus F \ominus e_i')) = w(T^* \oplus e_i' \ominus e_i') = w(T^*)$. Since T^* is a MST, $T^* \oplus e_i \ominus e_i'$ is also a MST and it contains edge e_i. Since $T^* \oplus e_i \ominus e_i'$ contains more consecutive common edges with T_a than T^*, this contradicts the definition of T^*. Hence, T_a is also a MST.

Necessary condition: Similar to the necessary condition proof of Theorem 1, we prove the necessity by showing that its contrapositive is correct, i.e., if there is at least one metric that does not satisfy the property in (3.6), there is at least one network topology for which the spanning tree generated by the generic edge-adding algorithm is not a MST.

Suppose, for a network topology, T_a is the spanning tree generated by the generic edge-adding algorithm, as shown in Figure 3.6 (I). By the generic edge-adding algorithm, it follows that

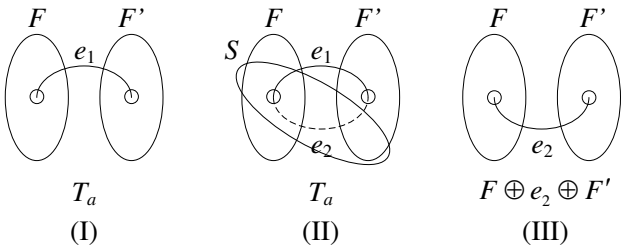


Figure 3.6: The necessary condition proof example for the generic edge-adding algorithm

$w(F \oplus e_1) \preceq w(F \oplus e_2), e_1, e_2 \in \partial(S), \partial(S) \cap E(F) = \emptyset$, where F is the partially generated forest before adding edge e_1 and $S \subset N(G)$ is the selected node set before adding edge e_1. Let $F' = T_a \ominus e_1 \ominus F$ as shown in Figure 3.6 (II). Since for the given metric the property in (3.6) is not guaranteed, it is possible that $w(T_a) = w(F \oplus e_1 \oplus F') \succ w(F \oplus e_2 \oplus F')$ as shown in Figure 3.6 (III). That is T_a cannot be the MST of the original network topology. Hence, if T_a is a MST, then the metric satisfies the property in (3.6). □

Note that the above proof can be applied to any specific order of adding edges. Therefore, Theorem 2 also holds for typical edge-adding algorithms, e.g., Kruskal's algorithm, Boruvka's algorithm and the GHS algorithm.

3.4 Broadcast Routing in Directed Network Topologies

In a directed network topology, the link from node n_1 to node n_2 may have different characteristics compared to the link from node n_2 to node n_1. The problem of optimal broadcast routing in directed network topologies can be formulated as the minimum weight spanning r-arborescence problem for directed graphs. We first define the r-arborescence. For more details about arborescence, readers are referred to [29].

Definition 4. *A r-arborescence is a directed graph in which node r is called the root node, and there exists exactly only one directed path from the root node r to any other non-root node.*

Next, we define the spanning r-arborescence and the partition subgraph of a directed graph as follows.

Definition 5. *Given a connected and directed graph D whose root node is r, a spanning r-arborescence is a directed subgraph of D such that there exists exactly one directed path from the root node r to any other non-root node.*

Definition 6. *Given a directed graph D whose root node is r, a partition subgraph is a subgraph D' of D such that the indegree of the root node r is equal to 0 and the indegrees of other nodes are less than or equal to 1, i.e., $d^-(r) = 0, d^-(n) \leq 1, \forall\ n \in N(D'), n \neq r$.*

The most well-known algorithm for solving the minimum weight spanning r-arborescence problem is Edmonds' algorithm [15]. In this section, for directed network topologies, the necessary and sufficient metric property of the generalized Edmonds' algorithm is developed and proved.

3.4.1 Edmonds' Algorithm

Overview of Edmonds' Algorithm

The core idea of Edmonds' algorithm is summarized as follows. Given a directed graph D, whose root node is r, remove all the inbound arcs of root node r, and denote the resultant subgraph as D_0. Edmonds' algorithm proceeds to build the minimum r-arborescence spanning D_0 following a three-phase procedure.

Initialization: Send D_0 to Phase I as Phase I's input.

Phase I: Given the input graph D_i for the i_{th} iteration of Phase I, create a directed graph T_i that includes all nodes in $N(D_i)$ and no arcs between nodes. Each node in T_i, hence, is a separate arborescence. For each non-root node n whose indegree is 0, a new arc $a^* \in E(D_0)$ is selected to be included in T_i. The new arc a^* is selected as follows:

$$a^* = \arg \min_{a \in \mathcal{A}(n)} \{w(a)\}, \tag{3.7}$$

where $\mathcal{A}(n) = \{a | a \in A^-(n), n \in N(D_i), n \neq r,\ d^-(n) = 0\}$, $A^-(n)$ is the set of arcs that terminate at node n, and $d^-(n)$ is the indegree of node n in T_i.

The above arc adding process ends when $d^-(n) = 1$ for any $n \in N(T_i), n \neq r$, i.e, the indegree of any non-root node in T_i is 1. If there is no circuit in T_i after the arc adding process, then go to phase III with T_i as its input. If there are circuits in T_i, then go to phase II and let T_i and D_i be its input.

Phase II: Given the input graph T_i and D_i, pick a circuit C_i in T_i to eliminate as follows. Find a pair of arcs $(a^*_{c'c}, a^*_{nc})$ that can be used to break C_i by replacing $a^*_{c'c}$ with a^*_{nc}, where $a^*_{c'c} \in C_i$, $a^*_{nc} \in D_i$ and $a^*_{nc} \notin T_i$. Both $a^*_{c'c}$ and a^*_{nc} are inbound arcs to node $c \in N(C_i)$. Arcs $a^*_{c'c}$ and a^*_{nc} are selected based on the following equation:

$$(a^*_{c'c}, a^*_{nc}) = \arg \min_{\substack{a_{nc} \notin T_i \\ a_{c'c} \in C_i}} \{w(C_i \ominus a_{c'c} \oplus a_{nc})\}, \tag{3.8}$$

where $a_{c'c}$ is any arc in C_i, $C_i \ominus a_{c'c}$ is the path generated by deleting arc $a_{c'c}$ from C_i, and a_{nc} is an inbound arc to node c. Denote P^*_i as the path generated by breaking the circuit C_i, i.e., $P^*_i = C_i \ominus a^*_{c'c} \oplus a^*_{nc}$.

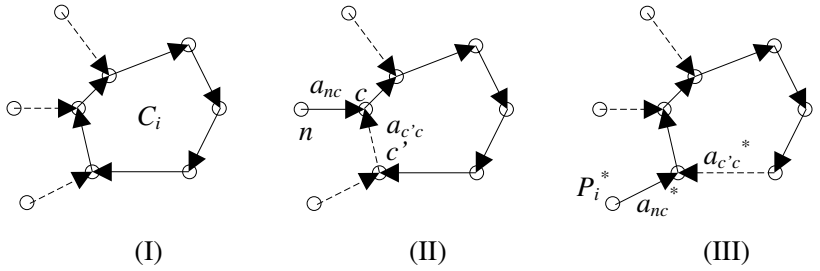


Figure 3.7: Illustration for Edmonds' algorithm

This circuit breaking operation can be illustrated by an example. Consider the circuit C_i in Figure 3.7 (I), where the circuit is the solid part of the graph. The circuit C_i is broken by replacing $a_{c'c}$ by a_{nc} as shown in Figure 3.7 (II), where the resultant arborescence is the solid part of the graph. The optimal circuit-breaking scheme is shown in Figure 3.7 (III), where the resultant arborescence P_i^* is the solid-line part of the graph.

With $(a_{c'c}^*, a_{nc}^*)$ identified, shrink the circuit C_i in D_i to a pseudo-node n_i. Replace $N(C_i)$ in D_i by n_i. Consider the terminating nodes of the outbound arcs going out of nodes in C_i. Their signatures remain unchanged. Set the inbound arc of n_i as a_{nc}^* and modify its signature to be the signature of P_i^*, i.e., the signature of the path used to replace the original circuit. Denote this newly generated graph through the shrinking operation as D_{i+1}. Go to the beginning of Phase I and let D_{i+1} be Phase I's input.

Phase III: Given the input T_l after l iterations of phase I and II, expand the pseudo-node n_i to P_i in the reverse order $(i = l, l-1, \ldots, 1)$ of their generation sequence in phase II. The resulting subgraph T_e is the minimum weight spanning r-arborescence of D_0.

Note that in most textbooks, Edmonds' algorithm is only discussed with linear metric based on linear link weight aggregation. The algorithm discussed in this section is extended to cover both linear and nonlinear link metric aggregation.

Metric Properties and Proof

The required routing metric property for Edmonds' algorithm can be expressed by the following definition.

Definition 7. Right $\oplus$-isotonicity for partition subgraphs*: A weight function $w(\cdot)$ of partition subgraphs is said to be right $\oplus$-isotonic for partition subgraphs over a directed graph D if for any partition subgraphs D_1, D_2 of D, we have*

$$w(D_1) \preceq w(D_2) \Rightarrow w(D_1 \oplus D_3) \preceq w(D_2 \oplus D_3), \tag{3.9}$$

where D_3 is also a partition subgraph of D such that $D_1 \oplus D_3, D_2 \oplus D_3$ are still partition subgraphs of D.

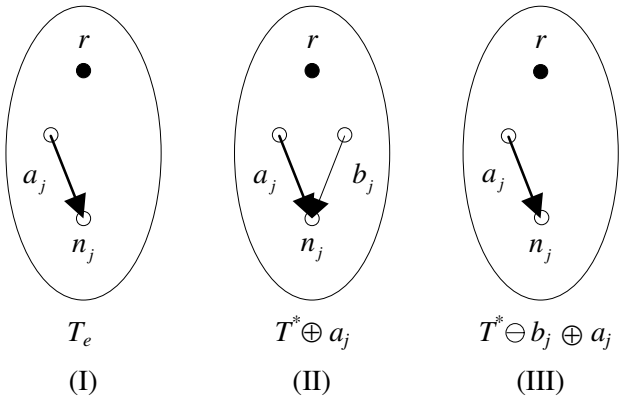


Figure 3.8: Sufficient proof illustration for Edmonds' algorithm

Essentially, the right $\oplus$-isotonicity for partition subgraphs specifies that the preference order $\preceq$ between two partition subgraphs D_1 and D_2 remains unchanged, when we expand them into larger partition subgraphs through adding a common set of arcs.

Theorem 3. *Given any connected and directed network topology D whose root node is r, Edmonds' algorithm produces the minimum weight r-arborescence spanning D, if and only if the broadcast routing metric is right $\oplus$-isotonic for partition subgraphs.*

Proof. Sufficient condition: Consider a connected and directed network topology D whose root node is r. We are going to prove that if the broadcast routing metric satisfies the property in (3.9), Edmonds' algorithm produces the minimum weight r-arborescence spanning D.

First, we prove that if there is no circuit generated after the arc adding procedure in phase I, phase I produces the minimum weight spanning r-arborescence T_i of its input graph D_i. Let T^* be a minimum weight spanning r-arborescence and T_e be the arborescence generated by phase I of Edmonds' algorithm. Assume $T_e \neq T^*$. There must exist some nodes $n_j, j = 1, \ldots, m$ whose inbound arc a_j in T_e is different from the inbound arc b_j in T^* as shown in Figure 3.8. By Edmonds' algorithm, $w(a_j) \preceq w(b_j)$. Starting from T^*, replace b_1 in T^* by the corresponding a_1 in T_e. Denote the generated subgraph as T_1^*. By the property in (3.9), it follows that $w(T_1^*) = w(T^* \ominus b_1 \oplus a_1) \preceq w(T^* \ominus b_1 \oplus b_1) = w(T^*)$. Similarly, replacing b_2 by a_2 in T_1^*, we get T_2^* that satisfies $w(T_2^*) \preceq w(T_1^*)$. By continuing this arc replacing process, we get a series of graphs $T_1^*, T_2^*, \cdots T_m^* = T_e$ that satisfy $w(T_{j+1}^*) \preceq w(T_j^*), j = 1, \ldots, m-1$. Hence, it follows that $w(T_e) \preceq w(T^*)$. Since T^* is a minimum weight spanning r-arborescence, $w(T_e) = w(T^*)$. T_e is also a minimum weight spanning r-arborescence.

Denote T_i^e as the subgraph generated after expanding the pseudo node n_i to P_i^* in phase III. We next prove that T_i^e is the minimum weight spanning r-arborescence of D_i through induction hypothesis. First, for the base case, note that T_l is the minimum weight spanning

r-arborescence of D_l and our hypothesis is satisfied. Then, assume that T^e_{i+1} is the minimum weight spanning r-arborescence of D_{i+1}. We next show that T^e_i is the minimum weight spanning r-arborescence of D_i.

Let T^0_i be an arbitrary spanning arborescence of D_i. By the fact that T^0_i is a spanning arborescence, there must exist an arc a_{nc} that emanates from some node $n \in N(D_i) - N(C_i)$ and terminates at some node $c \in N(C_i)$. Let $P^n_i = C_i \ominus a_{c'c} \oplus a_{nc}$ be the subgraph generated by replacing arc $a_{c'c}$ by arc a_{nc}, where $a_{c'c} \in C_i$. Denote the subgraph consisting of the inbound arcs of $N(C_i)$ in T^0_i as F_i. If $P^n_i = F_i$, then $w(P^n_i) = w(F_i)$. If $P^n_i \neq F_i$, there must exist some nodes in $N(C_i)$ whose inbound arc a_j in F_i is different from the inbound arc b_j in P^n_i, $j = 1, 2, ..., t$. By Edmonds' algorithm, $w(b_j) \preceq w(a_j)$. Starting from F_i, replace a_1 in F_i by the corresponding b_1 in P^n_i. Let the generated subgraph be F^1_i. By the property in (3.9), it follows that $w(F^1_i) = w(F_i \ominus a_1 \oplus b_1) \preceq w(F_i \ominus a_1 \oplus a_1) = w(F_i)$. Similarly, replacing a_2 by b_2 in F^1_i, we get F^2_i that satisfies $w(F^2_i) \preceq w(F^1_i)$. By continuing this arc replacing process, we get a series of graphs $F^1_i, F^2_i, \cdots F^t_i = P^n_i$ that satisfy $w(F^{j+1}_i) \preceq w(F^j_i), j = 1, \ldots, t-1$. Hence, it follows that $w(P^n_i) \preceq w(F_i)$.

Let $T^1_i = T^0_i \ominus F_i \oplus P^n_i$ be the subgraph generated by replacing F_i in T^0_i by P^n_i. Since $w(P^n_i) \preceq w(F_i)$, it follows that

$$w(T^1_i) = w(T^0_i \ominus F_i \oplus P^n_i) \preceq w(T^0_i \ominus F_i \oplus F_i) = w(T^0_i). \tag{3.10}$$

Shrink P^n_i in T^1_i into n_i to get a new graph T^2_i. By the fact that T^0_i is a r-arborescence of D_i, T^2_i is still a r-arborescence of D_{i+1}. Since T^e_{i+1} is the minimum weight spanning r-arborescence of D_{i+1}, we have

$$w(T^e_{i+1}) \preceq w(T^2_i). \tag{3.11}$$

By Edmonds' algorithm, the shrink and expansion operations do not change the weight of the signature of a graph, i.e., $w(T^e_i) = w(T^e_{i+1})$ and $w(T^2_i) = w(T^1_i)$. Hence, from (3.10) and (3.11), we get

$$w(T^e_i) \preceq w(T^0_i). \tag{3.12}$$

Since T^0_i is an arbitrary spanning arborescence of D_i, T^e_i is the minimum weight spanning r-arborescence of D_i. Hence, by induction hypothesis, T^e_i is the minimum weight spanning r-arborescence of D_i. Hence, after expanding all the l circuits, Edmonds's algorithm produces the minimum weight r-arborescence of D_0.

Necessary condition: Similar to the necessary condition proof of Theorem 1, we prove the necessity by showing its contrapositive is correct, i.e., if a broadcast routing metric does not satisfy the properties in (3.9), then there exists at least one network topology where Edmonds' algorithm does not guarantee optimality.

Consider the directed network topology D in Figure 3.9 (I), where node r is the root node. Suppose $w(a_1) \preceq w(a_2)$. Assume the metric does not satisfy the property in (3.9). Since $w(a_1) \preceq w(a_2)$, it is possible that $w(a_1 \oplus a_3) \succ w(a_2 \oplus a_3)$ as shown in Figure 3.9 (II) and (III). Meanwhile, the output of Edmonds' algorithm is $T^2_e = a_1 \oplus a_3$, which is not optimal.

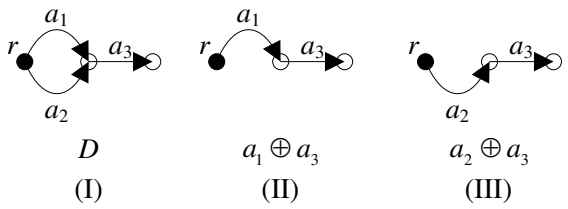


Figure 3.9: The necessary condition proof example for Edmonds' algorithm

Hence, Edmonds' algorithm fails to produce the minimum weight spanning r-arborescence. Therefore, the property in (3.9) is the necessary metric property for the statement that Edmonds' algorithm guarantees optimality on any network topology. □

3.5 Distributed Broadcast Routing

Theorems 1, 2, and 3 provide the necessary and sufficient metric properties for which Prim's algorithm, the generic edge-adding algorithm, and Edmonds' algorithm guarantee optimality, respectively. It is important to note that although all these BTC algorithms require knowledge of global topology, optimal broadcast routing protocols based on these algorithms do not need to be centralized. In this section, we discuss the distributed implementation of broadcast routing, and how its complexity is related to the underlying uniqueness problem of broadcast trees. Here, the uniqueness problem refers to whether the broadcast trees for different source nodes are the same. We find that this uniqueness problem is related to a property called root independence.

3.5.1 A Schematic Protocol

In the following, we outline a simple example of distributed broadcast routing protocols. Similar to link-state routing protocols such as OSPF [38], every node periodically advertises its local connectivity to the entire network. In this way, every node can learn the global topology. For each possible broadcast source node, a node runs one of the BTC algorithms to compute the broadcast tree rooted at the source node. In the broadcast tree, the node identifies its children and stores them as the outgoing links in its routing table. When each non-source node receives a broadcast routing packet, it checks the source address of the packet and forwards the packet to the outgoing links according to its routing table. Since each node has the same view of the topology and runs the same BTC algorithm, the forwarding information stored in each node's routing table is consistent with the same unique broadcast tree and hence routing loops are avoided.

In general, the OBT for one root node is different from the OBT for another root node so that per source computation for its broadcast tree and per source routing entry is needed at each

router of the network to support broadcast routing. The computation overhead and routing table size at the routers can be huge, if there are many broadcast source nodes. Therefore, we are interested in developing metric properties for which the computed broadcast trees for different roots have the same topology. That is the MST or minimum weight spanning arborescence rooted at one node is the same as the MST or minimum weight spanning arborescence rooted at another node. When this is true, each node, including the source node and non-source node, only computes and maintains one broadcast tree in their routing tables for all source nodes. This saves a lot of computation time and also reduces the routing table size, when the network scale is large and there are multiple broadcast source nodes.

3.5.2 Uniqueness Properties for Optimal Broadcast Trees

In this section, we show that to ensure the same broadcast tree for different sources, the routing metric must have a property called root independence.

Definition 8. *A metric $w(\cdot)$ is root independent, if $w(F_1) = w(F_2)$, where F_1 and F_2 are subgraphs that have different root nodes and satisfy $N(F_1) = N(F_2), E(F_1) = E(F_2)$.*

A simple example of a root independent metric is the summation of all the link weights in an undirected tree. Given an undirected tree, the metric weight remains unchanged regardless of the root node. An example of a root dependent metric is the tree depth metric that measures path length of the longest shortest path from the root node to any leaf node in a tree. Given a tree, in general, the tree depth changes when the root node changes.

Theorem 4. *Given any connected and undirected network topology G, the MST is the same for different roots, if and only if the metric definition is independent of root node.*

Proof. Sufficient condition: Let T_1^* be one MST of the connected and undirected network topology G whose root node is r_1. It follows that $w(T_1^*) \preceq w(T_1)$, where T_1 is any spanning tree rooted at r_1. Let T_2^* be the spanning tree generated by changing the root node of T_1^* from r_1 to r_2. Note that any spanning tree rooted at r_1 can also be a spanning tree rooted at r_2, and vice versa. Since the metric definition is independent of root node, for any r_1-rooted spanning tree T_1 of G, we have a r_2-rooted spanning tree T_2 of G_2 such that $N(T_1) = N(T_2), E(T_1) = E(T_2), w(T_1) = w(T_2)$. It follows that $w(T_2^*) = w(T_1^*) \preceq w(T_1) = w(T_2)$, i.e., T_2^* is a MST of G. Therefore, the MST is independent of the root node.

Necessary condition: Given any connected and undirected network topology with a root node, we need to prove that if the MST is the same for all sources then the metric definition is independent of the root node. This can be proved by showing that its contrapositive is correct, i.e., for a metric whose weight computation is related to the identity of the root node, there exists at least one topology, where the MST is dependent on the root node.

Suppose, for a network topology G whose root node is r_1, T_1^* is a MST rooted at r_1. It follows that $w(T_1^*) \preceq w(T_1)$, where T_1 is any spanning tree rooted at r_1. Change the root

Table 3.2: Summary of the case study

Metrics	Prim's	Edge-adding	Edmonds'	
Minimum tree depth broadcast routing	Y	N	N	
Most reliable widest bandwidth broadcast routing	N	N	N	
Minimum energy broadcast routing	N	N	N	
Lexicographically optimal broadcast routing	Y	Y	Y	
Maximum network lifetime broadcast routing	NA[1]	NA	Y	unequal residual energy
	Y	Y	Y	equal residual energy

[1]The modified Prim's algorithm (termed as DMST) in [30] is compatible with the network lifetime metric.

node r_1 of T_1^* and T_1 to r_2, and let the generated spanning trees as T_2^* and T_2, respectively. Since the metric definition is dependent on the root node, it is possible $w(T_1^*) \neq w(T_2^*)$ and $w(T_1) \neq w(T_2)$. Hence, it is possible that $w(T_2^*) \succ w(T_2)$. Therefore, the MST is dependent on the root node. □

Theorem 5. *Given any connected and directed network topology D whose root node is r, the minimum weight spanning arborescence is dependent on the root node r.*

Proof. In directed network topologies, the definition of minimum weight spanning arborescence is dependent on the root node r. For two nodes n_1 and n_2, the weight of the link from n_1 to n_2 is generally different from the weight of the link from n_2 to n_1. Moreover, the link between n_1 and n_2 may be unidirectional rather than bidirectional. For some node, there may not exist a spanning arborescence rooted at this node. Therefore, in the general case, the root-independence property does not hold for the minimum weight spanning arborescence. □

3.6 Case Study Based on Compatibility Analysis

In this section, we illustrate by examples about how to apply our analytical results in Sections 3.3 and 3.4 to judge the optimality compatibility between broadcast routing metrics and BTC algorithms. The case study results are summarized in Table 3.2, where Y means the algorithm is compatible with the metric, N means the algorithm is not compatible with the metric, and NA means the algorithm is not applicable with the metric.

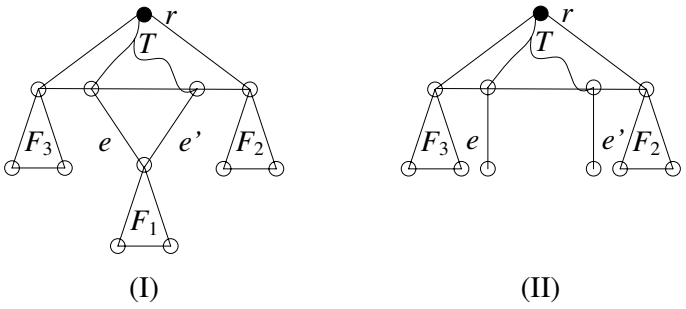


Figure 3.10: Two cases for Lemma 1 proof

3.6.1 Case 1: Minimum Tree depth Broadcast Routing

Consider the case where the objective is to minimize the message delivery delay in a possibly partitioned network by minimizing the depth of broadcast trees in each of the topology components. The definition of the routing metric for such a purpose is

$$w(F) = \max_{T \subset F} depth(T, i), \tag{3.13}$$

where F is a graph composed of multiple components, T is a component of F, and $depth(T, i)$ is the depth of component T with broadcast source node i. The depth of a component T is the maximum number of edges along the shortest simple path from the root node to any other nodes in T. Mathematically, it means that

$$depth(T, i) = \max_{j \in N(T), j \neq i} SP(i, j), \tag{3.14}$$

where $SP(i, j)$ is the length of the shortest path from node i to node j.

Compatibility with Prim's Algorithm

To facilitate the case study, we first prove the following lemma.

Lemma 1. *The metric in* (3.13) *satisfies the right* $\oplus$*-isotonicity property for trees in* (3.4)*.*

Proof. Let T be the tree generated by Prim's algorithm, and $T_1 = T \oplus e, T_2 = T \oplus e'$. Note that there is only one path from a non-root node to the root node in a tree. Suppose the longest path to the root node in T is $i, i \geq 0$ hops long. Since T is generated by Prim's algorithm, new edges e and e' can only be appended to nodes that are i hops to the root node in T, or be appended to nodes that are $i - 1$ hops to the root node in T if there exist nodes which are $i - 1$ hops to the root node and e or e' can be appended to. The lemma can be proved in two parts.

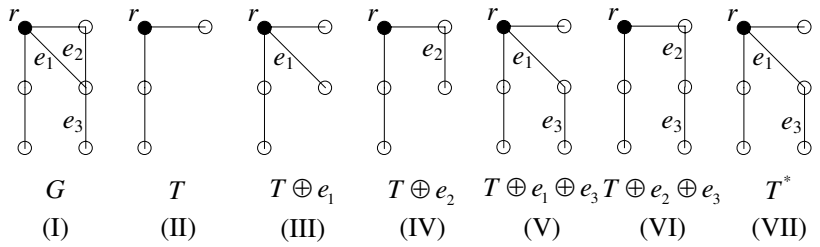


Figure 3.11: The counterexample for the consistency property

First, we consider the case $w(T \oplus e) = w(T \oplus e')$. In this case, either both e and e' are appended to nodes that are i hops to the root node in T, or both e and e' are appended to nodes that are $i-1$ hops to the root node in T. For both subcases, by the fact that $w(T \oplus e) = w(T \oplus e')$, we have $w(T \oplus e \oplus F) = w(T \oplus e' \oplus F)$.

Second, we consider the case $w(T \oplus e) \prec w(T \oplus e')$. In this case, e must be appended to a node that is $i-1$ hops to the root node in T and e' must be appended to a node that is i hops to the root node in T. There are two cases to consider. In the first case, e' is on the longest path to the root node in $T \oplus e' \oplus F$, and we further have two subcases to consider. In the first subcase, edges e and e' share the same end node in the node set $N(G) - N(T)$ as shown in Figure 3.10 (I), where $F = \{F_1 \oplus F_2 \oplus F_3\}$. For this subcase, subtrees of F can be appended to the shared end node of e and e', nodes that are i hops to the root node in T, or nodes that are $i-1$ hops to the root node in T. By the fact that $w(T \oplus e) \prec w(T \oplus e')$, we have $w(T \oplus e \oplus F) \preceq w(T \oplus e' \oplus F)$. In the second subcase, edges e and e' have different end nodes in the node set $N(G) - N(T)$ as shown in Figure 3.10 (II), where $F = \{F_2 \oplus F_3\}$. In this subcase, during the $T_1 \oplus F$ and $T_2 \oplus F$ operation, there exist no subtrees in F which can be appended to either e or e'. Otherwise, $T \oplus e \oplus F$ and $T \oplus e' \oplus F$ are forests rather than trees. It follows that subtrees of F can only be appended to nodes that are i hops to the root node in T or nodes that are $i-1$ hops to the root node in T. By the fact that $w(T \oplus e) \prec w(T \oplus e')$, we still have $w(T \oplus e \oplus F) \preceq w(T \oplus e' \oplus F)$. In the second case, e' is not on the longest path to the root node in $T \oplus e' \oplus F$, and we have $w(T \oplus e \oplus F) = w(T \oplus e' \oplus F)$. In summary, for the case $w(T \oplus e) \prec w(T \oplus e')$, we have $w(T \oplus e \oplus F) \preceq w(T \oplus e' \oplus F)$. □

By Lemma 1 and Theorem 1, Prim's algorithm can find the minimum depth broadcast tree.

Next, we show that the consistency property in [20] is only the sufficient but not necessary property for Prim's algorithm to find the MST. Consider the tree depth metric in (3.13), and the example topology in Figure 3.11 (I). Notice that $w(T \oplus e_1) = w(T \oplus e_2) = 2$ but $w(T \oplus e_1 \oplus e_3) = 2 \prec w(T \oplus e_2 \oplus e_3) = 3$. It follows that the metric depth metric does not satisfy the consistency property in [20]. By part 2 of Theorem 2 in [20], Prim's algorithm cannot find the MST. However, for this particular topology, Prim's algorithm can find the MST T^* as shown in Figure 3.11 (VII). Further, by Lemma 1, the tree depth metric satisfy our property in (3.4); therefore, by Theorem 1, Prim's algorithm can find the minimum depth spanning tree.

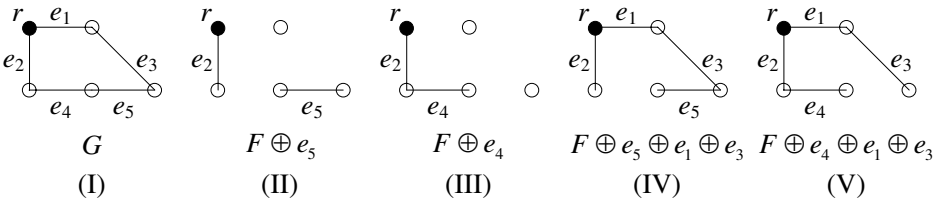


Figure 3.12: The generic edge-adding algorithm example for the minimum tree depth routing

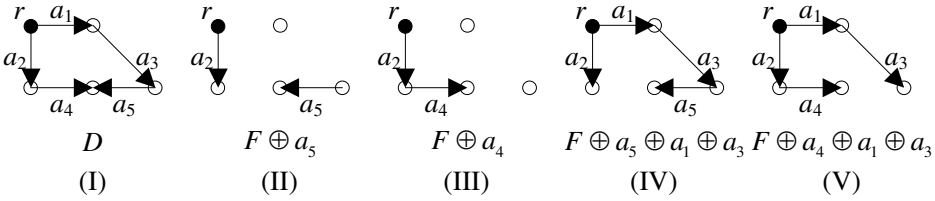


Figure 3.13: The Edmonds algorithm example for the minimum tree depth routing

Compatibility with the Generic Edge-adding Algorithm

The depth metric definition in (3.13) does not satisfy the property in (3.6). Consider the network topology G as shown in Figure 3.12 (I). Let $F = (N(G), \{e_2\})$. Note that $w(F \oplus e_5) = 1 \prec w(F \oplus e_4) = 2$ as shown in Figure 3.12 (II) and (III), but $w(F \oplus e_5 \oplus e_1 \oplus e_3) = 3 \succ w(F \oplus e_4 \oplus e_1 \oplus e_3) = 2$ as shown in Figure 3.12 (IV) and (V). Therefore, it does not satisfy the property in (3.6). Therefore, by Theorem 2, the generic edge-adding algorithm cannot guarantee finding the minimum depth broadcast tree. This can be verified by noticing that one possible tree generated by the generic edge-adding algorithm is $F \oplus e_5 \oplus e_1 \oplus e_3$, which is not optimal.

Compatibility with Edmonds' Algorithm

The depth metric in (3.13) does not satisfy the property in (3.9). Consider the network topology D as shown in Figure 3.13 (I). Let $F = (N(D), \{a_2\})$. Note that $w(F \oplus a_5) = 1 \prec w(F \oplus a_4) = 2$ as shown in Figure 3.13 (II) and (III), but $w(F \oplus a_5 \oplus a_1 \oplus a_3) = 3 \succ w(F \oplus a_4 \oplus a_1 \oplus a_3) = 2$ as shown in Figure 3.13 (IV) and (V). Therefore, the depth metric does not satisfy the property in (3.9). Therefore, by Theorem 3, Edmonds' algorithm cannot guarantee producing the minimum depth broadcast arborescence. This can be verified by noticing that one possible Edmonds' algorithm output is $F \oplus a_5 \oplus a_1 \oplus a_3$, which is not optimal.

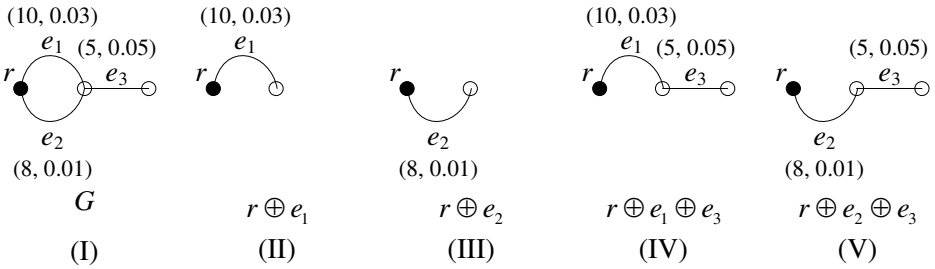


Figure 3.14: The Prim algorithm example for the most reliable widest bandwidth broadcast routing

3.6.2 Case 2: Most Reliable Widest Bandwidth Broadcast Routing

Consider a most reliable widest bandwidth metric whose design is based on the following two observations. First, to maximize the capacity of links over the broadcast tree T, it is desirable to design a metric maximizing the bandwidth of the tree. The bandwidth of the tree is defined as the minimum bandwidth among all the links of the tree. Second, to ensure successful delivery of a broadcast message, a proper broadcast routing metric should minimize the probability that the broadcast message is lost by some nodes in the broadcast tree T. This probability can be calculated as

$$1 - \prod_{i \in E(T)} (1 - p_i) \approx \sum_{i \in E(T)} p_i, \tag{3.15}$$

where p_i is the estimated packet loss rate for a link in the broadcast tree, and the approximation is based on the assumption that each link's packet loss rate is small enough. This is valid for most scenarios. Hence, it is reasonable to consider the following lexicographic metric

$$(b, p), \tag{3.16}$$

where b is the bandwidth of the tree, p is the packet loss rate of the tree. We have $w(b_1, p_1) \preceq w(b_2, p_2)$ if either $b_1 > b_2$ or $b_1 = b_2, p_1 \leq p_2$. The objective of the most reliable widest bandwidth broadcast routing is to find the lowest packet loss rate spanning tree within all widest bandwidth spanning trees.

Compatibility with Prim's Algorithm

The metric in (3.16) does not satisfy the required property in (3.4). Consider the network topology in Figure 3.14 (I). Let $w(e_1) = (10, 0.03), w(e_2) = (8, 0.01), w(e_3) = (5, 0.05)$. Note that $w(r \oplus e_1) = (10, 0.03) \prec w(r \oplus e_2) = (8, 0.01)$ but $w(r \oplus e_1 \oplus e_3) = (5, 0.08) \succ w(r \oplus e_2 \oplus e_3) = (5, 0.06)$, where $r \oplus e_1, r \oplus e_2, r \oplus e_1 \oplus e_3, r \oplus e_2 \oplus e_3$ are shown in Figure 3.14 (II),

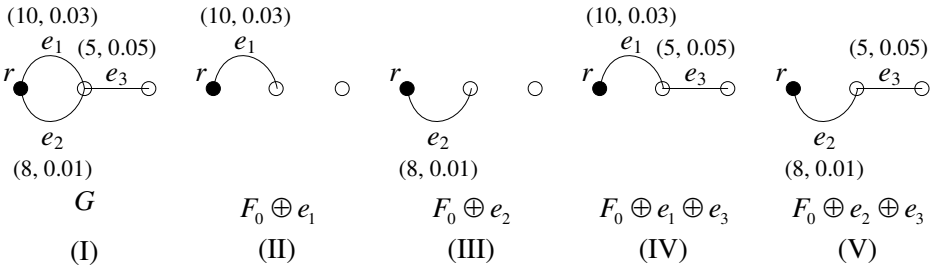


Figure 3.15: The generic edge-adding algorithm example for the most reliable widest bandwidth broadcast routing

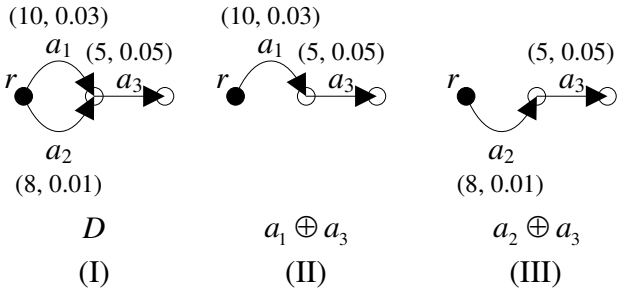


Figure 3.16: The Edmonds algorithm example for the most reliable widest bandwidth broadcast routing

(III), (IV), (V), respectively. Therefore, the most reliable widest bandwidth metric does not satisfy the property in (3.4). By Theorem 1, Prim's algorithm cannot guarantee producing the MST. This can be easily shown by noticing that the algorithm output is $r \oplus e_1 \oplus e_3$, which is not optimal.

Compatibility with the Generic Edge-adding Algorithm

The metric in (3.16) does not have the required property in (3.6). Consider the network topology G in Figure 3.15 (I). Let $w(e_1) = (10, 0.03), w(e_2) = (8, 0.01), w(e_3) = (5, 0.05)$. Note that $w(F_0 \oplus e_1) = (10, 0.03) \prec w(F_0 \oplus e_2) = (8, 0.01)$ but $w(F_0 \oplus e_1 \oplus e_3) = (5, 0.08) \succ w(F_0 \oplus e_2 \oplus e_3) = (5, 0.06)$, where $F_0 = (N(G), \emptyset)$, and $F_0 \oplus e_1, F_0 \oplus e_2, F_0 \oplus e_1 \oplus e_3, F_0 \oplus e_2 \oplus e_3$ are shown in Figure 3.15 (II), (III), (IV), (V), respectively. Therefore, the most reliable widest bandwidth metric does not satisfy the property in (3.6). By Theorem 2, the generic edge-adding algorithm cannot guarantee producing the MST. This can be easily shown by noticing that one possible algorithm output is $F_0 \oplus e_1 \oplus e_3$, which is not optimal.

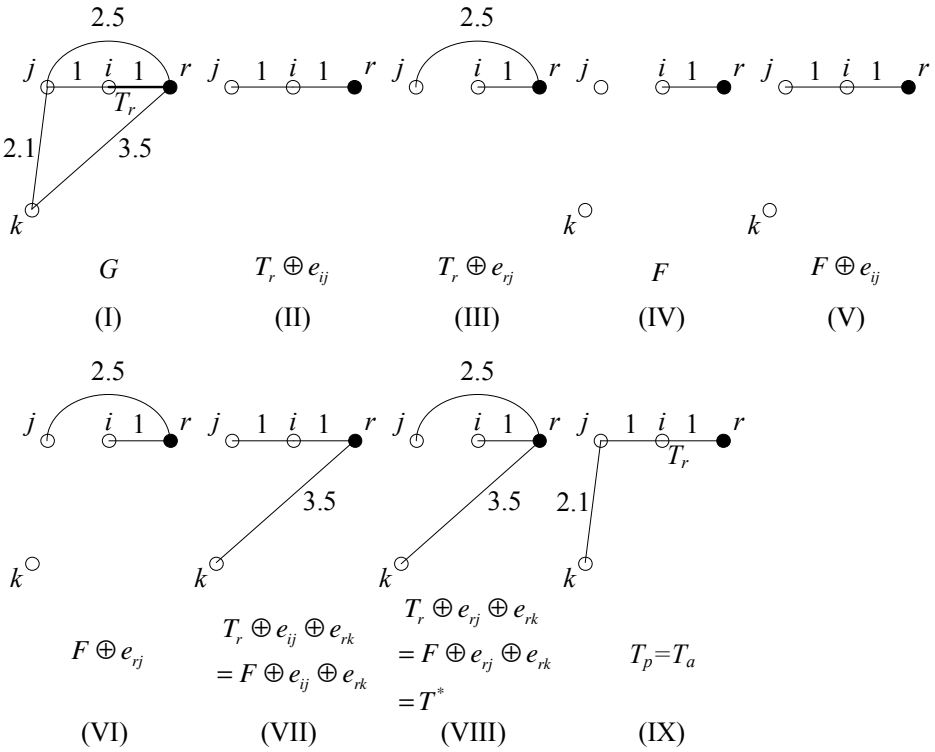


Figure 3.17: The Prim algorithm and generic edge-adding algorithm example for minimum energy broadcast routing

Compatibility with Edmonds' Algorithm

The metric in (3.16) does not have the required property in (3.9). Consider the network topology D shown in Figure 3.16 (I). Let $w(a_1) = (10, 0.03), w(a_2) = (8, 0.01), w(a_3) = (5, 0.05)$. Note that $w(a_1) \prec w(a_2), w(a_1 \oplus a_3) = (5, 0.08) \succ w(a_2 \oplus a_3) = (5, 0.06)$, where $a_1 \oplus a_3, a_2 \oplus a_3$ are shown in Figure 3.16 (II), (III), respectively. Therefore, the most reliable widest bandwidth metric does not satisfy the property in (3.9). By Theorem 3, Edmonds' algorithm does not guarantee optimality. This can be shown by noticing that one possible algorithm output $a_1 \oplus a_3$ is not the minimum weight spanning arborescence.

3.6.3 Case 3: Minimum Energy Broadcast Routing

Let us consider the minimum energy broadcast routing, whose metric is defined in (3.1).

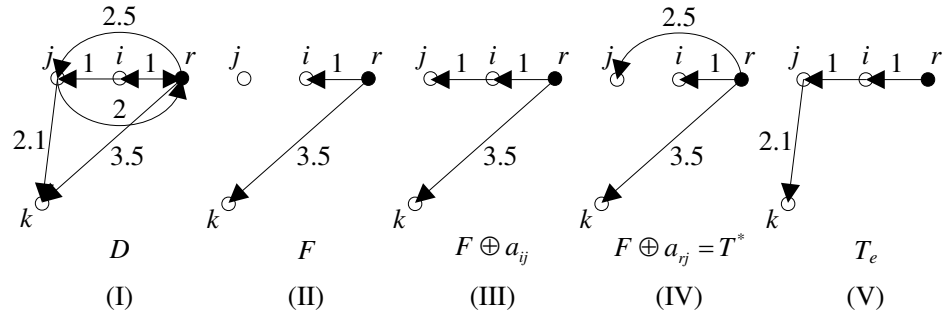


Figure 3.18: The Edmonds algorithm example for the minimum energy broadcast routing

Compatibility with Prim's Algorithm

The metric in (3.1) does not satisfy the property in (3.4). Consider the network topology G shown in Figure 3.17 (I). The numbers associated with the edges are the weights of these edges. Let the partial spanning tree be $T_r = (\{r, i\}, \{e_{ri}\})$ as shown by the bold part of Figure 3.17 (I). Note that $w(T_r \oplus e_{ij}) = 2 \prec w(T_r \oplus e_{rj}) = 2.5$ as shown in Figure 3.17 (II) and (III), but $w(T_r \oplus e_{ij} \oplus e_{rk}) = 4.5 \succ w(T_r \oplus e_{rj} \oplus e_{rk}) = 3.5$ as shown in Figure 3.17 (VII) and (VIII). Hence, the metric in (3.1) does not satisfy the property in (3.4). Therefore, by Theorem 1, Prim's algorithm cannot guarantee producing the minimum energy broadcast routing tree based on the metric in (3.1). This can be verified by noticing that $w(T_p) = 4.1 \succ w(T^*) = 3.5$, where T_p is the Prim tree generated by Prim's algorithm as shown in Figure 3.17 (IX) and T^* is the minimum energy broadcast tree as shown in Figure 3.17 (VIII).

Compatibility with the Generic Edge-adding Algorithm

The metric in (3.1) does not satisfy the property in (3.6). Again, consider the network topology G shown in Figure 3.17 (I). Suppose the partial spanning forest generated by the generic edge-adding algorithm is $F = (N(G), \{e_{ri}\})$ as shown in Figure 3.17 (IV). Note that $w(F \oplus e_{ij}) = 2 \prec w(F \oplus e_{rj}) = 2.5$ as shown in Figure 3.17 (V) and (VI), but $w(F \oplus e_{ij} \oplus e_{rk}) = 4.5 \succ w(F \oplus e_{rj} \oplus e_{rk}) = 3.5$ as shown in Figure 3.17 (VII) and (VIII). Hence, the metric does not satisfy the property in (3.6). Therefore, by Theorem 2, the generic edge-adding algorithm cannot guarantee producing the minimum weight broadcast routing tree based on the metric in (3.1). This can be verified by noticing that, for the generic edge-adding tree T_a and the minimum weight broadcast routing tree T^*, $w(T_a) = 4.1 \succ w(T^*) = 3.5$, as shown in Figure 3.17 (IX) and (VIII).

Compatibility with Edmonds' Algorithm

The metric in (3.1) does not satisfy the required property in (3.9). Consider the connected and directed network topology D whose root node is r as shown in Figure 3.18 (I). The numbers associated with arcs are the weights of these arcs. Let F be a partial spanning arborescence forest as shown in Figure 3.18 (II). Note that $w(a_{ij}) \prec w(a_{rj})$ and $w(F \oplus a_{ij}) = 4.5 \succ w(F \oplus a_{rj}) = 3.5$ as shown in Figure 3.18 (III) and (IV). Hence, the minimum energy broadcast routing metric in (3.1) does not satisfy the property in (3.9). Therefore, Edmonds' algorithm does not produce the minimum weight broadcast routing arborescence based on the metric in (3.1). This conclusion can be verified by noticing that $w(T_e) = 4.1 \succ w(T^*) = 3.5$, where T_e is the Edmonds arborescence generated by Edmonds' algorithm and T^* is the minimum energy broadcast routing arborescence.

3.6.4 Case 4: Lexicographically Optimal Broadcast Routing based on Packet Loss Rate

Consider a broadcast routing metric finding the lexicographically OBT in terms of packet loss rate on each link. The metric is a sorted list of link packet loss rates of the considered network topology. Elements of the list are sorted in a decreasing order. Mathematically, the metric can be expressed as: $w(a) = (a_1, a_2, ..., a_m)$, where m is the total number of links in the network topology a, and $a_1 \geq a_2 \geq ... \geq a_m$ are the packet loss rates of links. Consider two network topologies a and b, and let $w(a) = (a_1, a_2, ..., a_m), w(b) = (b_1, b_2, ..., b_n)$. Define $w(a) = w(b)$, if $m = n, a_i = b_i, i = 1, 2, ..., m$. Define $w(a) \prec w(b)$, if the following two conditions hold: first, there exists a number l such that $a_i = b_i, 1 \leq i \leq l-1$; second, either $a_l < b_l$ when $l - 1 < \min\{m, n\}$, or $m < n$ when $l - 1 = m$. We further define $w(a) \preceq w(b)$ if either $w(a) = w(b)$ or $w(a) \prec w(b)$. Practically, assuming that a reliable link level retransmission scheme is used to ensure reliable delivery of broadcast packets over each link, a metric like this may be used to consider the link retransmission count of the entire broadcast tree. To facilitate the case study, we first prove the following lemma.

Lemma 2. *The lexicographical optimality metric satisfies the properties in* (3.4)*,* (3.6) *and* (3.9)*.*

Proof. We first prove that the lexicographical optimality metric satisfies the properties in (3.4) and (3.6). Let p and q be network topologies consisting of m and n edges, respectively, and e and e' be two edges. Given that $w(p \oplus e) \preceq w(p \oplus e')$, we need to prove that $w(p \oplus e \oplus q) \preceq w(p \oplus e' \oplus q)$. By the metric definition, we have $w(p) = (p_1, p_2, ..., p_m)$ and $w(q) = (q_1, q_2, ..., q_n)$. Denote $a = p \oplus e, b = p \oplus e', c = p \oplus e \oplus q, d = p \oplus e' \oplus q$. When $w(p \oplus e) = w(p \oplus e')$, by the metric definition, we have $w(p \oplus e \oplus q) = w(p \oplus e' \oplus q)$. When $w(p \oplus e) \prec w(p \oplus e')$, by the metric definition, $\exists 1 \leq i \leq m+1$ such that $a_j = b_j$ for $1 \leq j \leq i-1$ and $a_i < b_i$. Denote $q^{>} = \{q_l | q_l \geq b_i, 1 \leq l \leq n\}, q^{<} = \{q_l | q_l \leq a_i, 1 \leq l \leq$

$n\}, q^{<>} = \{q_l | a_i < q_l < b_i, 1 \leq l \leq n\}$. During the $a \oplus q$ and $b \oplus q$ operation, elements in $q^{>}$ are added before a_i and b_i in $w(a)$ and $w(b)$, respectively, and elements in $q^{<}$ are added after a_i and b_i in $w(a)$ and $w(b)$, respectively. If $q^{<>} = \emptyset$, between $w(c)$ and $w(d)$ the first differing elements are a_i in $w(c)$ and b_i in $w(d)$. By the metric definition, we have $w(p \oplus e \oplus q) = w(c) \prec w(d) = w(p \oplus e' \oplus q)$. If $q^{<>} \neq \emptyset$, let $\widetilde{q^{<>}} = \{q_k, ..., q_{k+o}\}$ be the ordered set of $q^{<>}$ following the non-increasing order. Between $w(c)$ and $w(d)$, the first differing elements are q_k in $w(c)$ and b_i in $w(d)$. By the definition of q_k and $q^{<>}$, it follows that $q_k < b_i$. By the metric definition, we have $w(p \oplus e \oplus q) = w(c) \prec w(d) = w(p \oplus e' \oplus q)$. In summary, we have proved that given that $w(p \oplus e) \preceq w(p \oplus e')$, we have $w(p \oplus e \oplus q) \preceq w(p \oplus e' \oplus q)$. Therefore, the lexicographical optimality metric satisfies the properties in (3.4) and (3.6).

We next prove that the lexicographical optimality metric satisfies the property in (3.9). Let p and q be two network topologies consisting of m arcs, and u be a network topology consisting of n arcs. Given that $w(p) \preceq w(q)$, we need to prove that $w(p \oplus u) \preceq w(q \oplus u)$. By the metric definition, we have $w(p) = (p_1, p_2, ..., p_m)$, $w(q) = (q_1, q_2, ..., q_m)$ and $w(u) = (u_1, u_2, ..., u_n)$. When $w(p) = w(q)$, by the metric definition, we have $w(p \oplus u) = w(p \oplus u)$. When $w(p) \prec w(q)$, by the metric definition, $\exists 1 \leq i \leq m$ such that $p_j = q_j$ for $1 \leq j \leq i-1$ and $p_i < q_i$. Denote $u^{>} = \{u_l | u_l \geq q_i, 1 \leq l \leq n\}, u^{<} = \{u_l | u_l \leq p_i, 1 \leq l \leq n\}, u^{<>} = \{u_l | p_i < u_l < q_i, 1 \leq l \leq n\}$. During the $p \oplus u$ and $q \oplus u$ operation, elements in $u^{>}$ are added before p_i and q_i in $w(p \oplus u)$ and $w(q \oplus u)$, respectively, and elements in $u^{<}$ are added after p_i and q_i in $w(p \oplus u)$ and $w(q \oplus u)$, respectively. If $u^{<>} = \emptyset$, between $w(p \oplus u)$ and $w(q \oplus u)$ the first differing element in $w(p \oplus u)$ is p_i, and the first differing element in $w(q \oplus u)$ is q_i. By the metric definition, we have $w(p \oplus u) \prec w(q \oplus u)$. If $u^{<>} \neq \emptyset$, let $\widetilde{u^{<>}} = \{u_k, ..., u_{j+o}\}$ be the ordered set of $u^{<>}$ following the non-increasing order. Between $w(p \oplus u)$ and $w(q \oplus u)$, the first differing element in $w(p \oplus u)$ is u_k, and the first differing element in $w(p \oplus u)$ is q_i. By the definition of u_k and $u^{<>}$, it follows that $u_k < q_i$. By the metric definition, we have $w(p \oplus u) \prec w(q \oplus u)$. In summary, we have proved that given that $w(p) \preceq w(q)$, we have $w(p \oplus u) \preceq w(q \oplus u)$. Therefore, the lexicographical optimality metric satisfies the properties in (3.9). □

Compatibility with Prim's Algorithm

By Lemma 2 and Theorem 1, Prim's algorithm can find the OBT.

Compatibility with the Generic Edge-adding Algorithm

By Lemma 2 and Theorem 2, the generic edge-adding algorithm can find the OBT.

Compatibility with Edmonds' Algorithm

By Lemma 2 and Theorem 3, Edmonds' algorithm can find the OBT.

3.6.5 Case 5: Maximum Network Lifetime Broadcast Routing

Consider the broadcast routing metric which maximizes the network lifetime [30]. It is defined as

$$\min_{i \in N(T)} \frac{\varepsilon_i}{\max\limits_{(i,j) \in E(T)} P_{ij}}, \tag{3.17}$$

where $N(T)$ is the node set of the broadcast tree T, ε_i is the residual energy of node i, $E(T)$ is the link set of the broadcast tree T, and P_{ij} is the minimum power consumption of transmission from node i to node j. The broadcast routing metric definition given in (3.17) is essentially the minimum lifetime of all the links in the broadcast tree T. Also note the fact that the underlying network topology is undirected when the residual energy of each node is equal, and the underlying network topology is directed when the residual energy of each node is different [30].

Compatibility with Prim's Algorithm

Note that the property definition in Theorem 1 is based on undirected network topologies. Therefore, when the residual energy of each node is different, the metric in (3.17) is not compatible with the property definition in Theorem 1. When the residual energy of each node is equal, the network can be modeled as an undirected graph and hence Theorem 1 can be used to check the applicability of Prim's algorithm. In this case, it is clear that the minimization operation of the metric satisfies the property in (3.4). Therefore, we have the following conclusions.

- If the residual energy of each node is different, Prim's algorithm cannot find the maximum network lifetime broadcast tree based on the metric in (3.17).
- If the residual energy of each node is equal, Prim's algorithm produces the maximum network lifetime broadcast tree based on the metric in (3.17).

Compatibility with the Generic Edge-adding Algorithm

Since the property definition in Theorem 2 is based on undirected network topologies, the metric in (3.17) is not compatible with the property definition in Theorem 2 when the residual energy of each node is different. When the residual energy of each node is equal, we further check if the metric in (3.17) satisfies the property in (3.6). Clearly, the minimization operation of the metric satisfies the property in (3.6). Therefore, we have the following conclusions.

- If the residual energy of each node is different, the generic edge-adding algorithm cannot find the maximum network lifetime broadcast tree based on the metric in (3.17).

- If the residual energy of each node is equal, the generic edge-adding algorithm produces the maximum network lifetime broadcast tree based on the metric in (3.17).

Compatibility with Edmonds' Algorithm

Since the property definition in Theorem 3 is based on directed network topologies and undirected topologies can be transformed into directed network topologies, the metric in (3.17) is compatible with the property definition in Theorem 3. Next, we check if the metric in (3.17) satisfies the property in (3.9). Clearly, the minimization operation of the maximum network lifetime metric satisfies the property in (3.9). Therefore, we have the following conclusions.

- If the residual energy of each node is different, Edmonds' algorithm produces the maximum network lifetime broadcast arborescence based on the metric in (3.17).
- If the residual energy of each node is equal, Edmonds' algorithm also produces the maximum network lifetime broadcast arborescence based on the metric in (3.17).

It is important to note that besides Edmonds' algorithm there are other algorithms which can also find the OBT for the network lifetime metric (3.17). In fact, in [30], a new algorithm, termed as DMST, has been shown to be able to find the OBT for (3.17). Currently, the exact necessary and sufficient condition for DMST to compute OBTs for any given metric is still an open problem and is part of our future work.

3.7 Conclusion

In this chapter, the potential incompatibility between broadcast routing metrics and BTC algorithms is identified. Using our broadcast routing algebra model, we develop and prove the necessary and sufficient properties of broadcast routing metrics which guarantee optimal broadcast routing in both undirected network topologies and directed network topologies. Further, we study the uniqueness properties required by distributed routing implementation. These properties are used to identify optimality compatibility between broadcast routing metrics and broadcast routing algorithms.

Chapter 4

Optimal Cache-based Route Repair for Real-time Traffic in Mobile Ad hoc Networks

4.1 Introduction

In mobile ad hoc networks (MANETs), the constant movement of wireless nodes and the fading of channels can cause frequent route breaks for on-going communications. Real-time applications, on the other hand, require stable routes to ensure satisfactory performance. Hence, fast route repair mechanisms that can quickly reestablish new routes upon route breaks must be used in MANETs to minimize the interruption to on-going real-time communications.

Among the existing designs of ad hoc routing protocols, on-demand protocols, such as DSR [28] and AODV [43], have been proven to be able to quickly find paths in highly mobile networks and hence are very popular for supporting real-time flows. In these protocols, a source node re-floods the entire network to search for a new route when the current route breaks. Such flooding-based route discovery can return up-to-date new paths. However, flooding-based route discovery is usually very expensive in terms of message overhead. Hence, it can potentially cause network-wide congestion. When congestion happens, it may require a fairly long time to find a path. Therefore, path repair schemes that solely rely on flooding-based route discovery cannot satisfy the need of many time critical real-time applications.

As a solution to the above problem, cache-based route repair schemes [34, 39, 22, 32] have been proposed to reduce the frequency of flooding. These schemes exploit the fact that a round of flooding-based route discovery usually finds multiple paths to the destination. The source node can cache these paths as backups. When the route in use breaks, the source first tries to use the cached paths to continue its communications before it resorts to flooding.

Hence, the cached paths may provide quicker route repair and impose less message overhead. However, aggressive use of the cached paths for route repair does not always bring benefit since there is always a chance that the cached paths are also broken due to the mobility of nodes [44]. Sending packets along broken cached paths can only cause excessive packet losses and delay. Therefore, there exists an interesting trade-off point between the flooding-based route discovery and the cache-based route repair schemes. Most of existing works on cache-based route repair [34, 39, 22, 32, 52], unfortunately, only focus on heuristic designs and simulation-based evaluation. There is only one work that provides theoretical analysis on cache-based route repair [35]. This work, however, only computes the interruption and overhead caused by using the cached paths and ignores the interruption and overhead caused by flooding-based route discovery. This omission is not tolerable since the flooding-based route discovery often imposes much higher overhead and incurs much larger interruption duration than cache-based route repair. Therefore, there is still a severe lack of correct insights into the trade-off point between cache-based route repair and flooding-based route discovery.

In this chapter, we provide a mathematical analysis on this trade-off point and derive an optimal cache policy that is the best strategy between flooding and cache-based repair. Our unique contributions are listed as follows:

- While using cached paths for route repair has been proposed in many existing literature, we are the first to formulate an analytical framework to identify the optimal trade-off point between cache-based route repair and flooding-based route discovery in MANETs. This analytical framework accurately captures the factors that affect the performance of route repair strategies. These factors include path length, mobility of nodes and network size.
- Based on our analytical framework, an easy-to-implement optimal cache policy is proposed and proved for cache-based route repair schemes. This policy finds the optimal trade-off point between flooding and cache-based route repair. Its correctness and advantages are analytically proved and then validated by simulations.

The remaining part of this chapter is organized as follows. The system model is presented in Section 4.2. The optimization objective is developed in Section 4.3 and its relationship with the characteristics of cached paths is studied in Section 4.4. The optimal cache policy is derived in Section 4.5, and the policy is generalized to cover more optimization objectives in Section 4.6. The performance advantages of the optimal cache policy are validated by simulations in Section 4.7. Finally, Section 4.8 concludes the whole chapter.

4.2 System Model

To derive the optimal strategy for cache-based route repair, we first need to have a model of the network mobility, a model of the network, and a model for the routing protocol. Then, we can use these models to derive the optimal cache policy.

4.2.1 Mobility Model

In this chapter, the mobility of nodes is modeled by the Semi-Markov Smooth (SMS) mobility model [70]. We use SMS mobility model because it has been shown in [70] to be able to accurately capture realistic node movement. For example, it can capture physical motion's smooth nature, can adapt to diverse network application scenarios and can ensure the node distribution does not change over time. All of these realistic features are not present in other more traditional but more naive mobility models such as the random-way-point model.

Briefly speaking, the SMS model works as follows. In the SMS model, node movement consists of four consecutive phases: speed up phase (α phase), middle smooth phase (β phase), slow down phase (γ phase), and pause phase.

At the beginning of the speed up phase, the node is stationary. Then, it accelerates from the stationary state to the target speed v_α. The direction ϕ_α does not change in this process. The target speed $v_\alpha \in [v_{min}, v_{max}]$, the direction $\phi_\alpha \in [0, 2\pi)$, and the total number of time steps $\alpha \in [\alpha_{min}, \alpha_{max}]$. The parameters v_α, ϕ_α and α are independent and uniformly distributed.

After the speed up phase, the node transits into the middle smooth phase. In this phase, the initial speed $v_0 = v_\alpha$ and the initial direction $\phi_0 = \phi_\alpha$. The speed and direction change from v_α and ϕ_α at each time step based on a memory level parameter $\zeta \in [0, 1]$. This memory level parameter captures the temporal correlation of velocity between consecutive steps. This phase lasts β steps, where $\beta \in \mathbb{Z}^+$ is uniformly distributed over $[\beta_{min}, \beta_{max}]$.

After the middle smooth phase, the node enters the slow down phase. In this phase, the node comes to a complete stop within $\gamma \in \mathbb{Z}^+$ steps, where γ is uniformly distributed over $[\gamma_{min}, \gamma_{max}]$. The direction ϕ_γ is randomly chosen, but is correlated to node movement direction at the end of the middle smooth phase. The direction ϕ_γ does not change in this phase.

4.2.2 Network Model

Consider a MANET, where there are N mobile nodes in the network, and each node's movement is independent and identically distributed (i.i.d.). For each node, the average speed is $\bar{v}$, and the radio coverage radius is R. When two nodes are within the communication range of R, we define that there exists a link between the two nodes, i.e., these two nodes

can communicate with each other. When there is a set of links which can connect the source node with the destination node, we define that there exists a path between the source node and the destination node.

4.2.3 Protocol Model

Our model of cache-based route repair can capture any routing protocol that uses flooding-based route discovery, regardless if the routing protocol is source routing (e.g. DSR) or hop-by-hop routing (e.g. AODV). In our model, we assume that a source node first uses flooding-based route discovery to find a path to its destination. It sends out a broadcast route request (RREQ) message. Each intermediate node may receive multiple copies of the RREQ message and only selects one to forward to its own neighbors to limit the message overhead. When the destination receives multiple route request messages, it can discover multiple paths and return this information to the source node through a route reply (RREP) message. The best path among the returned paths is then set up as the path to relay data packets to the destination as follows. In the hop-by-hop routing case (e.g., AODV), the routing tables at the relaying nodes along the best path are setup by the RREP message that travels back to the source. In the source routing case (e.g. DSR), after the source receives the RREP message, the entire selected best path is copied to the headers of the data packets to the destination. The other paths discovered by the flooding process are returned by the RREP to the source and are cached as backup paths for path repair.

We denote this best path as p_0 and the set of cached paths as $\mathcal{I} = \{p_1, p_2, p_3, ..., p_{|\mathcal{I}|}\}$, where $\mathcal{I}$ is a sorted list of paths and is called the path cache. We assume that only the node-disjoint paths to the source node are cached. This is a common practice widely used in multipath routing and path repair to ensure that the cached backup paths are independent of the status of the path in use. Essentially, this means that $p_0, p_1, p_2, ..., p_{|I|}$ are node-disjoint. The flooding process has a large routing message overhead, especially when the network scale is relatively large or the network resource (e.g. bandwidth and energy) is limited. The routing messages sent during the flooding process are called the *flooding packets* in this chapter.

Upon a route break, if $\mathcal{I}$ is not empty, the source picks the first cached path $p_1 \in \mathcal{I}$ for route repair. It sends out an *exploration packet* to the destination along p_1. If p_1 is already broken, a route error message is generated when the exploration packet reaches the first broken link. This route error message travels back to the source to notify the failure of p_1. Then, p_1 is removed from the path cache $\mathcal{I}$ and the next path p_2 in $\mathcal{I}$ is explored for path repair. This cache-based path exploration process continues until finally a valid path p_i is found. In such a case, the destination receives the exploration packet and returns the confirmation to the source. The confirmation message will trigger the right routing table information to be set up for the paths p_i. In the case of hop-by-hop routing (e.g. AODV), all the intermediate nodes on p_i will have their routing table updated by the confirmation message. In the case of source routing (e.g. DSR), only the routing table in the source will be updated. Then,

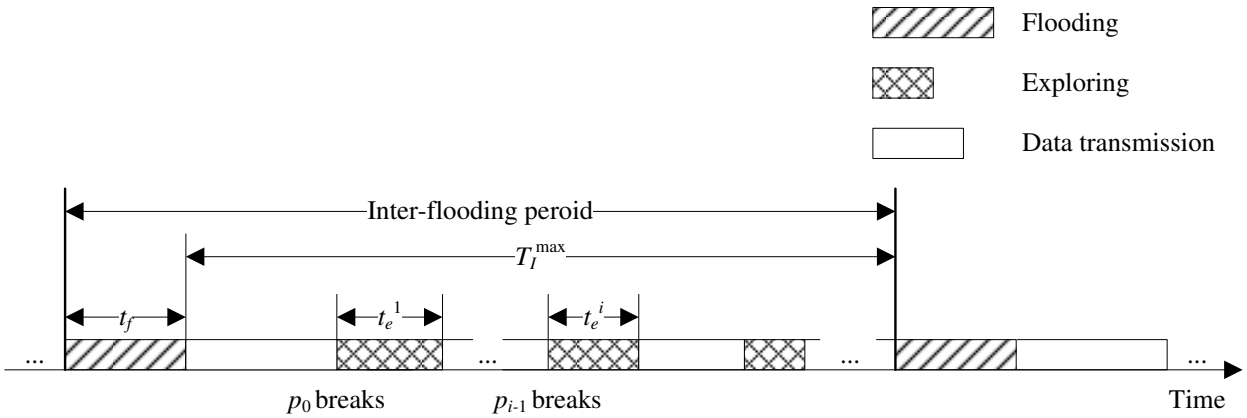


Figure 4.1: A repair cycle

the source removes p_i from the path cache $\mathcal{I}$ and resumes its communication using p_i. The cache-based path exploration is successful in this case. If $\mathcal{I}$ becomes empty before a valid path is found, the source node resorts to flooding-based route discovery for repairing the broken route.

4.3 Problem Formulation

To formulate the problem of optimal cache-based route repair strategy, let us first take a detailed analysis of the cache-based route repair process. Upon a route break, a source node has two possible actions to repair the route. The first action is flooding-based route discovery, which happens when the source node has no cached path in $\mathcal{I}$. The source gets the up-to-date set of paths from the flooding process but pays for the high message overhead of flooding.

The second action is to explore the head-of-line path in the path cache $\mathcal{I}$. The paths in $\mathcal{I}$ are learned from the last flooding process. The exploration action happens when $\mathcal{I}$ is not empty. If the explored path is already broken, the exploration process incurs message overhead and increases the duration of communication interruption.

Essentially, the frequency, overhead and delay of the above two path repair actions greatly affect the interruption duration and the routing overhead of a real-time session. These two actions are in turn determined by the *cache policy*, which determines the set of paths cached in $\mathcal{I}$ and these paths' sequence in $\mathcal{I}$. For example, if the number of paths in $\mathcal{I}$ is small, the frequency of flooding is large. If the paths and their sequence in $\mathcal{I}$ is not appropriately selected, the exploration action may often fail. Therefore, the problem of optimal cache-based route repair is essentially a design problem for the optimal cache policy, which jointly minimizes the interruptions to communications and the message overhead.

We formulate this goal for the optimal cache policy as an objective function that captures both the message overhead and the time interruption of the two path repair actions. For a real-time communication session that lasts T seconds, the objective function is defined as:

$$\max_{\pi(\mathcal{I})} \frac{N_d^T}{N_o^T}, \tag{4.1}$$

where $\pi(\mathcal{I})$ is a cache policy that sets the path cache $\mathcal{I}$, N_d^T is the number of data packets transmitted during the lifetime of the real-time session, and N_o^T is the number of overhead packets for establishing and repairing routes for the real-time session. The physical meaning of the objective function is to maximize the ratio between the number of real-time data packets delivered to the destination and the number of overhead packets for repairing and establishing routes. This objective function captures both the time and message overhead for establishing and repairing routes. If a cache policy creates long interruptions to the real-time session, it reduces the number of real time packets that can be delivered during time T and hence decreases the value of the objective function. If a cache policy creates too much routing overhead, it increases the denominator in the objective function and hence also reduces the objective function's value.

It is important to note that we do not claim that (4.1) is the only possible formulation of objective functions. Actually, our analysis for solving (4.1) can be generalized to cover a large range of possible objective functions. In Section 4.6, we will discuss these other possible formulations. Also note that when packet sizes are different for different types of packets, we can also take into account the packet size in the objective function (4.1) by multiplying the packet transmission time before the packet numbers. This packet transmission time for the same packet type can be approximated as a constant. This is because the variations in control packet (e.g., RREQ) sizes usually are negligible compared to the dominant packet transmission overhead in physical and MAC layers. Hence, considering packet sizes simply means multiplying constant packet transmission time to the variables that we want to optimize in (4.1). This modification of the optimization problem in (4.1) does not affect our analytical process. Therefore, we focus on the objective in (4.1) in the rest of this chapter.

4.4 Relationship Between Objective Function and Cached Paths

In this section, we identify the relationship between objective function in (4.1) and the characteristics of cached paths, so that an optimal cache policy can be designed based on this relationship. First, a mathematical model of the cache-based route repair process is established. Based on this model, the objective function in (4.1), which is defined over the entire lifetime of a session, is converted to a tractable formulation over a short time period. Finally, by analyzing this short time period, the objective function is expressed by the characteristics of cached paths.

4.4.1 Renewal Model of the Cache-based Route Repair Process

To establish a mathematical model of the cache-based route repair process, note that when a flooding-based route discovery is completed at time t, the cache $\mathcal{I}$ in the source is filled with newly discovered paths and all the old cached paths are removed. This removal of old cached paths means that the future communication states of the real-time session has no dependence on its states before time t. Hence, the communication states of the real-time flow can be modeled as a renewal process, where each flooding-based route discovery is a renewal event. We define the interval between two consecutive flooding events, i.e. the inter-flooding period, as a *repair cycle*. As shown in Figure 4.1, each repair cycle begins with a flooding-based route discovery that lasts t_f time. For each occurrence of route break in the cycle, the source node explores the cached paths one by one until it finds one valid path to resume the communication session. Each exploring period is the duration for exploring one cached path for route repair and t_e^i denotes this duration for exploring path $p_i \in \mathcal{I}$. The number of exploring periods in the repair cycle equals the number of paths initially cached in $\mathcal{I}$ after the flooding at the beginning of the cycle.

4.4.2 Objective Function Defined over the Repair Cycle

With the above renewal model of the path repair process, we can translate the objective function in (4.1), which is defined over the entire duration of a real-time session, to an objective function that is defined over a repair cycle. The purpose of this step is to simplify the mathematical analysis since it enables us to focus on a relatively short period.

To translate the objective function in (4.1), we first divide the duration of the real-time session T as the combination of multiple repair cycles. Assuming there are L repair cycles over the time interval T, the objective function in (4.1) becomes:

$$\max_{\pi(\mathcal{I})} \frac{N_d^T}{N_o^T} = \max_{\pi(\mathcal{I})} \frac{\sum_{l=1}^{L} N_d^l}{\sum_{l=1}^{L} N_o^l}, \tag{4.2}$$

where N_d^l is the number of data packets transmitted during the lth repair cycle, and N_o^l is the number of the overhead packets transmitted during the lth repair cycle.

Assume that T is much larger than the inter-flooding period, i.e. L is a very large number. Since each inter-flooding period is independent and identically distributed, by the law of

large numbers, the objective function in (4.2) becomes

$$\max_{\pi(\mathcal{I})} \frac{\sum_{l=1}^{L} N_d^l \Big/ L}{\sum_{l=1}^{L} N_o^l \Big/ L} \approx \max_{\pi(\mathcal{I})} \frac{E[N_d^l]}{E[N_o^l]} = \max_{\pi(\mathcal{I})} \frac{N_d}{N_o}, \tag{4.3}$$

where N_d is the expected number of data packets transmitted during a repair cycle, and N_o is the expected number of overhead packets transmitted during a repair cycle. Essentially, the objective function in (4.3) is defined over a repair cycle and hence we can further analyze it by closely examining the formation of a repair cycle.

4.4.3 Objective Function v.s. Path Characteristics

In this section, we derive the relationship between the objective function in (4.3) and the characteristics of cached paths in $\mathcal{I}$. First, denoting the real-time session data rate as r packets per unit time, the objective function in (4.3) becomes

$$\max_{\pi(\mathcal{I})} \frac{N_d}{N_o} = \max_{\pi(\mathcal{I})} \frac{T_d r}{N_f + N_e}, \tag{4.4}$$

where T_d is the expected time used for data transmission during a repair cycle, N_f is the expected number of flooding packets transmitted during the flooding-based route discovery, and N_e is the expected number of exploring packets transmitted for exploring cached paths during a repair cycle. Note that exploring a cached path may either succeed when the path is alive or fail when the path is broken. Hence, N_e includes the exploring packets incurred in both cases.

Since r is a constant, (4.4) is equivalent to:

$$\max_{\pi(\mathcal{I})} \frac{T_d}{N_f + N_e} = \max_{\pi(\mathcal{I})} \frac{E[T_{\mathcal{I}}^{\max} - \sum_{p_i \in \mathcal{I}} t_e^i]}{N_f + \sum_{p_i \in \mathcal{I}} N_i}, \tag{4.5}$$

where the term $T_{\mathcal{I}}^{\max}$ is the duration of a repair cycle minus the time spent for flooding the network (See Figure 4.1). The term t_e^i is the time used for exploring the ith path in the set $\mathcal{I}$ during a repair cycle. The term N_i is the expected number of overhead packets transmitted when exploring the ith path in $\mathcal{I}$. Note that, in general, the time used for exploring paths is much less than the time length of the inter-flooding period minus the flooding period, i.e., $\sum_{p_i \in \mathcal{I}} t_e^i \ll T_{\mathcal{I}}^{\max}$. Therefore, the objective function in (4.5) can be approximated as follows

$$\max_{\pi(\mathcal{I})} \frac{E[T_{\mathcal{I}}^{\max} - \sum_{p_i \in \mathcal{I}} t_e^i]}{N_f + \sum_{p_i \in \mathcal{I}} N_i} \approx \max_{\pi(\mathcal{I})} \frac{E[T_{\mathcal{I}}^{\max}]}{N_f + \sum_{p_i \in \mathcal{I}} N_i}. \tag{4.6}$$

To compute N_f in (4.6), note that flooding packets include the RREQ packets and the route reply (RREP) packets. The number of RREQ packet equals the total number of nodes N in a fully connected network. The number of RREP packets is determined by the hop count of the path used by the RREP message. Both the numbers of RREQ and RREP packets are not related to path cache $\mathcal{I}$. In large scale MANETs, in general, the number of RREQ packets is much larger than the number of RREP packets. To simplify the analytical model, we assume that the expected number of flooding packets can be approximated as a constant during a repair cycle, and we further use the total number of nodes N in the network to approximate the total number of flooding packets. With this approximation, the objective function becomes

$$\max_{\pi(\mathcal{I})} \frac{E[T_{\mathcal{I}}^{\max}]}{N + \sum\limits_{p_i \in \mathcal{I}} N_i}. \tag{4.7}$$

Hence, only the $E[T_{\mathcal{I}}^{\max}]$ and $\sum\limits_{p_i \in \mathcal{I}} N_i$ parts in (4.7) can be optimized through the control of the paths cached in $\mathcal{I}$ and the sequence of cached path exploration. We next identify their relationship with the characteristics of cached paths.

Relationship between $E[T_{\mathcal{I}}^{\max}]$ and $\mathcal{I}$

Denote T_i as the path lifetime of the ith path (i.e. p_i) in $\widetilde{\mathcal{I}} = \mathcal{I} \cup p_0$. Note that all the cached paths in $\widetilde{\mathcal{I}}$ are node-disjoint. Based on rigorous theoretical analysis and simulation validation in [57], under the SMS mobility model, the lifetime of a path can be approximated as an exponentially distributed random variable with parameter

$$\lambda = \frac{n\bar{v}}{R}, \tag{4.8}$$

where n is the hop count of the path, $\bar{v}$ is the average node speed, and R is the node radio coverage radius. Therefore, T_i is continuous, independent and exponentially distributed with a parameter λ_i. Using this property, the relationship between $E[T_{\mathcal{I}}^{\max}]$ and $\mathcal{I}$ can be identified as follows.

Lemma 3. [1] *The expected value of* $T_{\mathcal{I}}^{\max}$ *is given by*

$$\begin{aligned} E[T_{\mathcal{I}}^{\max}] &= E\left[\max_{p_i \in \widetilde{\mathcal{I}}}\{T_i\}\right] \\ &= \sum_{p_i \in \widetilde{\mathcal{I}}} \frac{1}{\lambda_i} - \sum_{\substack{i<j \\ p_i, p_j \in \widetilde{\mathcal{I}}}} \frac{1}{\lambda_i + \lambda_j} + \sum_{\substack{i<j<k \\ p_i, p_j, p_k \in \widetilde{\mathcal{I}}}} \frac{1}{\lambda_i + \lambda_j + \lambda_k} - \ldots + (-1)^{|\widetilde{\mathcal{I}}|-1} \frac{1}{\sum\limits_{p_i \in \widetilde{\mathcal{I}}} \lambda_i}. \end{aligned} \tag{4.9}$$

[1] Here, we implicitly assume that a broken route cannot be locally recovered.

Proof. Note that $T_{\mathcal{I}}^{\max}$ is the duration of an inter-flooding period minus the duration for a flooding-based route discovery. As shown in Figure 4.1, a $T_{\mathcal{I}}^{\max}$ duration starts at the end of a flooding-based route discovery process and lasts until all paths in the path cache $\mathcal{I}$ have been explored and turned out broken, which ends the repair cycle by triggering the next flooding-based route discovery. Hence, the duration of $T_{\mathcal{I}}^{\max}$ is the maximum lifetime among all the paths in $\widetilde{\mathcal{I}}$, which can be expressed as

$$T_{\mathcal{I}}^{\max} = \max_{p_i \in \widetilde{\mathcal{I}}} \{T_i\}. \tag{4.10}$$

Since T_i is independent and exponentially distributed with parameter λ_i,

$$P\{T_{\mathcal{I}}^{\max} \leq t\} = P\{T_i \leq t, \forall p_i \in \widetilde{\mathcal{I}}\} = \prod_{p_i \in \widetilde{\mathcal{I}}} P\{T_i \leq t\} = \prod_{p_i \in \widetilde{\mathcal{I}}} (1 - e^{-\lambda_i t}). \tag{4.11}$$

Therefore, we have

$$\begin{aligned} P\{T_{\mathcal{I}}^{\max} > t\} &= 1 - P\{T_{\mathcal{I}}^{\max} \leq t\} \\ &= 1 - \prod_{p_i \in \widetilde{\mathcal{I}}} (1 - e^{-\lambda_i t}) \\ &= \sum_{p_i \in \widetilde{\mathcal{I}}} e^{-\lambda_i t} - \sum_{\substack{i<j \\ p_i, p_j \in \widetilde{\mathcal{I}}}} e^{-(\lambda_i + \lambda_j)t} + \sum_{\substack{i<j<k \\ p_i, p_j, p_k \in \widetilde{\mathcal{I}}}} e^{-(\lambda_i + \lambda_j + \lambda_k)t} - \ldots + (-1)^{(|\widetilde{\mathcal{I}}|-1)} e^{-\sum\limits_{p_i \in \widetilde{\mathcal{I}}} \lambda_i t}. \end{aligned} \tag{4.12}$$

Notice that $T_i \geq 0$ for any $p_i \in \widetilde{\mathcal{I}}$. It follows that

$$\begin{aligned} E\{T_{\mathcal{I}}^{\max}\} &= \int_0^{\infty} P\{T_{\mathcal{I}}^{\max} > t\} dt \\ &= \sum_{p_i \in \widetilde{\mathcal{I}}} \frac{1}{\lambda_i} - \sum_{\substack{i<j \\ p_i, p_j \in \widetilde{\mathcal{I}}}} \frac{1}{\lambda_i + \lambda_j} + \sum_{\substack{i<j<k \\ p_i, p_j, p_k \in \widetilde{\mathcal{I}}}} \frac{1}{\lambda_i + \lambda_j + \lambda_k} - \ldots + (-1)^{|\widetilde{\mathcal{I}}|-1} \frac{1}{\sum\limits_{p_i \in \widetilde{\mathcal{I}}} \lambda_i}. \end{aligned} \tag{4.13}$$

□

Relationship between N_i and $\mathcal{I}$

We next compute the relationship between the $\sum\limits_{p_i \in \mathcal{I}} N_i$ in (4.7) and the path cache $\mathcal{I}$. When exploring the ith cached path p_i in $\mathcal{I}$, there are two possible outcomes: p_i is still valid or p_i is already broken. When p_i is still valid, the number of overhead packets for exploring p_i is $2n_i$, where n_i is the hop count of p_i. When p_i is broken, the number of overhead packets can be calculated as follows. Assume that the broken link is uniformly distributed among the n_i

links. When there is a link break, the expected number of the overhead packets for the ith path is $2\frac{1+n_i}{2} - 1 = n_i$. Hence, the expected number of overhead packets for exploring the ith path is

$$2n_i a_i + n_i(1 - a_i) = n_i(1 + a_i), \tag{4.14}$$

where a_i is the probability that the ith path is still alive. Notice that the path lifetime is independent and exponentially distributed with parameters $\lambda_i = \frac{n_i v}{R}$. As the ith path in $\mathcal{I}$, p_i is explored only when paths $p_0, p_1, ..., p_{i-1}$ are all broken. Hence, based on (4.8), the probability that p_i is still valid when it is explored is

$$\begin{aligned} a_i &= P\{T_j \leq T_i, \forall 0 \leq j \leq i-1\} \\ &= \prod_{j=0}^{i-1} P\{T_j \leq T_i\} \\ &= \prod_{j=0}^{i-1} \frac{\lambda_j}{\lambda_j + \lambda_i} \\ &= \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}. \end{aligned} \tag{4.15}$$

Therefore, (4.7) can be derived to

$$\max_{\pi(\mathcal{I})} \frac{E[T_{\mathcal{I}}^{\max}]}{N + \sum_{p_i \in \mathcal{I}} [n_i(1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i})]}, \tag{4.16}$$

where $E[T_{\mathcal{I}}^{\max}]$'s expression is defined in (4.9).

Expression (4.16) is essentially what we are looking for: an objective function in terms of the characteristics of cached paths. In the next section, we design an optimal cache policy, which controls the paths in $\mathcal{I}$ to always have the optimal characteristics that maximize the objective function.

4.5 Analysis of Optimal Cache Policy

To find the optimal cache policy to maximize the objective function in (4.16), we take a three-step approach. First, given that $\mathcal{I}$ only caches K paths, we show how to select the best K paths to be cached in $\mathcal{I}$. Second, we show what is the optimal sequence for exploring the paths in $\mathcal{I}$ to repair routes. Finally, we show how to compute the optimal cache size K.

4.5.1 Optimal K Path Selection

Proposition 1. *If the size of the path cache $\mathcal{I}$ is fixed at a constant K, (i.e. $|\mathcal{I}| = K$), the policy that caches the shortest K paths among all discovered paths is better than any other path selection policies.*

Proof. Define $\mathcal{D}$ as the set of all paths discovered in a flooding-based route discovery process. Only K number of these paths are selected to be cached in $\mathcal{I}$ and later explored for route repair. We next prove that the policy that selects the shortest K paths in $\mathcal{D}$ is better than other selection policies.

Suppose $\mathcal{I}$ does not hold the K shortest paths in $\mathcal{D}$. Then, we can substitute a longer path p_l ($l \leq K$) in $\mathcal{I}$ with a shorter path p_l' in $\mathcal{D} - \mathcal{I}$.

This substitution decreases the denominator $N + \sum_{p_i \in \mathcal{I}} [n_i(1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j+n_i})]$ in (4.16). This is because by Lemma 9 in Appendix A, $f(n_i) = n_i(1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j+n_i})$ is an increasing function of n_i. Therefore, after substituting p_l with p_l', the term $n_l(1+\prod_{j=0}^{l-1} \frac{n_j}{n_j+n_l})$ decreases. For all $i < l$, $f(n_i)$ does not change and for all $i > l$, $f(n_i)$ also decreases. Therefore, the denominator decreases.

In addition, the substitution increases the numerator in (4.16). This is because the expected value of $T_{\mathcal{I}}^{\max}$ is the expected maximum lifetime among p_0 and the paths in $\mathcal{I}$ according to Lemma 3. Since stochastically a shorter path has a longer lifetime, the substitution of a shorter path into the set $\mathcal{I}$ increases the expectation of $T_{\mathcal{I}}^{\max}$.

Since the numerator increases and the denominator decreases, the objective function value increases after the substitution. Continue this substitution process until all paths in $\mathcal{D} - \mathcal{I}$ is no shorter than the paths in $\mathcal{I}$. The resultant $\mathcal{I}$ consists of the K shortest paths in $\mathcal{D}$. Hence, the policy of choosing the shortest K paths is better than other policies. □

4.5.2 Optimal Sequence of Path Exploration

Proposition 2. *Given a path cache $\mathcal{I}$, the optimal policy for using the cached paths in $\mathcal{I}$ for route repair is to always explore the shortest path first. In other words, the paths in $\mathcal{I}$ should be sorted in an increasing order of their hop counts.*

Proof. Assume that an optimal path exploration sequence $\mathcal{S} = \{p_1, p_2, ...\}$ maximizes the objective function in (4.16). Assume that this optimal sequence is not the shortest path first policy. Hence, in $\mathcal{S}$, there must exist two path p_l and p_{l+1} that are explored consecutively

while their path lengths satisfy $n_l > n_{l+1}$. We next show that the value of the objective function in (4.16) increases if we switch the order of p_l and p_{l+1} in $\mathcal{S}$ to get a new sequence $\mathcal{S}' = \{p_1, \cdots, p_{l-1}, p_{l+1}, p_l, p_{l+2}, \cdots\}$

The numerator $T_I^{\max}$ in (4.16) is the same for both $\mathcal{S}$ and $\mathcal{S}'$. The denominator of (4.16) is

$$\begin{aligned} g(\mathcal{S}) =& C + n_l(1 + \frac{n_0}{n_0+n_l}\frac{n_1}{n_1+n_l}\cdots\frac{n_{l-1}}{n_{l-1}+n_l}) \\ &+ n_{l+1}(1 + \frac{n_0}{n_0+n_{l+1}}\frac{n_1}{n_1+n_{l+1}}\cdots\frac{n_l}{n_l+n_{l+1}}) \end{aligned} \tag{4.17}$$

for $\mathcal{S}$ and

$$\begin{aligned} g(\mathcal{S}') =& C + n_{l+1}(1 + \frac{n_0}{n_0+n_{l+1}}\frac{n_1}{n_1+n_{l+1}}\cdots\frac{n_{l-1}}{n_{l-1}+n_{l+1}}) \\ &+ n_l(1 + \frac{n_0}{n_0+n_l}\frac{n_1}{n_1+n_l}\cdots\frac{n_{l-1}}{n_{l-1}+n_l}\frac{n_{l+1}}{n_{l+1}+n_l}) \end{aligned} \tag{4.18}$$

for $\mathcal{S}'$. Here, $C = N + \sum_{p_i \in \mathcal{I}, i\neq l, i\neq l+1} [n_i(1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j+n_i})]$ captures the terms that are the same for both $\mathcal{S}$ and $\mathcal{S}'$.

If $g(\mathcal{S}) - g(\mathcal{S}') > 0$, the switch of n_l with n_{l+1} increases objective function in (4.16). Note that:

$$\begin{aligned} g(\mathcal{S}) - g(\mathcal{S}') =& \frac{n_0\cdots n_{l-1}n_l}{(n_0+n_l)\cdots(n_{l-1}+n_l)} + \frac{n_0\cdots n_l n_{l+1}}{(n_0+n_{l+1})\cdots(n_l+n_{l+1})} \\ &- \frac{n_0\cdots n_{l-1}n_{l+1}}{(n_0+n_{l+1})\cdots(n_{l-1}+n_{l+1})} - \frac{n_0\cdots n_l n_{l+1}}{(n_0+n_l)\cdots(n_{l-1}+n_l)(n_{l+1}+n_l)}. \end{aligned} \tag{4.19}$$

Divide both sides of the above equation with a positive number

$$H = \frac{n_0\cdots n_{l-1}}{(n_0+n_l)\cdots(n_{l-1}+n_l)(n_0+n_{l+1})\cdots(n_l+n_{l+1})}, \tag{4.20}$$

we get:

$$\begin{aligned} \frac{g(\mathcal{S}) - g(\mathcal{S}')}{H} =& n_l(n_0+n_{l+1})\cdots(n_l+n_{l+1}) + n_l n_{l+1}(n_0+n_l)\cdots(n_{l-1}+n_l) \\ &- n_{l+1}(n_0+n_l)\cdots(n_{l-1}+n_l)(n_{l+1}+n_l) - n_l n_{l+1}(n_0+n_{l+1})\cdots(n_{l-1}+n_{l+1}) \\ =& (n_l - n_{l+1})\sum_{j\in\mathcal{I}}\prod_{i\in\mathcal{I}, i\neq j} n_i > 0 \text{ (since } n_l > n_{l+1}). \end{aligned} \tag{4.21}$$

Hence, $g(\mathcal{S}) > g(\mathcal{S}')$. Therefore, switching n_l with n_{l+1} in $\mathcal{S}$ increases objective function in (4.16). This contradicts with the assumption that $\mathcal{S}$ is the optimal sequence for path exploration. Hence, the optimal policy must be the shortest path first policy. □

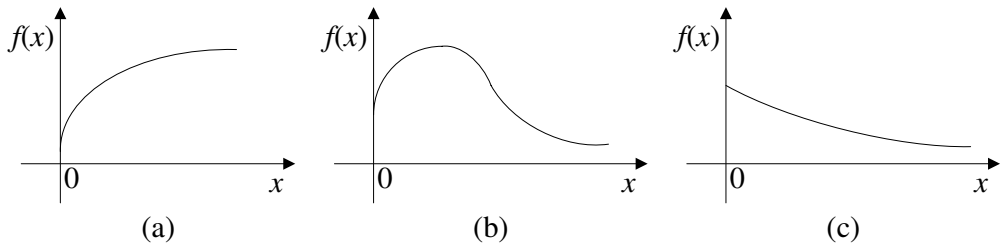


Figure 4.2: Possible Shapes of the objective function $f(x)$

4.5.3 Optimal Number of Cached Paths

By Propositions 1 and 2, given a path cache size K, i.e., $|\mathcal{I}| = K$, the optimal cache policy that maximizes the objective function in (4.16) is to cache the shortest K paths in an increasing order of their hop counts. We next show how to compute the optimal value of K.

Proposition 3. *The objective function in* (4.16) *with respect to* $|\mathcal{I}|$ *has the following three possible shapes.*

1. *The objective function is concave and monotonically non-decreasing;*
2. *The first section of the objective function is concave and non-decreasing. Then, the objective function turns to monotonically non-increasing;*
3. *The objective function is monotonically non-increasing.*

The above three shapes are shown in Figure 4.2.

Proof. Define $K := |\mathcal{I}|$. Let $\bar{h}(K)$ and $\bar{g}(K)$ be the numerator and denominator of the objective function in (4.16), respectively. Hence, the objective function becomes $\bar{f}(K) = \frac{\bar{h}(K)}{\bar{g}(K)}$. By Lemma 10 in Appendix B, $\bar{h}(K)$ is a positive discrete concave function. Hence, we can construct $\bar{h}(K)$'s continuously differentiable counterpart function $h(x), x \geq 1$ such that $h(x)|_{x=K} = \bar{h}(K)$ and $h(x)$ is concave. Similarly, by Lemma 11 in Appendix C, $\bar{g}(K)$ is a positive discrete convex function. Hence, we can construct $\bar{g}(K)$'s continuously differentiable and convex counterpart function $g(x), x \geq 1$ that satisfies $g(x)|_{x=K} = \bar{g}(K)$. Then the continuous version of the objective function becomes: $f(x) = \frac{h(x)}{g(x)}$, where $f(x)$ is continuously differentiable. By studying the property of $f(x) = \frac{h(x)}{g(x)}$, we can get the property of $\bar{f}(K)$.

Note that

$$f'(x) = \frac{h'(x)g(x) - h(x)g'(x)}{[g(x)]^2} \tag{4.22}$$

and

$$f''(x) = \frac{-2g'(x)}{g(x)} f'(x) + \frac{h''(x)g(x) - h(x)g''(x)}{[g(x)]^2}. \tag{4.23}$$

Since $h''(x) < 0, g''(x) > 0, h'(x) > 0, g'(x) > 0, h(x) > 0$ and $g(x) > 0$, we get that when $f'(x) > 0$, we always have $f''(x) < 0$. This means that at the increasing section of $f(x)$, $f(x)$ has a concave shape. Since $f(x)$ is a continuously differentiable function, if at any point it turns from increasing to decreasing, its concave increasing part can only be followed by a concave decreasing part. In addition, whenever $f(x)$ starts decreasing, $f(x)$ can never turn to increasing again. Based on the above observation, there is only three possible shapes of $f(x)$ depicted in Figure 4.2. In the first possible shape (Figure 4.2 (a)), $f(x)$ is an non-decreasing concave function. In the second possible shape (Figure 4.2 (b)), $f(x)$ is formed by two sections. In the first section, $f(x)$ is a concave function that increases with x. In the second section, $f(x)$ becomes a decreasing function. In the final possible shape (Figure 4.2 (c)), $f(x)$ is a non-increasing function. □

The correctness of Proposition 3 is validated by our numerical results in Appendix D. Proposition 3 indicates that the optimal path repair policy does not necessarily cache all paths discovered in a flooding-based route discovery. There exists a unique optimal number of paths to be cached in $\mathcal{I}$, which is denoted as K^* and is smaller or equal to the number of paths discovered in a flooding-based route discovery. If the shape of the objective function in (4.16) is similar to Figure 4.2(a), then the more path cached, the better. Hence, K^* equals the number of paths discovered in a flooding-based route discovery. If the shape of the objective function is similar to Figure 4.2(b), K^* can be computed by finding the turning point where the objective function turns from increasing to decreasing. If the shape of the objective function is similar to Figure 4.2(c), K^* equals 0, which means that no path should be cached.

4.5.4 Optimal Cache Policy

Based on our results in Propositions 1, 2 and 3, we can finally design the optimal cache policy as Algorithm 1 that picks the best set of paths and caches them in the optimal sequence in $\mathcal{I}$. The input $\mathcal{D}$ to Algorithm 1 is the set of paths discovered in a flooding-based route discovery.

Using Algorithm 1, an optimal cache-based repair scheme works as follows:

1. Determine the node-disjoint path set $\mathcal{D}$ discovered by the latest flooding;
2. Following Algorithm 1, cache the optimal set of paths in path cache $\mathcal{I}$;

Algorithm 1 The Optimal Cache Policy

Require: Discovered path set $\mathcal{D} \neq \emptyset$;
  $p_0 \leftarrow$ pop the shortest path in $\mathcal{D}$;
  $\mathcal{I} \leftarrow \emptyset$;
  Calculate the objective function value val in (4.16);
  $maxVal \leftarrow val$;
  $K^* \leftarrow 0$.
  while $\mathcal{D} \neq \emptyset$ **do**
    $p \leftarrow$ pop the shortest path in $\mathcal{D}$;
    Append p at the tail of $\mathcal{I}$;
    Calculate the objective function value val in (4.16);
    if $val \leq maxVal$ **then**
      Remove p from the tail of $\mathcal{I}$;
      return $\mathcal{I}$ and K^*;
    else
      $maxVal \leftarrow val$;
      $K^* \leftarrow K^* + 1$;
    end if
  end while
  return $\mathcal{I}$ and K^*;

3. Whenever there is a route break, explore the cached paths in $\mathcal{I}$ following the sequence that they are stored in $\mathcal{I}$. Paths that are found broken are removed from $\mathcal{I}$. Flood the whole network when $\mathcal{I}$ becomes empty.

The running time of Algorithm 1 is $O(|D|)$, where $|D|$ is the number of discovered node-disjoint paths per flooding and usually is a small number. Hence, the complexity of Algorithm 1 is fairly small.

4.6 Extension and Discussion

While the optimal cache policy in Algorithm 1 is developed with respect to the objective in (4.1), it is also applicable to other objectives as long as the objective can also make Propositions 1, 2, 3 hold. For example, consider the case of a real-time application that periodically generates delay-sensitive packets. If a real-time packet is generated when the current path to the destination is still valid, the packet can arrive at its destination with a fairly small delay. If the packet is generated when the current path to the destination is broken, the packet will experience a much longer delay and have a very high chance to miss its deadline while waiting for the route to be repaired. Hence, denoting the entire lifetime of the real-time application as T, the problem of maximizing the percentage of real-time

packets that can meet their deadline can be approximated as

$$\max \frac{\text{Data transmission time}}{T}. \tag{4.24}$$

Dividing T into repair cycles and following a method similar to the method in Sections 4.3 and 4.4, the objective function in (4.24) can be transformed to

$$\max \frac{T_d}{T_d + T_o}, \tag{4.25}$$

where T_d is the expected data packet transmission time within a repair cycle, and T_o is the expected overhead packet transmission time within a repair cycle. The objective function in (4.25) is equivalent to

$$\max \frac{T_d}{T_o} = \max \frac{T_d}{T_f + T_e} \approx \max \frac{E\{T_{\mathcal{I}}^{\max}\}}{T_f + T_e}, \tag{4.26}$$

where T_f is the expected flooding time within a repair cycle, and T_e is the expected exploration time within a repair cycle. Suppose that T_f and T_e can be computed as follows: $T_f(N)$ is an increasing function of network node count N, and $T_e = \sum_{p_i \in \mathcal{I}} n_i t_e (1 + a_i)$, where n_i is the hop count of path p_i, t_e is the expected exploration time for one link, and a_i is the probability that path p_i is still alive when it is explored. It can be shown that for the objective in (4.26), Propositions 1, 2 and 3 hold. Therefore, the optimal cache policy for the objective in (4.26) can be computed using Algorithm 1.

Another possible groups of objective functions can be formed by making the formulation of the objective functions in (4.1) and (4.24) from shape $\max A/B$ to $\max(A - B)$. It is very easy to go through similar analysis in the previous sections to show that Propositions 1, 2 and 3 also hold for these objective functions.

4.7 Simulation Results

We conduct simulations to validate our work. We first implement our optimal route repair scheme in the original DSR protocol, and then run simulations in NS 2.32 [59] to prove the advantages of the proposed optimal cache policy. In our simulations, we compare the performance of the optimal cache policy with that of the some heuristic policies, including the no-cache policy, 1-path policy, and 2-path policy. For the no-cache policy, no path is cached for path repair. For the 1-path policy, at most 1 path is cached for path repair. For the 2-path policy, at most 2 paths are cached for path repair.

In the simulation, 200 mobile nodes are uniformly distributed in a 1000×1000 m^2 rectangular region. The transmission range is 250 m, the interference range is 550 m, and the channel

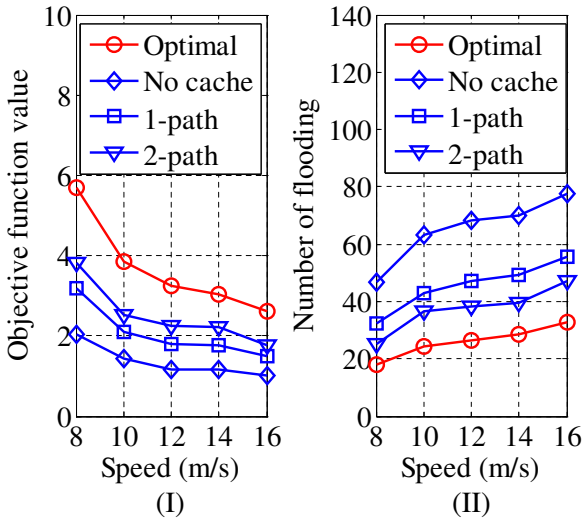


Figure 4.3: Objective function value and number of flooding

bandwidth is 1 Mb/s. We measure the performance of a single CBR flow between a randomly selected pair of source node and destination node. The packet size of the CBR flow is 512 bytes, and the packet interval is 0.05 s. For the mobility of nodes, we choose the SMS mobility model [70] due to its capability to capture realistic characteristics of node movements. The mean pause time of each node is 0 s. The minimum number of phase steps is 6. The maximum number of phase steps is 30. The unit time slot is 1 s. The memory factor parameter is 0.5. The average speed of each node ranges from 8 m/s to 16 m/s. For each average speed, we do 20 runs, and each run lasts 1000s.

We compute the objective function in (4.1) in the simulations and show the result in Figure 4.3 (I). As shown in the figure, the value of the objective function under the optimal policy is larger than that of other heuristic schemes. The number of flooding-based discovery in the simulations is shown in Figure 4.3 (II). It can be seen that the optimal policy drastically reduces the frequency of flooding comparing to other heuristic schemes. We also show the average throughput and packet loss rate in Figure 4.4 (I) and (II), respectively. It can be seen that the optimal policy outperforms other heuristic schemes in terms of both throughput and packet loss rate.

4.8 Conclusions and Future Work

While there have already existed some routing protocols aiming at optimizing cache-based route repair in MANETs, their methods are mainly heuristics and do not provide provably optimal routing policies. This chapter studies the optimal cache-based path repair problem in a rigorous mathematical approach. We focus on the problem of how to use cached node-

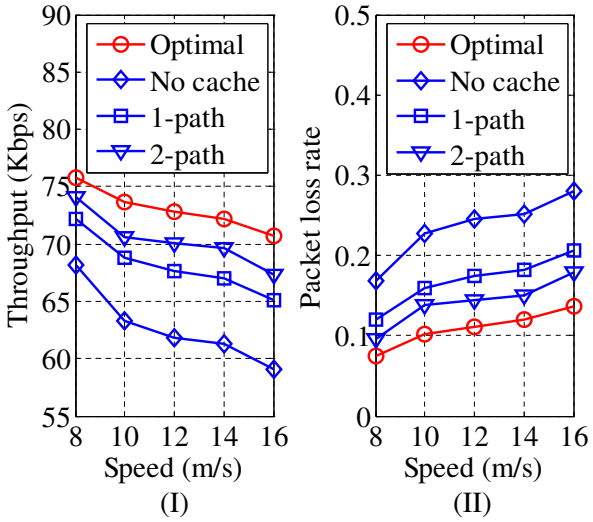


Figure 4.4: Throughput and packet loss rate

disjoint paths to reduce the flooding frequency while maintaining reasonable control packet overhead. An optimal path cache policy is proposed and its correctness is proved. Its performance advantages are validated by simulations. We have also shown that our mathematical approach can be applied to a wide range of optimization goals for cache-based route repair. In the future, we plan to apply this approach to various objectives for different application scenarios.

Chapter 5

Optimal Secondary User Node Placement in Cognitive Radio Networks

5.1 Introduction

5.1.1 Motivation

Of late, cognitive radio networks (CRNs) have received lots of attention due to its potential to improve wireless spectrum efficiency. Extensive research efforts have been dedicated on understanding the fundamental properties and performance limits of this new type of networks. Among these works, many of them are on the capacity analysis or throughput maximization. While their results help us to understand what the limit of spectrum utilization efficiency is, they cannot tell us what kind of applications a CRN can support. It is still unclear whether a CRN can support delay-sensitive applications. If it can, what is the performance limit of delay-sensitive traffic transmitted on a CRN? In the current literature, delay modeling and analysis has received less attention yet has significant impact on the service qualities of applications in CRNs. This motivates us to study the limit of delay performance of CRNs.

In a wireless network, packet delay is the aggregation of the following parts over multiple relaying hops: propagation delay, transmission delay, queueing delay, and packet processing delay. The propagation delay is determined by the physical property of wireless medium, e.g., the propagation speed of electromagnetic waves. Transmission delay is determined by the wireless channel capacity. Queueing delay is determined by the traffic load of the network and the underlying scheduling algorithm. Packet processing delay captures the time used to process the packet at current node. Due to the additive property of delay, these delay

components can be decoupled and many existing works have provide thorough analytical works for each component's computation in traditional wireless networks.

While most of these general delay analytical works can be applied in CRNs, existing results for transmission delay, however, may not hold in CRNs. Different from other wireless networks, in CRNs, the capacity of the wireless channel between a pair of secondary nodes are not only determined by secondary user (SU) nodes' own choice of transmission power and modulation scheme, but is also related to the activities of primary users (PUs). When two SU nodes are within each other's communication range, the communication link between them is not guaranteed because of the presence of active PUs in the neighborhood. The SU traffic has to wait until PU traffic vacates the channel. The existence of PU traffic in a SU network essentially reduces the link capacity between SU nodes and, hence, introduces additional PU-induced transmission delay.

Based on the above observation, in this chapter, we decouple this unique PU-induced transmission delay in CRNs from other delay components and focus on modeling and analyzing this unique delay component in CRNs. We capture this unique PU-induced delay component by defining a metric called *information propagation speed (IPS)*. IPS is the speed that a single small piece of SU information can travel through a CRN under the influence of PU traffic. The expected PU-induced transmission delay for two nodes equals the distance between these two nodes divided by the IPS.

In our work about IPS, we have made the following discoveries and contributions.

1. We have found out that IPS is not only related to the PU traffic pattern, but also is related to the placement of SUs. There exists trade-off between how to place or choose relaying SU nodes for a SU flow to maximize the IPS of the SU flow.

2. We established a model of IPS in CRNs and categorized IPS maximization problems into two cases. The first case, named the maximum network IPS problem, maximizes the IPS across a network topology over an infinite plane. The second case, named the maximum flow IPS problem, maximizes the IPS between a given pair of source and destination nodes separated by a fixed distance. We reveal how the maximum network IPS and the maximum flow IPS are determined by the PU activity level and the positions of SU relay nodes.

3. We have designed numerical methods to compute the two maximum IPSs and built optimal relay placement strategies to realize these two maximum IPSs under different PU activity levels. The correctness of our analytical results is validated by extensive simulations and numerical experiments.

4. We have shown that our IPS study can be used as a valuable benchmark for network design. The SU delay estimation formula can be use to predict the expected delay for a given SU path. The IPS bounds can be used to check whether certain delay-sensitive

traffic is supportable in a particular network setting. The optimal node placement settings that can achieve the maximum IPS can also be used as useful design guidelines in CRN planning. Also, optimal node placement results can be used to choose next hop relay nodes in delay-sensitive routing.

The rest of this chapter is organized as follows. The network model and IPS model are presented in Section 5.2 and 5.3, respectively. The analyses for the network and flow IPS are in Section 5.4 and 5.5, respectively. The applications of our analytical work are discussed in Section 5.6. Our analytical results are validated by simulations and numerical experiments in Section 6.7. Finally, Section 5.8 concludes the chapter.

5.1.2 Related Work

In the current literature, there are three bodies of work related to our work. The first body of related work [64, 48] is the scaling law capacity analysis of CRNs. Our work is different from this body of work in two folds. First, instead of focusing on scaling law capacity bounds which are asymptotic, we derive tighter delay bounds which are achievable. Second, both papers assume PU channel availability is static. This assumption is too restrictive, since PU traffic may be dynamic and the same PU channel may experience multiple busy and idle periods. Our analysis is applicable to both static and dynamic PU channel availability cases, and captures the delay caused by the temporal dynamics of PU channel availability. In [64], the authors investigate a CRN where the PU network is sparser than the SU network. Their results show that both PU and SU networks can achieve the same throughput scaling law as a stand-alone wireless network. In contrast to [64], the authors of [48] study a more general case where available bands are heterogeneous and the SU network is not necessarily denser than the PU network.

The second body of related work [9, 56, 33] is on delay optimization or analysis in single hop CRNs. While the analysis in [56, 33] does not consider the impact of SU one-hop distance on SU traffic delay, we characterize the relationship between delay and SU one-hop distance. Also, different from the single hop analysis [9, 56, 33], we are interested in delay analysis in multi-hop multichannel CRNs. In [9], the authors design and analyze a class of threshold-based access policies to minimize average delay while keeping the collision probability to PU packets below a threshold in a single hop single channel CRN. In [56], the authors use Poisson driven stochastic differential equations to analyze the SU queueing delay in a single hop multichannel CRN. In [33], the authors use a large deviation approach to characterize the relationship between SU traffic load and the SU queue distribution in a single hop multichannel CRN.

The third body of related research [71, 62, 26, 27, 3] is on the IPS in multi-hop wireless networks, which are single-tier networks. Their results cannot be applied to CRNs due to CRNs' unique two-tier architecture, where the delay in a CRN not only is related to

the settings of the CRN itself but also the traffic activities of PUs. Our work provides fundamental understanding on the delay in multihop multichannel CRNs. The authors of [71, 62] study the IPS in general wireless networks. They focus on the fundamental tradeoff between limited channel capacity and hop distance per link. In [71], the authors show that a packet can be disseminated at a constant upper bound under some specific conditions. In [62], the authors derive tighter packet propagation speed bounds for both broadcast and unicast communications. In [26], the authors derive the packet propagation speed upper bound for opportunistic routing, whose major delay is caused by retransmission. In contrast to [26], authors of [27] derive a generic theoretical upper bound of IPS in a large scale but finite mobile and delay tolerant network, where the major delay is caused by mobility and intermittent connectivity. In [3], the authors investigate the IPS in bidirectional vehicular delay tolerant networks, and focus on delay caused by waiting for next relaying vehicle. Both [27] and [3] quantify how mobility helps information propagation in the network.

To the best of our knowledge, the closest related work to our research is [54], which studies the delay distribution and limits achieved by SU node mobility in a mobile single channel CRN. In their work, they assume static PU traffic, and investigate the impact of SU node movement on SU traffic delay. In our work, we focus on the delay caused by dynamic PU traffic.

5.2 Network Model

In a CRN, PUs have a higher priority to access the channel than SUs. When a PU node is transmitting over the channel, the SU node has to wait until the PU node finishes its own transmission and vacates the channel. When transmitting its traffic, the SU node also guarantees that it will not interfere with other nearby PU receivers. To minimize the impact and modification to existing PU infrastructures, we do not assume that PU nodes will relay SU traffic. We model a CRN as a network formed by SU nodes and assume it is overlaid by a PU network in an infinite two dimensional region. The location distribution of PU nodes is assumed to be a two-dimensional Poisson point process. We assume that there are K channels and an active PU or SU only uses one channel. Based on the measurement study of realistic wireless PU activities in cellular networks [61], the PU traffic can be accurately modeled as a Poisson arrival process with the mean arrival rate λ_P per unit area, while the service time of PU is usually not exponential. [1] Since PU nodes may also relay traffic from other PU nodes, the aggregate traffic may not be Poisson in the strict sense. To simplify the analysis and get insight in the problem, we approximate the aggregate traffic as Poisson. As

[1] In our preliminary work [19], we study a loose upper bound on network IPS based on less general and less realistic assumptions such as the PU channel selection is not adaptable, PU service time is exponential, and SU transmission power is not adaptable. In this work, our analysis is much more realistic and general and we derive the exact maximum network IPS and flow IPS. The mathematical analysis in the preliminary work is very different from this chapter.

shown in [53, 72], this approximation is fairly accurate.

We assume half-duplex communications between the SU transmitter and receiver (e.g., the SU receiver sends ACK back for received data). Hence, both the SU transmitter and the SU receiver need to avoid interfering with PU receivers. Denote d_s as the sensing radius of SU for PU receivers [2] and let $U(d)$ be the union of the sensing regions of a pair of SU transmitter and receiver that are d distance apart. The shape of $U(d)$ is shown by the grey region in Figure 5.1 (II). When there are no active PU receivers within $U(d)$, the SU transmitter and receiver can communicate and we call that the SU link between the SU transmitter and receiver is feasible. Note that here we implicitly assume that SUs can cancel the interference from PU transmitters to the SU receiver through interference cancelation schemes. An example interference cancelation scheme based on multiple access channel and superposition coding is proposed in [45].

It is important to note that given cognitive radio's capability to adapt transmission power and hence its potential interference to PUs, the sensing radius of SU should not be treated as a fixed value. In our model, we assume that a SU controls its transmission power so that its communication range equals its distance d to its receiver. In this way, the interference to PUs is minimized. Under this optimal power control policy, we get:

$$\frac{P_t}{d^\alpha} = T_r, \tag{5.1}$$

where $\alpha \geq 2$ is the attenuation exponent, P_t is the transmission power of the SU transmitter and T_r is the receiver's sensitivity level. Assume that when the signal of a SU exceeds the interference threshold T_s at a PU receiver, the PU receiver is interfered by the SU transmission. Therefore, the SU's sensing radius d_s can be expressed as:

$$\frac{P_t}{d_s^\alpha} = T_s. \tag{5.2}$$

Combining (5.1) and (5.2), we have

$$d_s = \left(\frac{T_r}{T_s}\right)^{\frac{1}{\alpha}} d = C_1 d, \tag{5.3}$$

where $C_1 = \left(\frac{T_r}{T_s}\right)^{\frac{1}{\alpha}}$. The above equation shows that the sensing radius d_s is proportional to the SU communication range that equals the distance d between the SU transmitter and SU receiver under the optimal power control policy.

[2] PU receiver detection can be realized by exploiting the feedback mechanisms in two-way PU communications as shown by [66, 69].

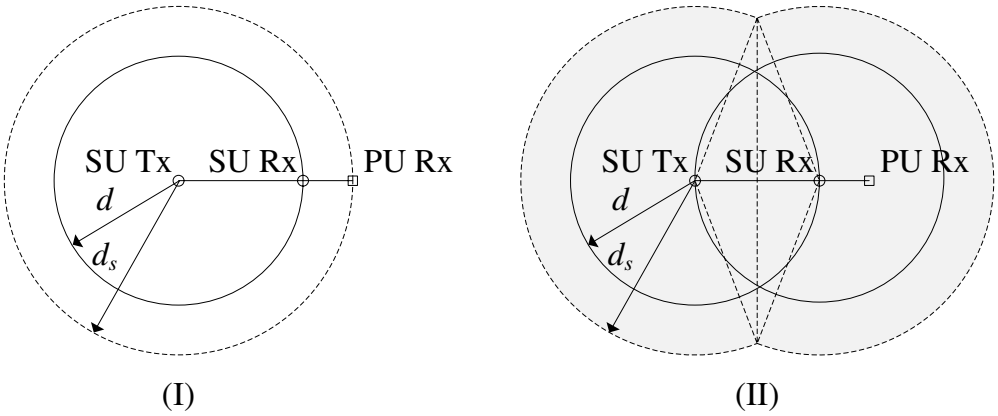


Figure 5.1: One hop sensing region

5.3 Problem Formulation

Based on the assumptions in the previous section, we can model IPS, denoted as v, in a multihop CRN as:

$$v = \frac{D}{\tau(D)} = \frac{\sum_i P(d_i)}{\sum_i \tau(d_i)}, \tag{5.4}$$

where D is the distance between a pair of SU source node and destination node, and $\tau(D)$ is the expected information propagation delay over this distance D. d_i is the transmission distance of the ith hop, and $P(d_i)$ is the projection of d_i on the straight line from the source to the destination as shown in Figure 5.2. $\tau(d_i)$ is the expected information propagation delay over the ith hop. By observing (5.4), we could see a trade-off exists in the setting of d_i and v. Clearly, when SU relay nodes are placed on the straight line between the SU source and destination, the corresponding IPS is larger than the IPS for other SU relay node deployment. In this case, while a larger d_i increases the numerator in (5.4), it also increases $\tau(d_i)$ since it has a large $U(d_i)$ and a SU link cannot be used if there is any active PU receiver in the area of $U(d_i)$. The objective of our research, hence, can be formulated as

$$\begin{aligned} &\max_{\mathbf{d}} \frac{\sum_i P(d_i)}{\sum_i \tau(d_i)}, \\ &s.t.\ 0 < d_i \leq r_c, \forall d_i \in \mathbf{d}. \end{aligned} \tag{5.5}$$

where r_c is the SU communication range constrained by the SU radio's maximum transmission power and $\mathbf{d}$ is the set of all hops.

In the following two sections, we study two different cases for solving the above optimization problem by optimally setting $\mathbf{d}$. In Section 5.4, we maximize IPS while assuming that all the SU nodes can be flexibly placed in an infinite plane. We call it the maximum network IPS. In Section 5.5, we maximize the IPS for a specific flow with fixed source and destination locations. We call it the maximum flow IPS.

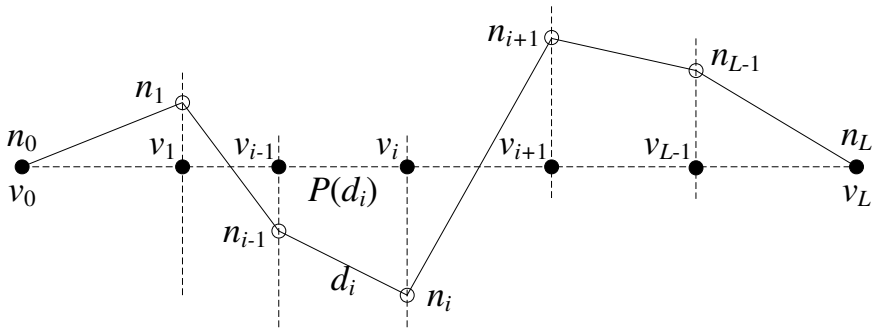


Figure 5.2: One-hop progress distance

5.4 Network IPS

Based on (5.5), when a CRN is over an infinite plane and all nodes' locations are flexible, the maximum network IPS, denoted as V_n, is:

$$\begin{aligned} V_n &= \max_{\mathbf{d}} \frac{\sum_i P(d_i)}{\sum_i \tau(d_i)} \\ &= \frac{P(d^*)}{\tau(d^*)}, \end{aligned} \tag{5.6}$$

where d^* is the optimal one-hop distance that results in the maximum IPS in that hop. The second equality in (5.6) is derived by noticing that

$$\frac{\sum_i P(d_i)}{\sum_i \tau(d_i)} \leq \max_i \frac{P(d_i)}{\tau(d_i)}. \tag{5.7}$$

Equation (5.6) means that the maximum network IPS is achieved when every hop has the same optimal one-hop distance d^*. Therefore, the problem for deriving the maximum IPS in a multi-hop CRN has been transformed to the problem of finding the maximum one-hop IPS. In the reminder of this section, we derive the expected one-hop delay function, and then study its monotonicity and convexity, based on which we derive the optimal one-hop distance d^* that achieves the maximum network IPS.

5.4.1 One-hop Delay Function

The expected one-hop delay over a transmission distance d can be expressed as

$$\tau(d) = E\{T\} + \tau_0, \tag{5.8}$$

where $E\{T\}$ is the mean one-hop delay induced by waiting for PU traffic to vacate a channel. τ_0 is the sum of other approximately constant delay components, such as channel sensing delay for determining the channel availability.

While τ_0 is fixed given a particular cognitive radio design and setting, the value of $E\{T\}$ is related to the level of PU traffic load and equals the time that it takes for a channel to be vacated by PUs when a SU is ready to transmit a packet. Note that in Section 5.2, we have pointed out that measurement results in [61] show that realistic PU traffic follows a Poisson arrival process and has complex service time distribution. Hence, we can treat PUs as a high priority flow, SUs as a low priority flow and the K channels as K servers in a two-priority preemptive M/G/K queuing model. Under this model, $E\{T\}$ is equivalent to the queuing delay of the low priority flow when the packet service time of the low priority flow approaches 0.

The analytical work of the two-priority preemptive M/G/K queue in [5] shows that the queuing delay of the low priority flow can be approximated as:

$$delay_1 = \begin{cases} \frac{1}{2K\rho_1}[\frac{s_1\rho_1+s_2\rho_2}{1-\rho_1-\rho_2}\frac{(\rho_1+\rho_2)^K+(\rho_1+\rho_2)}{2} - \frac{s_2\rho_2}{1-\rho_2}\frac{\rho_2^K+\rho_2}{2}], \\ \rho_2 > 0.7, \rho_1+\rho_2 > 0.7; \\ \frac{1}{2K\rho_1}[\frac{s_1\rho_1+s_2\rho_2}{1-\rho_1-\rho_2}\frac{(\rho_1+\rho_2)^K+(\rho_1+\rho_2)}{2} - \frac{s_2\rho_2}{1-\rho_2}\rho_2^{\frac{K+1}{2}}], \\ \rho_2 < 0.7, \rho_1+\rho_2 > 0.7; \\ \frac{1}{2K\rho_1}[\frac{s_1\rho_1+s_2\rho_2}{1-\rho_1-\rho_2}(\rho_1+\rho_2)^{\frac{K+1}{2}} - \frac{s_2\rho_2}{1-\rho_2}\rho_2^{\frac{K+1}{2}}], \\ \rho_2 < 0.7, \rho_1+\rho_2 < 0.7. \end{cases} \tag{5.9}$$

Here, subscript 1 is for the low priority flow, and subscript 2 is for the high priority flow. We have

$$s_i := (1 + C_{B_i}^2)\bar{X}_i, \tag{5.10}$$

$$C_{B_i}^2 := \frac{\sigma_i^2}{\bar{X}_i^2}, \tag{5.11}$$

and

$$\rho_i := \frac{\lambda_i \bar{X}_i}{K}, \tag{5.12}$$

where X_i is the service time for priority i traffic, σ_i^2 is the variance of X_i, $\bar{X}_i$ is the mean value of X_i, and λ_i is the arrival rate of the priority i traffic. Therefore, assuming that the PU traffic load is reasonable (a.k.a. $\rho_2 < 0.7$), we can get $E\{T\}$ as:

$$\begin{aligned} E\{T\} &= \lim_{\rho_1 \to 0, s_1 \to 0} delay_1 \\ &= \frac{(K+1)s_2\rho_2^{\frac{K+1}{2}}}{4K(1-\rho_2)} + \frac{s_2\rho_2^{\frac{K+3}{2}}}{2K(1-\rho_2)^2}. \end{aligned} \tag{5.13}$$

The reason that we assume $\rho_2 < 0.7$ is because a CRN is usually used in scenarios where PUs have low utilization of licensed spectrum bands. Hence, the assumption that the PU traffic

load is light to medium is valid in our application scenario. The PU traffic load ρ_2 in (5.13) can be computed as follows. Note that within a unit area, the PU traffic is a Poisson arrival process with parameter λ_P. Hence, the aggregate PU traffic arrival rate within region $U(d)$ is Poisson with parameter $\lambda_a = \lambda_P A(d)$, where $A(d)$ is the area of region $U(d)$ as shown in Figure 5.1. By (5.3) and Figure 5.1 (II), it can be shown that

$$A(d) = Cd^2, \tag{5.14}$$

where

$$C = 2\pi C_1^2 - 2C_1^2 \cos^{-1} \frac{1}{2C_1} + \sqrt{C_1^2 - \frac{1}{4}} \tag{5.15}$$

and $C_1 = (\frac{T_r}{T_s})^{\frac{1}{\alpha}}$. Therefore, we have

$$\rho_2 = \frac{Cd^2\lambda_P}{\mu_P K} = Cd^2\rho, \tag{5.16}$$

where $\rho := \frac{\lambda_P}{\mu_P K}$ and $\mu_P := \frac{1}{X_2}$ is the PU service rate. With ρ_2 known, we can finally combine (5.8) and (5.13) to get:

$$\tau(d) = \frac{(K+1)s_2\rho_2^{\frac{K+1}{2}}}{4K(1-\rho_2)} + \frac{s_2\rho_2^{\frac{K+3}{2}}}{2K(1-\rho_2)^2} + \tau_0. \tag{5.17}$$

Given PU node density, PU traffic load, and hop distances of a particular SU path, equations (5.15), (5.16) and (6.21) can be used to estimate the expected delay on the SU path.

5.4.2 Properties of One-hop Delay Function

Next, we study two important properties of $\tau(d)$: monotonicity and convexity. We show that the one-hop delay function $\tau(d)$ is monotonically increasing and strictly convex. To simplify the mathematical derivation, we denote

$$h_1(\rho_2) = \frac{(K+1)s_2\rho_2^{\frac{K+1}{2}}}{4K(1-\rho_2)}, \tag{5.18}$$

and

$$h_2(\rho_2) = \frac{s_2\rho_2^{\frac{K+3}{2}}}{2K(1-\rho_2)^2}. \tag{5.19}$$

We can prove the following lemmas.

Lemma 4. *For* (5.18) *and* (5.19)*, it follows*

$$h_1'(\rho_2) > 0, h_2'(\rho_2) > 0, \tag{5.20}$$

and

$$h_1''(\rho_2) > 0, h_2''(\rho_2) > 0. \tag{5.21}$$

Proof. By (5.18) and (5.19), we have

$$h_1'(\rho_2) = \frac{(K+1)s_2}{4K} \frac{\frac{K+1}{2}\rho_2^{\frac{K-1}{2}}(1-\rho_2) + \rho_2^{\frac{K+1}{2}}}{(1-\rho_2)^2} > 0, \tag{5.22}$$

and

$$h_2'(\rho_2) = \frac{s_2}{2K} \frac{\frac{K+3}{2}\rho_2^{\frac{K+1}{2}}(1-\rho_2) + 2\rho_2^{\frac{K+3}{2}}}{(1-\rho_2)^3} > 0. \tag{5.23}$$

Therefore, (5.20) is true.

Next, we prove that (5.21) is true. By (5.22) and (5.23), we have

$$h_1'(\rho_2) = \frac{(K+1)s_2}{4K} \frac{\frac{K+1}{2}\rho_2^{\frac{K-1}{2}} - \frac{K-1}{2}\rho_2^{\frac{K+1}{2}}}{(1-\rho_2)^2}, \tag{5.24}$$

and

$$h_2'(\rho_2) = \frac{s_2}{2K} \frac{\frac{K+3}{2}\rho_2^{\frac{K+1}{2}} - \frac{K-1}{2}\rho_2^{\frac{K+3}{2}}}{(1-\rho_2)^3}. \tag{5.25}$$

It follows that

$$\begin{aligned} h_1''(\rho_2) = \frac{(K+1)s_2}{4K(1-\rho_2)^3}[&\frac{K^2-1}{4}(\rho_2^{\frac{K-3}{2}} - \rho_2^{\frac{K-1}{2}})(1-\rho_2) \\ &+ 2(\frac{K+1}{2}\rho_2^{\frac{K-1}{2}} - \frac{K-1}{2}\rho_2^{\frac{K+1}{2}})] > 0, \end{aligned} \tag{5.26}$$

and

$$\begin{aligned} &h_2''(\rho_2) \\ =&\frac{s_2}{2K(1-\rho_2)^4}[(\frac{K+3}{2}\frac{K+1}{2}\rho_2^{\frac{K-1}{2}} - \frac{K-1}{2}\frac{K+3}{2}\rho_2^{\frac{K+1}{2}}) \\ &(1-\rho_2) + 3(\frac{K+3}{2}\rho_2^{\frac{K+1}{2}} - \frac{K-1}{2}\rho_2^{\frac{K+3}{2}})] > 0. \end{aligned} \tag{5.27}$$

Therefore, the lemma holds. □

Lemma 5. *The function $\tau(d), 0 < d \leq r_c$ is monotonically increasing.*

Proof. By (5.16), (6.21), (5.18) and (5.19), we have

$$\tau(d) = h_1(\rho C d^2) + h_2(\rho C d^2) + \tau_0, \tag{5.28}$$

and

$$\tau'(d) = 2\rho C d[h_1'(\rho C d^2) + h_2'(\rho C d^2)]. \tag{5.29}$$

By (5.20), we have $\tau'(d) > 0$. Therefore, the lemma follows. □

Lemma 6. *The function $\tau(d), 0 \leq d \leq r_c$ is strictly convex.*

Proof. By (5.16), (6.21), (5.18) and (5.19), we have

$$\begin{aligned}\tau''(d) = {} & 2\rho C h_1'(\rho C d^2) + 4\rho^2 C^2 d^2 h_1''(\rho C d^2) \\ & + 2\rho C h_2'(\rho C d^2) + 4\rho^2 C^2 d^2 h_2''(\rho C d^2).\end{aligned} \tag{5.30}$$

By (5.20) and (5.21), we have $\tau''(d) > 0$. Hence, the lemma follows. □

5.4.3 Maximum IPS Analysis

Clearly, IPS can only be maximized when SU nodes are aligned on the straight line between the source and destination. When this happens, the one-hop progress distance along the straight line from the source to the destination equals the one hop distance, i.e., $P(d) = d$. Therefore, the optimal one-hop distance d^* in (5.6) can be computed as

$$d^* = \arg \min_{0<d\leq r_c} \{\frac{\tau(d)}{d}\}. \tag{5.31}$$

Since the physical meaning of $\frac{\tau(d)}{d}$ is the slope of the line passing through the origin and a point on the function $\tau(d)$ curve, it follows that d^* is the d value that minimizes the line slope.

Optimal One-hop Distance Analysis

By Lemma 5 and 6, $\tau(d), 0 < d \leq r_c$ is monotonically increasing and strictly convex. Hence, there are two possibilities when determining d^* as shown in Figure 5.3. Consider the tangent line of the curve $\tau(d)$ that passes the origin and touches $\tau(d)$ at a point $(d_0, \tau(d_0))$. When

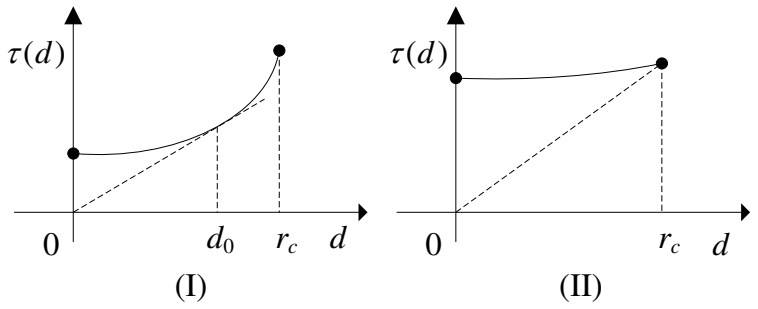


Figure 5.3: Two examples of $\tau(d)$: (I) $\exists 0 < d_0 \leq r_c,\ s.t.\ \tau'(d_0) = \frac{\tau(d_0)}{d_0}$; (II) $\nexists 0 < d_0 \leq r_c,\ s.t.\ \tau'(d_0) = \frac{\tau(d_0)}{d_0}$.

there is such a tangent line as shown in Figure 5.3 (I), we have $d^* = d_0$. When there is no such a tangent line as shown in Figure 5.3 (II), we have $d^* = r_c$. Mathematically, we have

$$d^* = \begin{cases} d_0, \text{ if } \exists 0 < d_0 \leq r_c,\ s.t.\ \tau'(d_0) = \frac{\tau(d_0)}{d_0} \\ r_c, \text{ if } \nexists 0 < d_0 \leq r_c,\ s.t.\ \tau'(d_0) = \frac{\tau(d_0)}{d_0} \end{cases} \tag{5.32}$$

To determine d^*, we need to determine if there is a real root to equation $\tau'(d) = \frac{\tau(d)}{d}, 0 < d \leq r_c$. To solve this root existence problem, we define

$$\begin{aligned} f(\rho, d) &= \tau'(d) - \frac{\tau(d)}{d} \\ &= 2Cd\rho h_1'(Cd^2\rho) + 2Cd\rho h_2'(Cd^2\rho) \\ &\quad - \frac{h_1(Cd^2\rho) + h_2(Cd^2\rho)}{d} - \frac{\tau_0}{d}, \end{aligned} \tag{5.33}$$

and study the root existence problem for equation

$$f(\rho, d) = 0, 0 < d \leq r_c. \tag{5.34}$$

Threshold Property of d^*

Next, we solve the root existence problem in (5.34) and determine d^*. Our analytical results are summarized in the following proposition.

Proposition 4. *There exists a threshold $\rho_u \in (0, \frac{1}{(Cr_c^2)})$ such that $d^* = r_c$ when $\rho < \rho_u$ and $d^* < r_c$ when $\rho > \rho_u$. This threshold ρ_u is a function of r_c satisfying $f(\rho_u, r_c) = 0$.*

Proof. By Lemma 5 and 6, $\tau(d), 0 < d \leq r_c$ is monotonically increasing and strictly convex. By (5.33), it follows that $f(\rho, d)$ is an increasing function of d. Since $\lim_{d \to 0^+} f(\rho, d) = -\infty$,

it follows that $f(\rho, d)$ increases from $-\infty$ to $f(\rho, r_c)$ when d increases from 0 to r_c. When $f(\rho, r_c) > 0$, there must exist a real root to the problem in (5.34). By (5.32), we have $d^* < r_c$. When $f(\rho, r_c) < 0$, we have $f(\rho, d) < 0, \forall 0 < d \leq r_c$, i.e., there does not exist a real root to the problem in (5.34). By (5.32), we have $d^* = r_c$.

Next, we study when $f(\rho, r_c) > 0$ and when $f(\rho, r_c) < 0$. Since r_c is fixed, the positivity of $f(\rho, r_c)$ depends on ρ and $\rho \in (0, \frac{1}{(Cr_c^2)})$. We study the positivity of $f(\rho, r_c)$ when ρ changes. By (5.33), we have

$$\begin{aligned} f(\rho, r_c) = & 2Cr_c\rho h_1'(Cr_c^2\rho) + 2Cr_c\rho h_2'(Cr_c^2\rho) \\ & - \frac{h_1(Cr_c^2\rho) + h_2(Cr_c^2\rho)}{r_c} - \frac{\tau_0}{r_c}. \end{aligned} \tag{5.35}$$

It can be shown that $\lim_{\rho \to 0^+} f(\rho, r_c) < 0$ and $\lim_{\rho \to (\frac{1}{Cr_c^2})^-} f(\rho, r_c) = \infty$. Therefore, there is at least one real root to equation

$$f(\rho, r_c) = 0, 0 < \rho < \frac{1}{Cr_c^2}. \tag{5.36}$$

In the following, we prove that there is only one real root. By (5.35), we have

$$\begin{aligned} \frac{\partial f(\rho, r_c)}{\partial \rho} = & Cr_c h_1'(Cr_c^2\rho) + Cr_c h_2'(Cr_c^2\rho) \\ & + 2\rho C^2 r_c^3 h_1''(Cr_c^2\rho) + 2\rho C^2 r_c^3 h_2''(Cr_c^2\rho). \end{aligned} \tag{5.37}$$

By (5.20) and (5.21), we have $\frac{\partial f(\rho, r_c)}{\partial \rho} > 0$, i.e., $f(\rho, r_c)$ is an increasing function with respect to ρ. Therefore, there exists only one real root to equation $f(\rho, r_c) = 0, 0 < \rho < \frac{1}{Cr_c^2}$. Denote the root as ρ_u. Recall that $\lim_{\rho \to 0^+} f(\rho, r_c) < 0$ and $\lim_{\rho \to (\frac{1}{Cr_c^2})^-} f(\rho, r_c) = \infty$. It follows that $f(\rho, r_c) > 0$ when $\rho > \rho_u$ and $f(\rho, r_c) < 0$ when $\rho < \rho_u$. Therefore, when $\rho > \rho_u$ we have $d^* < r_c$, and when $\rho < \rho_u$ we have $d^* = r_c$. □

Although it is difficult to derive a closed form expression for the threshold ρ_u and d^*, we can numerically derive it from (5.36). The physical intuition behind Proposition 4 is as follows. When the actual ρ is small, the delay of SU traffic caused by yielding to PU transmissions in the region $U(d)$ is negligible. The SU's IPS is only constrained by the maximum transmission power. Therefore, the optimal one-hop distance $d^* = r_c$. When ρ is large, the delay caused by PU transmissions in the region $U(d)$ dominates other delay components. Hence, a shorter one-hop distance d incurs a smaller $U(d)$ size, resulting in a smaller delay. Therefore, the optimal one-hop distance $d^* < r_c$.

5.5 Flow IPS

Beyond the maximum network IPS for all possible flows, we are also interested in the maximum IPS for a particular given flow, called the maximum flow IPS. Since in this case the

source-destination distance is fixed, the problem of maximizing the IPS is equivalent to minimizing the total propagation delay from the source to the destination. Therefore, for the flow IPS case, by (5.4), the maximum IPS, denoted as V_f, can be modeled as

$$V_f = \max\{\frac{D}{\sum_i \tau(d_i)}\} = \frac{D}{\min\{\sum_i \tau(d_i)\}}, \tag{5.38}$$

where $\tau(\cdot)$ is the expected one-hop propagation delay. It is clear that the total delay is minimized when all the SU nodes are placed on the straight line between the source and destination. Mathematically, the problem of optimal node placement is transformed to

$$\begin{aligned} &\min \sum_i \tau(d_i), \\ &s.t. \sum_i d_i = D. \end{aligned} \tag{5.39}$$

We decompose the above minimization problem to two subproblems: the problem of placing SU relaying nodes given a fixed number of such nodes and the problem of determining the optimal number of relaying nodes between the source and the destination. By solving these two subproblems, we show that the IPS is maximized when an optimal number of relay nodes are evenly spaced along a straight line between the source and destination.

5.5.1 Optimal Node Placement

We first study the problem of how to place relay nodes to minimize the total delay when the total number of relay nodes is given. We decompose the problem of multi-hop path delay to a series of two-hop path problems. Our analysis shows that the total delay is minimized when the inter-node distances are equal.

Lemma 7. *Consider a K-hop ($K \geq 2$) SU path between a given pair of source and destination nodes. The total expected delay from the source to the destination is minimized when all the $K-1$ relay nodes on the path are evenly spaced on the straight line from the source to the destination.*

Proof. Consider a two-hop SU path, whose source node and destination node is y distance apart and $0 < y < 2r_c$. Then, the total delay of the two-hop path is

$$\tau_2(x) = \tau(x) + \tau(y - x), \tag{5.40}$$

where either $y - r_c < x \leq r_c, r_c \leq y < 2r_c$ or $0 < x < r_c, 0 < y < r_c$. By Lemma 5 and 6, we have $\tau_2''(x) = \tau''(x) + \tau''(y - x) > 0$. Hence, $\tau_2(x)$ is strictly convex. Therefore, $\tau_2(x)$ is minimized when $\tau_2'(x) = \tau'(x) - \tau'(y - x) = 0$, i.e., $x = \frac{y}{2}$. This means that the total delay of a two-hop path is minimized when the relay node is placed in the middle point between the source node and destination node.

Next, we prove the lemma by contradiction. Given a fixed number of relay nodes, suppose that the minimum total expected delay from the source to the destination is achieved when nodes are not evenly spaced along the straight line between the source and destination. Denote such a path as P. Then, there must exist a two-hop subpath P_s on the path P such that the middle SU node of the two-hop subpath is not in the middle position between the source and destination of the subpath. By placing the middle SU node to the middle position, the expected delay on this two-hop subpath decreases, and the total expected delay of path P decreases. This contradicts the assumption that path P minimizes the total expected delay from the source to the destination. Therefore, the lemma follows. □

5.5.2 Optimal Number of Relay Nodes

Next, we determine the optimal number of relay nodes to minimize the total delay between the source and the destination. Note that to guarantee connectivity between the source and destination, there are at least $\left\lfloor \frac{D}{r_c} \right\rfloor$ SU nodes placed on the straight line between the source and destination. Denote n as the number of SU nodes to add, and $m = n + 1$ as the number of hops between the source and destination. It follows that $n \geq \left\lfloor \frac{D}{r_c} \right\rfloor$ and $m \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$. By (5.39) and Lemma 7, given a m hop SU path ($n = m - 1$ relay nodes), the minimum total delay is $t(m) = m\tau(\frac{D}{m})$. Therefore, the optimization problem in (5.39) can be transformed to

$$\begin{aligned} &\min t(m) = m\tau(\frac{D}{m}), \\ &s.t.\ m \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1, m \in Z^+. \end{aligned} \tag{5.41}$$

Consider its continuous counterpart problem

$$\begin{aligned} &\min t(x) = x\tau(\frac{D}{x}), \\ &s.t.\ x \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1, x \in R^+. \end{aligned} \tag{5.42}$$

It can be shown that $t'(x) = \tau(\frac{D}{x}) - \frac{D}{x}\tau'(\frac{D}{x})$, and

$$t''(x) = \frac{D^2}{x^3}\tau''(\frac{D}{x}) > 0. \tag{5.43}$$

By (5.43), $t(x)$ is a strictly convex function over $x \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$. There are two possibilities when solving the problem in (5.42). When $t'(\left\lfloor \frac{D}{r_c} \right\rfloor + 1) > 0$, the optimal solution is $x^* = \left\lfloor \frac{D}{r_c} \right\rfloor + 1$. When $t'(\left\lfloor \frac{D}{r_c} \right\rfloor + 1) \leq 0$, the optimal solution $x^* = x_0$, where $t'(x_0) = 0$ and $x_0 \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$.

Therefore, the optimal solution to the problem in (5.41) is as follows:

$$m^* = \begin{cases} \left\lfloor \frac{D}{r_c} \right\rfloor + 1, \text{ if } t'(\left\lfloor \frac{D}{r_c} \right\rfloor + 1) > 0, \\ \arg \min\limits_{m \in \{m_1, m_2\}} \{t(m)\}, \text{ if } t'(\left\lfloor \frac{D}{r_c} \right\rfloor + 1) \leq 0, \end{cases} \tag{5.44}$$

where $m_1 = \lfloor x^* \rfloor$, $m_2 = \lceil x^* \rceil$, and $t'(x^*) = 0$.

5.5.3 A Search Method of Calculating m^*

Note that directly computing m^* from (5.44) involves solving the equation $t'(x_0) = 0, x_0 \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$, which may be computationally intensive. This motivates us to find alternative methods to determine m^*. Since we have proved that $t(x), x \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$ is convex, $t(m), m \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$ can be either monotonically increasing, or first monotonically decreasing and then monotonically increasing. Therefore, m^* is the smallest m such that $t(m+1) > t(m)$. Mathematically,

$$\begin{aligned} m^* = \min\{m | (m+1)\tau(\frac{D}{m+1}) > m\tau(\frac{D}{m}), \\ m \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1, m \in Z^+\}. \end{aligned} \tag{5.45}$$

Based on (5.45), it is straightforward to search for m^* from $m = \lfloor \frac{D}{r_c} \rfloor + 1$.

5.5.4 A Table Look-up Method Based on Threshold Property of m^*

While it is possible to determine m^* by (5.45), the search algorithm may take many steps before finding m^*, when m^* is much larger than $\left\lfloor \frac{D}{r_c} \right\rfloor + 1$. Hence, the search algorithm may not be efficient enough for run-time computation. This motivates us to find a faster method to determine m^*.

Our basic idea is to determine m^* by considering whether adding a relay node decreases the total delay. Our analysis shows that this is determined by a threshold of PU load. By (5.45), adding a relay decreases the total delay when $(m+1)\tau(\frac{D}{m+1}) < m\tau(\frac{D}{m})$. From (5.16) and

(6.21), we define

$$\begin{aligned} g(\rho, m) &= (m+1)\tau(\frac{D}{m+1}) - m\tau(\frac{D}{m}) \\ &= (m+1)[h_1(\frac{\rho CD^2}{(m+1)^2}) + h_2(\frac{\rho CD^2}{(m+1)^2})] \\ &- m[h_1(\frac{\rho CD^2}{m^2}) + h_2(\frac{\rho CD^2}{m^2})] + \tau_0, \end{aligned} \tag{5.46}$$

When $g(\rho, m) < 0$, adding a relay node decreases the total relay. When $g(\rho, m) > 0$, adding a relay node increases the total relay. Given m, the positivity of $g(\rho, m)$ depends on ρ. Next, we study the positivity of $g(\rho, m)$ when ρ changes.

Lemma 8. *Consider a m-hop SU path, whose source-destination distance is D. There exists a $\rho_f \in (0, \frac{m^2}{CD^2})$ that is a function of m (a.k.a $\rho_f = \rho_f(m)$) and is defined by $g(\rho_f, m) = 0$. This ρ_f has the following properties.*

- *When $\rho > \rho_f$, adding a relay node and evenly spacing all relay nodes decreases the total delay.*
- *When $\rho < \rho_f$, adding extra relay nodes increases the total delay.*
- *ρ_f is monotonically increasing with respect to m.*

Proof. Note that $\lim_{\rho \to 0^+} g(\rho, m) = \tau_0 > 0$, and $\lim_{\rho \to (\frac{m^2}{CD^2})^-} g(\rho, m) = -\infty < 0$. Therefore, there is at least one real root of ρ to equation $g(\rho, m) = 0$ in the range $0 < \rho < \frac{m^2}{CD^2}$. Next, we show that there is only one real root. We prove this by showing that $g(\rho, m)$ is monotonically decreasing with respect to ρ. By (5.46), we have

$$\begin{aligned} \frac{\partial g(\rho, m)}{\partial \rho} &= \frac{CD^2}{m+1}[h_1'(\frac{\rho CD^2}{(m+1)^2}) + h_2'(\frac{\rho CD^2}{(m+1)^2})] \\ &- \frac{CD^2}{m}[h_1'(\frac{\rho CD^2}{m^2}) + h_2'(\frac{\rho CD^2}{m^2})]. \end{aligned} \tag{5.47}$$

By (5.20), we have $h_1'(\frac{\rho CD^2}{(m+1)^2}) + h_2'(\frac{\rho CD^2}{(m+1)^2}) > 0$, and

$$\begin{aligned} \frac{\partial g(\rho, m)}{\partial \rho} &< \frac{CD^2}{m}[h_1'(\frac{\rho CD^2}{(m+1)^2}) + h_2'(\frac{\rho CD^2}{(m+1)^2})] \\ &- \frac{CD^2}{m}[h_1'(\frac{\rho CD^2}{m^2}) + h_2'(\frac{\rho CD^2}{m^2})]. \end{aligned} \tag{5.48}$$

By (5.21), it follows that $h_1'(\rho_2)$ and $h_2'(\rho_2)$ are monotonically increasing. Therefore, by (5.48) we have $\frac{\partial g(\rho, m)}{\partial \rho} < 0$. Hence, there is only one real root to equation $g(\rho, m) = 0, 0 < \rho < \frac{m^2}{CD^2}$.

Denote the root as ρ_f. Obviously, ρ_f is a function of m. We denote this function as $\rho_f(m)$. Recall that $\lim_{\rho\to 0^+} g(\rho, m) > 0$, and $\lim_{\rho\to(\frac{m^2}{CD^2})^-} g(\rho, m) < 0$. Hence, we have the following conclusions. There exists a threshold value $0 < \rho_f < \frac{m^2}{CD^2}$ such that $g(\rho_f, m) = 0$. When $\rho < \rho_f$, it follows that $g(\rho, m) > 0$. By (5.46), adding a relay node increases the total delay. When $\rho > \rho_f$, it follows that $g(\rho, y) < 0$. By (5.46), adding a relay node and placing all the nodes equal distance apart decreases the total delay.

Next, we prove that $\rho_f(m)$ is a monotonically increasing function of m, i.e., $\rho_f(m+1) > \rho_f(m)$. Recall that $\lim_{\rho\to 0^+} g(\rho, m) = \tau_0 > 0$, which is not dependent on m, and $\frac{\partial g(\rho,m)}{\partial \rho} < 0$. To show $\rho_f(m+1) > \rho_f(m)$, it is equivalent to show that $g(\rho_f(m), m+1) > 0$. By (5.46) and the definition of $\rho_f(m)$, we have

$$g(\rho_f(m), m) = (m+1)\tau(\frac{D}{m+1}) - m\tau(\frac{D}{m}) = 0. \tag{5.49}$$

Since we have proved that $t(x) = x\tau(\frac{D}{x})$ is strictly convex, we have $t(m+2) - t(m+1) > t(m+1) - t(m)$, i.e.,

$$\begin{aligned} &(m+2)\tau(\frac{D}{m+2}) - (m+1)\tau(\frac{D}{m+1}) \\ >&(m+1)\tau(\frac{D}{m+1}) - m\tau(\frac{D}{m}). \end{aligned} \tag{5.50}$$

Therefore, given $\rho = \rho_f(m)$, we have

$$\begin{aligned} g(\rho_f(m), m+1) &= (m+2)\tau(\frac{D}{m+2}) - (m+1)\tau(\frac{D}{m+1}) \\ &> (m+1)\tau(\frac{D}{m+1}) - m\tau(\frac{D}{m}) \\ &= g(\rho_f(m), m) = 0. \end{aligned} \tag{5.51}$$

Hence, $\rho_f(m)$ is a monotonically increasing function of m. □

The significance of Lemma 8 is that it can be used to determine the range of PU traffic load (a.k.a. the range of ρ) that makes a particular value of m optimal. Given a $m \geq \left\lfloor \frac{D}{r_c} \right\rfloor + 1$, by equation $g(\rho, m) = 0$, we can numerically derive the threshold $\rho_f(m)$. Then, we have the following proposition.

Proposition 5. *Given an actual ρ, we have*

1. $m^* = \lfloor \frac{D}{r_c} \rfloor + 1$, *if* $\rho \in (0, \rho_f(\lfloor \frac{D}{r_c} \rfloor + 1)]$;
2. $m^* = m$, *if* $\rho \in (\rho_f(m-1), \rho_f(m)]$.

Proof. When $\rho > \rho_f(m)$, adding a relay node decreases the total delay and increments m, which in turn increases the threshold value $\rho_f(m)$ by Lemma 8. Keep incrementing m until ρ is less than the new threshold value $\rho_f(m)$. At this stage, adding extra relay nodes increases the total delay. Therefore, the optimal hop count $m^* = m$, for $\rho \in (0, \rho_f(m)], m = \left\lfloor \frac{D}{r_c} \right\rfloor + 1$ or $\rho \in (\rho_f(m-1), \rho_f(m)], m > \left\lfloor \frac{D}{r_c} \right\rfloor + 1$. □

Note that each optimal hop count m^* corresponds to its own range of ρ determined by function $\rho_f(m)$. Since $\rho_f(m)$ can be numerically computed, these ρ intervals can be computed off-line and stored in a table. When PU traffic load changes (a.k.a. ρ changes), the optimal hop count m^* can be derived simply by looking up the table. This saves a lot of online computation overhead. With m^* computed, the optimal number of relay nodes $n^* = m^* - 1$ can be easily determined. Therefore, we conclude the following proposition.

Proposition 6. *In the flow IPS case, the maximum IPS is achieved when $n^* = m^* - 1$ relay nodes are evenly spaced along the straight line between the source and the destination, where m^* is given by* (5.44), (5.45) *or the table lookup method.*

5.6 Applications and Discussions

In this section, we show that our IPS study can be used as a valuable tool for network design in the following three ways.

1. The SU delay estimation expression in (6.21) can be used to predict the expected delay for a given SU path as follows. First, based on the CRN designs, many system design parameters can be easily obtained, such as SU's constant part of the per hop delay τ_0, the number of channels K, SU receiver sensitivity level T_r, and PU receiver interference threshold T_s. Then, through frequency utilization monitoring, the PU traffic parameters, such as PU traffic arrival rate per unit area λ_P, PU traffic service rate μ_P, and PU traffic service time variance σ_2, can be obtained. Based on the above information, we can compute all the parameters needed for formula (6.21) and derive the delay for any hop given its hop distance. Adding the delay of all the hops together, we then obtain the expected delay of any given path. The accuracy of our path delay prediction will be validated by simulations in Section 5.7.1.

2. The IPS bounds can be used to check whether certain delay-sensitive traffic is supportable in a particular network setting and also help network planning. Given a particular PU setting and certain delay-sensitive traffic, our IPS bounds of the optimal network IPS and optimal flow IPS can be computed. If the IPS of delay-sensitive traffic is larger than the optimal IPS, then it is not supportable whatever network planning techniques are used. If it is smaller than the optimal IPS, then the traffic can be supported, and the minimum delay is achieved by following our optimal node placement policy.

3. Our results also can help routing protocols to reduce the delay of SU flows when nodes' positions are all fixed and delay is a critical routing requirement. A routing protocol that chooses relaying nodes that are closest to our theoretical optimal relay positions can significantly reduce the total delay of a path. In Section 5.7.4, we demonstrate this point by simulating a simple routing design that is aided by our method of computing theoretical optimal relay positions.

5.7 Simulation and Numerical Validation

In this section, we validate the correctness of our analysis by simulations and numerical experiments. We also apply our optimal node placement results to geographic routing, and simulate its delay performance.

5.7.1 Validation of the Delay Estimation

To show the accuracy of delay estimation in (6.21), we compare the theoretical delay estimation with the average delay obtained from simulations in both broadcast and unicast cases. In the broadcast case, the source node is fixed, and random paths are chosen. In the unicast case, both source and destination nodes are fixed, and random paths that guarantee connectivity are chosen.

The simulation region is a square with edge length 10000m. The PU transmitters are uniformly distributed within the simulation region. In the simulation, $K = 20$, $r_c = 110$m, $\tau_0 = 0.1$ms, $\frac{T_r}{T_s} = 2$, $\alpha = 3$ and $\mu_P^{-1} = 1$ms. In the broadcast case, we simulate 1-hop, 2-hop, and 3-hop SU paths. For each path, we generate 50 PU transmitter distribution, and measure the delays for 20000 packet deliveries between the source and the destination. Three possible PU service time distributions are simulated: exponential distribution, uniform distribution, and constant. Their simulation results are shown in Figure 5.4(a), 5.4(b) and 5.4(c), respectively. In the unicast case, the source-destination distance is 500m. We simulate 5-hop, 6-hop and 7-hop SU paths. Other settings are the same as that of broadcast case. Simulation results for different PU service time distribution are shown in Figure 5.5(a), 5.5(b) and 5.5(c), respectively. As shown in these figures, theoretical estimation matches the simulation results, showing high accuracy of (6.21).

5.7.2 Validation of the Theoretical Maximum IPS

Network IPS Case

To validate the correctness of our theoretical network IPS results, we next compare our theoretical maximum network IPS with the actual network IPS computed from simulations. The simulation region is a square with edge length 10000m. The PU transmitters are uniformly distributed within the simulation region. We simulate 1-hop, 2-hop, and 3-hop SU paths. For each path length, we generate 50 paths and for each path, we generate 50 PU transmitter distribution. For each of the settings, we measure the delays for 20000 packet deliveries between the source and the destination. In the simulation, $K = 20$, $r_c = 110$m, $\tau_0 = 0.1$ms, $\frac{T_r}{T_s} = 2$, $\alpha = 3$ and $\mu_P^{-1} = 1$ms. Three possible PU service time distributions are simulated: exponential distribution, uniform distribution, and constant. Their simulation results are shown in Figure 5.6(a), 5.6(b) and 5.6(c), respectively.

We perform two sets of simulations for each distribution. In the first set of simulations, we randomly position SU nodes. The maximum network IPSs in simulations are shown in Figure 5.6(a) (I), 5.6(b) (I), and 5.6(c) (I). The mean and the standard deviation of network IPS in simulations are shown in Figure 5.6(a) (II), 5.6(b) (II), and 5.6(c) (II). The maximum network IPSs from the simulation are below the theoretical maximum network IPS, validating the correctness of our theoretical analysis. When the path hop count increases, the simulated network IPS decreases. This is because a longer SU path has a higher probability that SU nodes may not be aligned on the straight line between the source and destination, causing an excessive delay. When the ρ value increases, the simulated network IPS decreases. This is because the PU traffic becomes heavier when ρ increases, causing a larger delay.

In the second set of simulations, SU nodes are evenly spaced along the straight line between the source and the destination. We focus on examining the delay of a 3-hop SU path. The one-hop distance d is set to d^*, $0.8d^*$ and $1.2d^*$. The simulated network IPSs are shown in Figure 5.6(a) (III), 5.6(b) (III), and 5.6(c) (III). When $d = d^*$, the simulated network IPS curves match the theoretical maximum network IPS curve. When $d = 0.8d^*, 1.2d^*$, the simulated IPS curves are below the theoretical maximum network IPS curves. This proves that our maximum network IPS can be achieved when SU nodes are optimally deployed.

Flow IPS Case

Next, we compare our theoretical maximum IPS with the actual IPS computed from simulations in the flow IPS case. The simulation region is a square with edge length 10000m. The PU transmitters are uniformly distributed within the simulation region. The source-destination distance is 500m. We simulate m^*-hop, $m^* + 1$-hop and $m^* + 2$-hop SU paths, where m^* is the optimal number of relay nodes. For each path length, we generate 50 paths and for each path, we generate 50 PU transmitter distribution. For each of the settings, we

measure the delays for 20000 packet deliveries between the source and the destination. In the simulation, $K = 20$, $r_c = 110$m, $\tau_0 = 0.1$ms, $\frac{T_r}{T_s} = 2$, $\alpha = 3$ and $\mu_P^{-1} = 1$ms. Three possible PU service time distributions are simulated: exponential distribution, uniform distribution, and constant. Their simulation results are shown in Figure 5.7(a), 5.7(b) and 5.7(c), respectively.

We perform two sets of simulations for each distribution. In the first set of simulations, we randomly position SU nodes as long as they maintain connectivity between the source and the destination. The maximum flow IPSs in simulations are shown in Figure 5.7(a) (I), 5.7(b) (I), and 5.7(c) (I). The mean and the standard deviation of flow IPS in simulations are shown in Figure 5.7(a) (II), 5.7(b) (II), and 5.7(c) (II). The maximum flow IPSs from the simulation are below the theoretical maximum flow IPS, validating the correctness of our theoretical analysis.

In the second set of simulations, SU nodes are evenly spaced along the straight line between the source and the destination. The simulated mean and their standard deviation are shown in Figure 5.7(a) (III), 5.7(b) (III), and 5.7(c) (III). When $m = m^*$, the simulated flow IPS curves match the theoretical maximum flow IPS curve. When $m = m^* + 1, m^* + 2$, the simulated flow IPS curves are below the theoretical maximum flow IPS curves. This proves the correctness of Lemma 7 and Proposition 6.

5.7.3 Optimal One-hop Distance, Optimal Number of SU Relay Nodes and Theoretical Upper Bounds

Network IPS Case

To demonstrate the correctness of Proposition 4, we numerically compute the optimal one-hop distance d^* and the corresponding maximum network IPS for different network settings, e.g., communication range r_c, and PU service time distribution. The d^* is derived from (5.32) and (5.34). We perform numerical experiments for multiple cases with different r_c values and PU service time distributions. In the experiments, we have $K = 20$, $\frac{T_r}{T_s} = 2$, $\alpha = 3$, $\tau_0 = 0.1$ms and $\mu_P^{-1} = 1$ms. We consider three PU service time distributions: exponential distribution, uniform distribution and constant as shown in Figure 5.8(a), 5.8(b), and 5.8(c), respectively.

For each case, the optimal one-hop distances d^* and the corresponding theoretical maximum network IPSs with respect to ρ values are shown. As shown in these figures, for each r_c value, there exists a threshold ρ value. Below this threshold, the optimal one-hop distance $d^* = r_c$. Above this threshold, the optimal one-hop distance $d^* < r_c$. We also observe that the optimal one-hop distance d^* decreases when the ρ value increases. This is because a higher ρ value indicates heavier PU traffic. Therefore, for a higher ρ value, the optimal one-hop distance d^* is shorter to avoid excessive blocking from PU traffic. These results validate

the correctness of Proposition 4. For all different r_c values, the bound decreases when the ρ value increases. This is because a higher ρ value indicates heavier PU traffic, which slows down the IPS.

Flow IPS Case

To demonstrate the correctness of Proposition 5, we numerically compute the optimal number of relay nodes n^* and the corresponding theoretical maximum flow IPSs for different network settings, e.g., communication range r_c and PU service time. $n^* = m^* - 1$ and is derived from (5.45). We perform numerical experiments for multiple cases with different r_c values and PU service time distributions. In the experiments, we have $K = 20$, $\frac{T_r}{T_s} = 2$, $\alpha = 3$, $\tau_0 = 0.1$ms and $\mu_P^{-1} = 1$ms. We consider three PU service time distributions: exponential distribution, uniform distribution, and constant as shown in Figure 5.9(a), 5.9(b) and 5.9(c), respectively.

For each case, the optimal number of relay nodes n^* and the corresponding theoretical maximum flow IPSs with respect to ρ values are shown. As shown in these figures, for each r_c value, there are threshold ρ values. Above these thresholds, n^* increments. We also observe that n^* is a non-decreasing function of ρ. This is because when ρ is large, the flow IPS is mainly constrained by the interference from PU traffic. A larger number of relay nodes result in a shorter one-hop distance and a shorter sensing range, rendering less interference from PU traffic. Therefore, n^* is a non-decreasing function of ρ. These results validate the correctness of Proposition 5. The general trends and underlying rationales are the same as that of the network IPS case. The maximum flow IPSs are slightly smaller than the maximum network IPSs. This is because the source-destination distance is not necessarily a multiple of d^* in the network IPS case, which decreases the achievable maximum flow IPS.

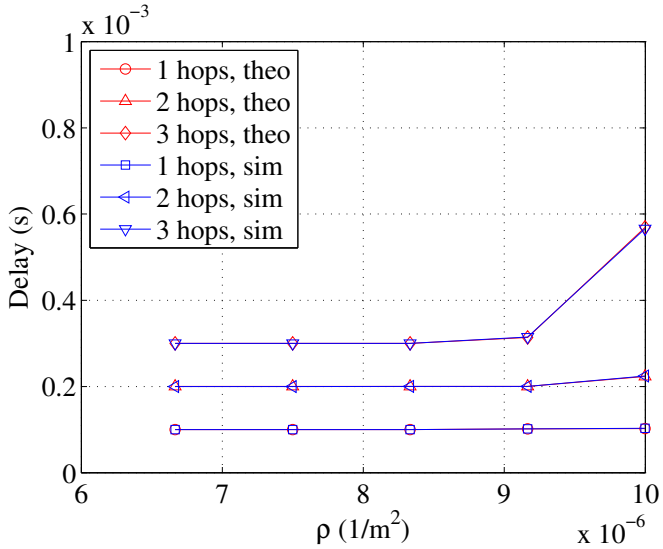
5.7.4 Applications to Delay-sensitive Routing

Next, we apply our analytical results in Proposition 6 to find the best next hop relay node in geographic routing. In the ith hop, the current forwarding node determines the n^* optimal positions on the straight line between the current node and the destination node. Due to limited SU node density, it is often impossible to find a SU node exactly at optimal positions. Our protocol, named modified greedy forwarding, chooses the SU node that is closest to the next optimal position and also within the SU communication range as the next hop node. This process continues until the next optimal position is the position of the destination, i.e., $n^* = 0$. We compare our protocol with greedy forwarding and most forwarding progress within radius (MFR) [47]. In greedy forwarding, the current forwarding node forwards packets to the node that is closest to the destination node. In MFR, the current forwarding node forwards packets to the node, which is within the communication range and also has the largest projection progress along the straight line from the current node to destination node.

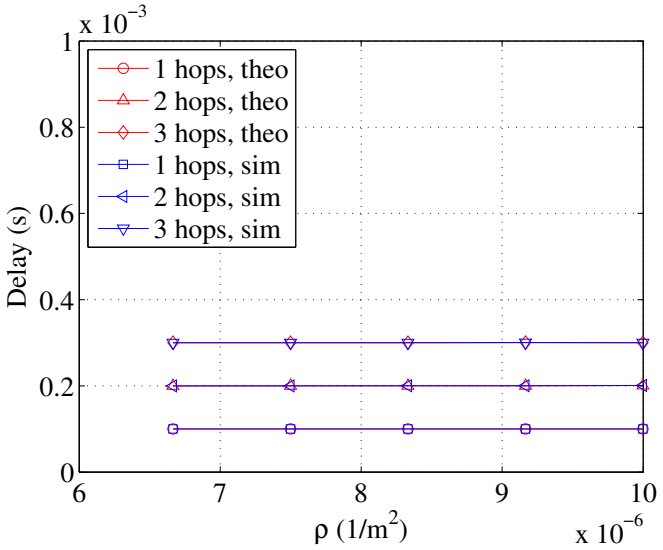
The simulation region is a square with edge length 10000m. In the simulation, $K = 20$, $r_c = 110$m, $\tau_0 = 0.1$ms, $\frac{T_r}{T_s} = 2$, $\alpha = 3$ and $\mu_P^{-1} = 1$ms. The PU transmitters are uniformly distributed within the simulation region. The SU source-destination distance is 500m. 1000 SU nodes are uniformly distributed within a disk region centered at the source node and with a radius of 610m. We simulate our modified greedy forwarding, greedy forwarding and MFR for 50 PU transmitter distribution. For each setting, we measure the delays for 20000 packet deliveries between the source and the destination. Three possible PU service time distributions are simulated: exponential distribution, uniform distribution, and constant. Their simulation results are shown in Figure 5.10(a), 5.10(b) and 5.10(c), respectively. As shown in these figures, modified greedy forwarding significant outperforms greedy forwarding and MFR especially in the high PU traffic load region.

5.8 Conclusions

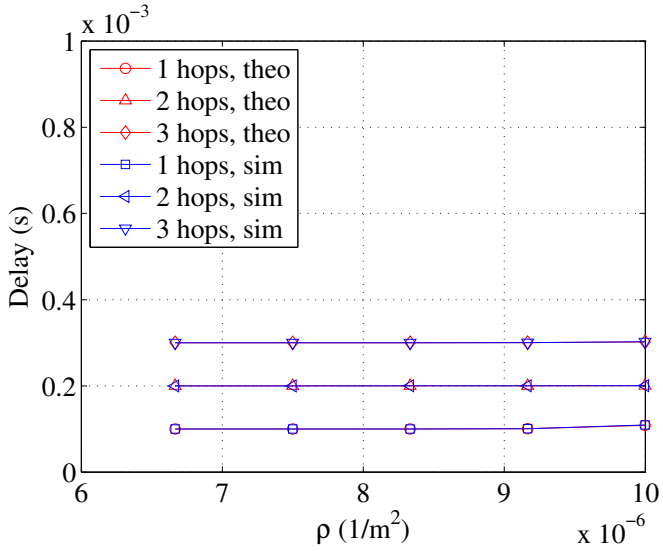
In this chapter, we derive the maximum IPSs in CRNs. We study two cases: the network IPS and the flow IPS. In the network IPS case, we discover that the maximum network IPS is related to a threshold value of the PU activity level. Below the threshold, the maximum IPS is achieved when the one-hop distance equals the communication range of cognitive radios. Above the threshold, the maximum IPS is achieved when using an optimal one-hop distance which is less than the communication range. We design efficient numerical methods to compute the optimal one-hop distance and the corresponding maximum IPS. In the flow IPS case, we discover that the maximum IPS is achieved when an optimal number of SU relay nodes are evenly spaced on the straight line between the source node and destination node. The optimal number of relay nodes shows a stair-like incremental trend, when the PU activity level increases. We design multiple numerical methods to compute the optimal number of SU relay nodes. Our simulation and numerical results prove the correctness of our analyses.



(a) Exponential distribution service time

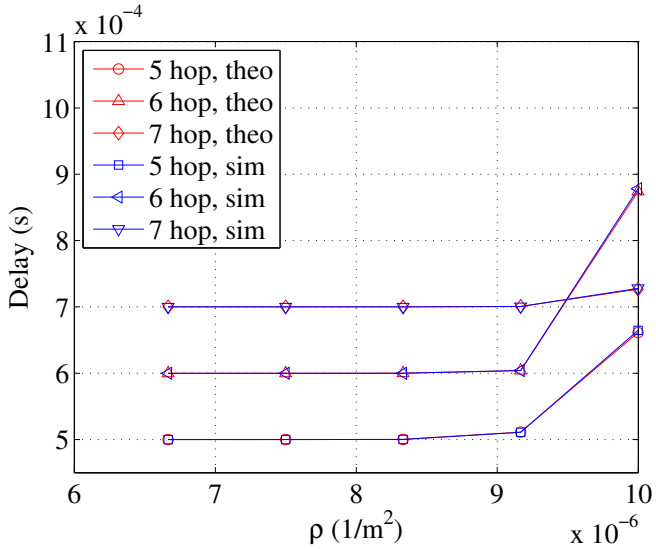


(b) Uniform distribution service time

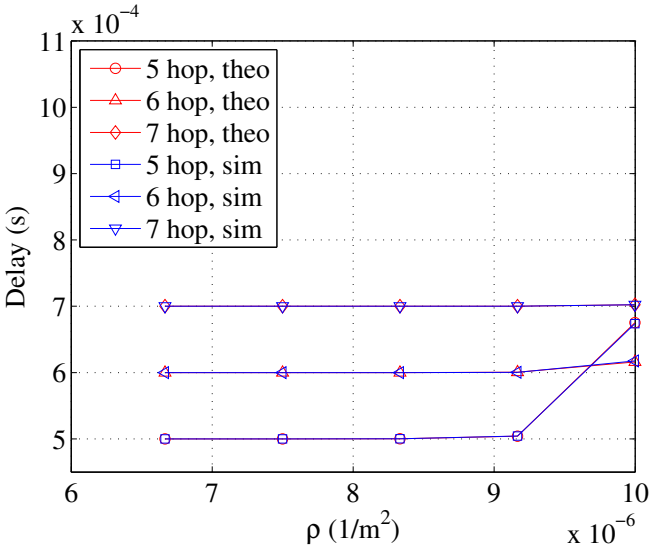


(c) Constant service time

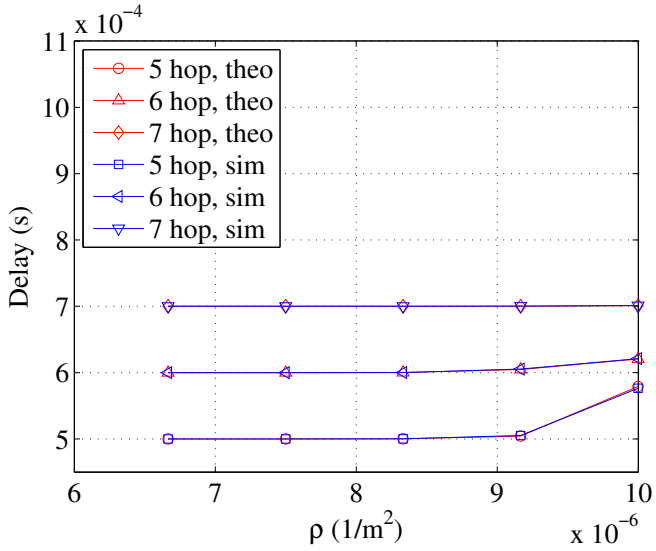
Figure 5.4: Theoretical delay estimation and the simulated delay for given broadcast paths



(a) Exponential distribution service time

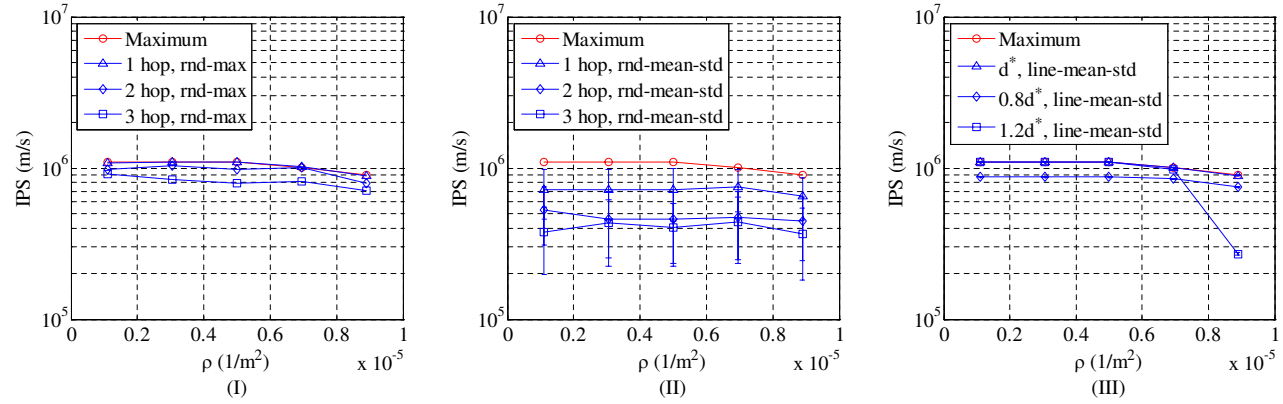


(b) Uniform distribution service time

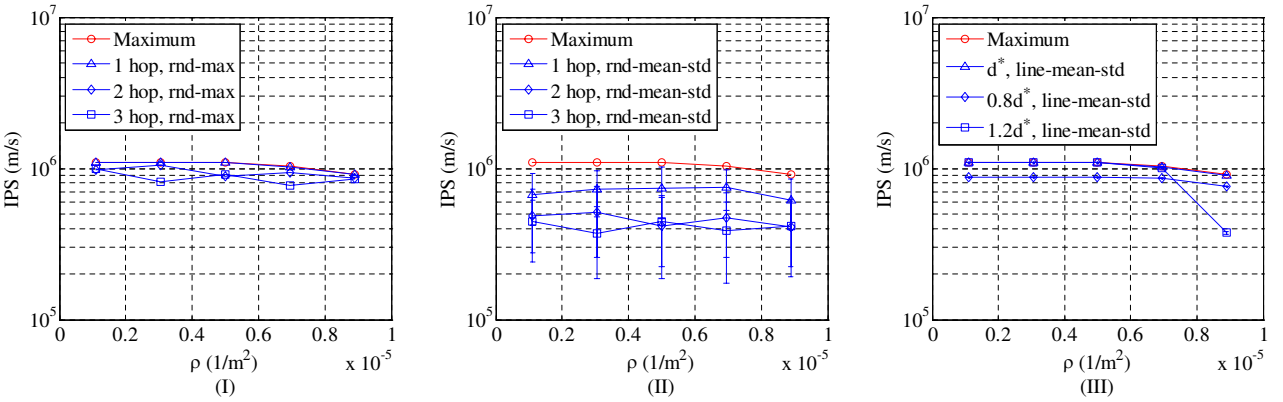


(c) Constant service time

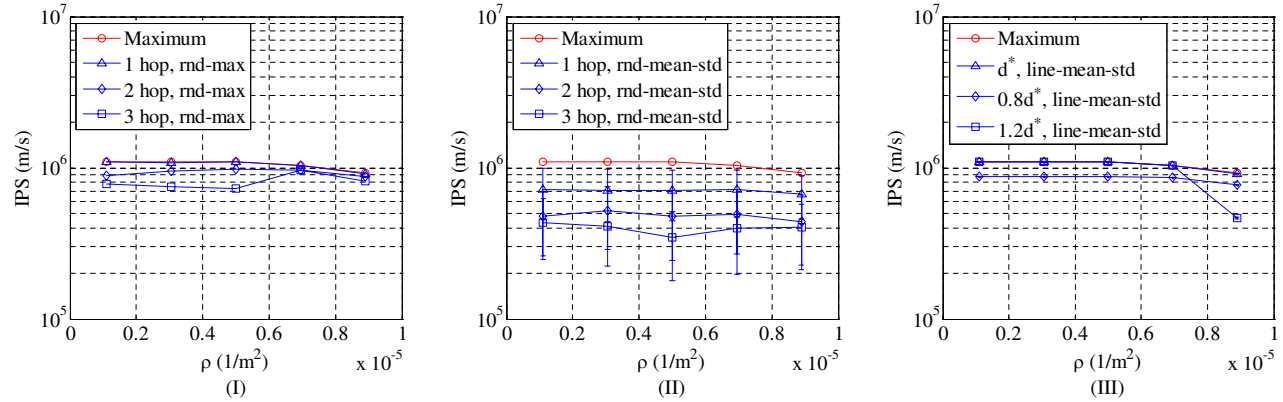
Figure 5.5: Theoretical delay estimation and the simulated delay for given broadcast paths



(a) Exponential distribution service time

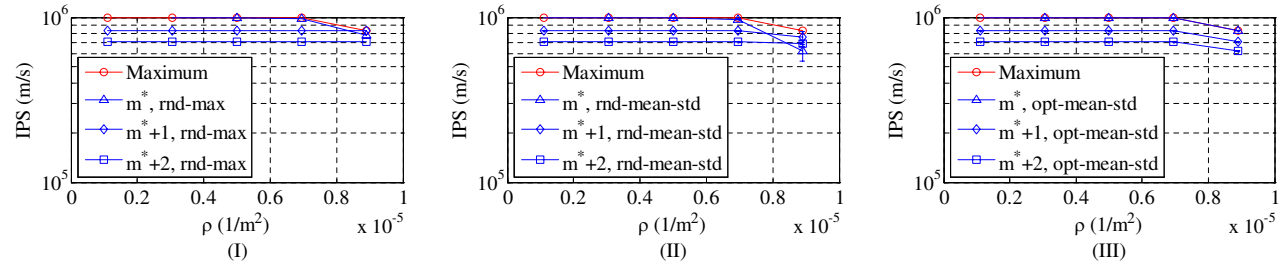


(b) Uniform distribution service time

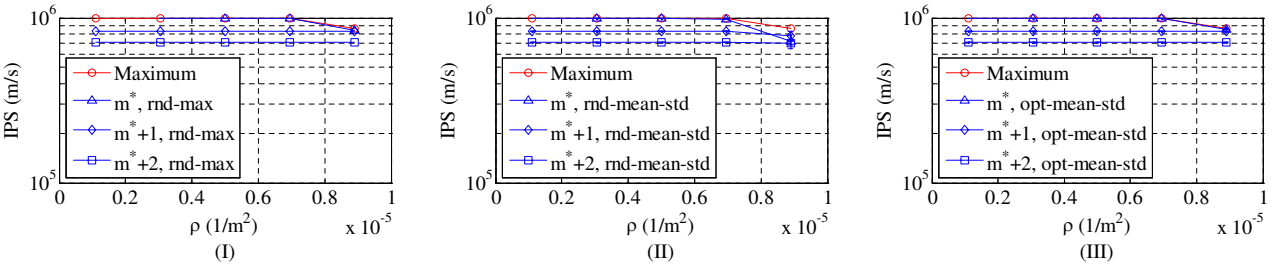


(c) Constant service time

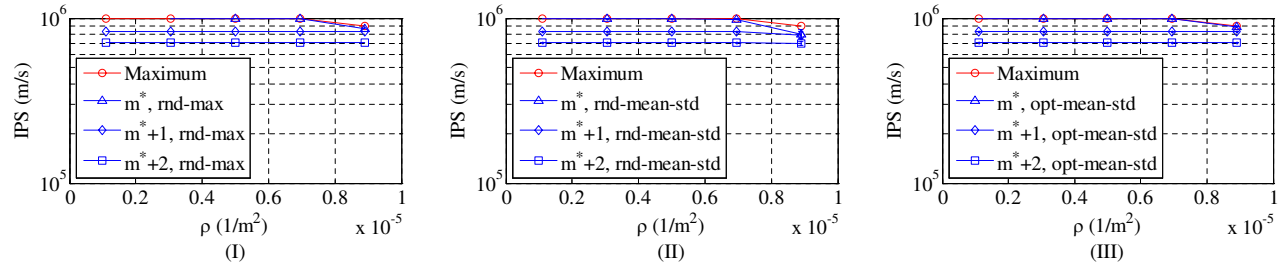
Figure 5.6: Theoretical maximum IPS and the simulated IPS for the network IPS case



(a) Exponential distribution service time

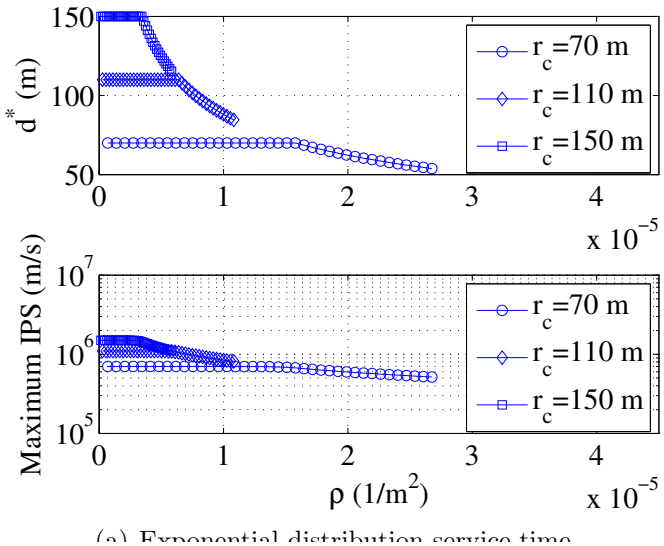


(b) Uniform distribution service time

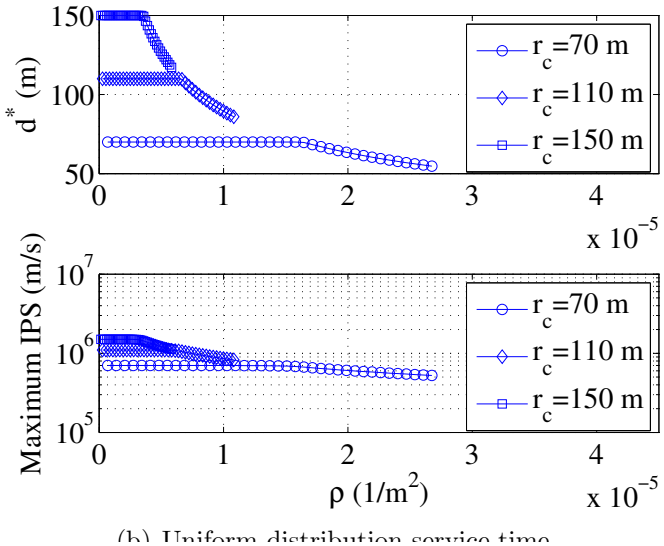


(c) Constant service time

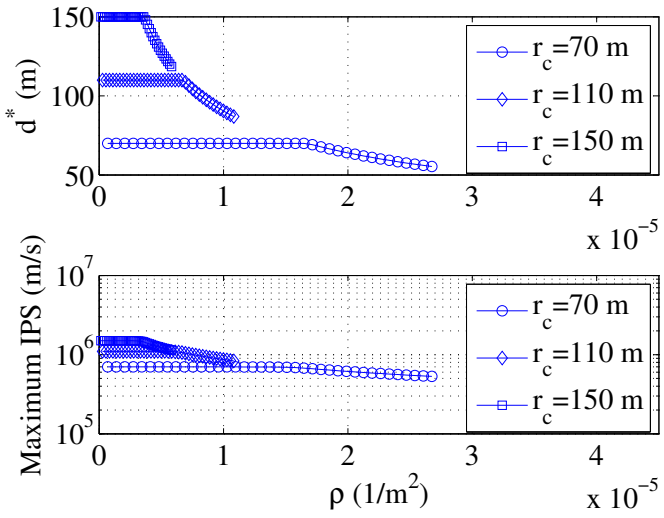
Figure 5.7: Theoretical maximum IPS and the simulated IPS for the flow IPS case



(a) Exponential distribution service time

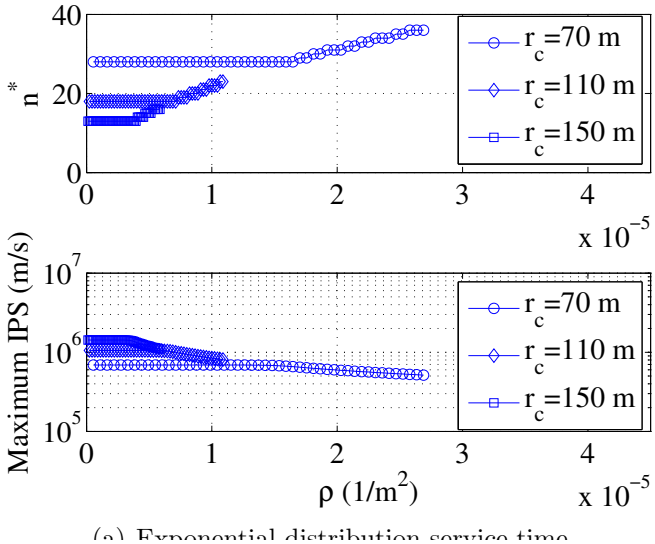


(b) Uniform distribution service time

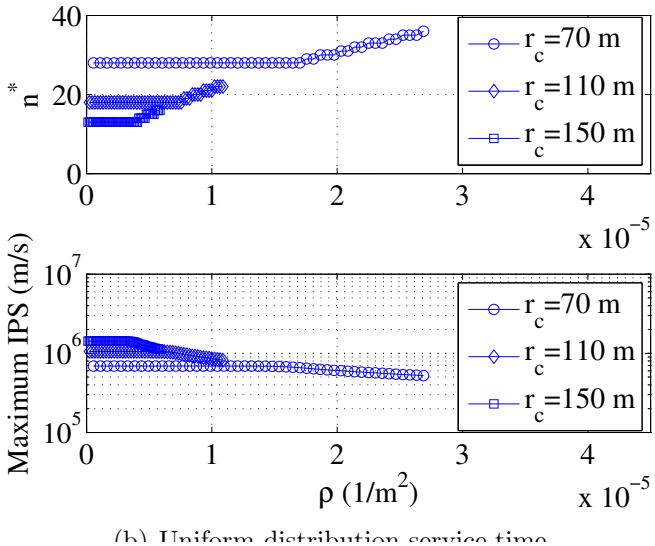


(c) Constant service time

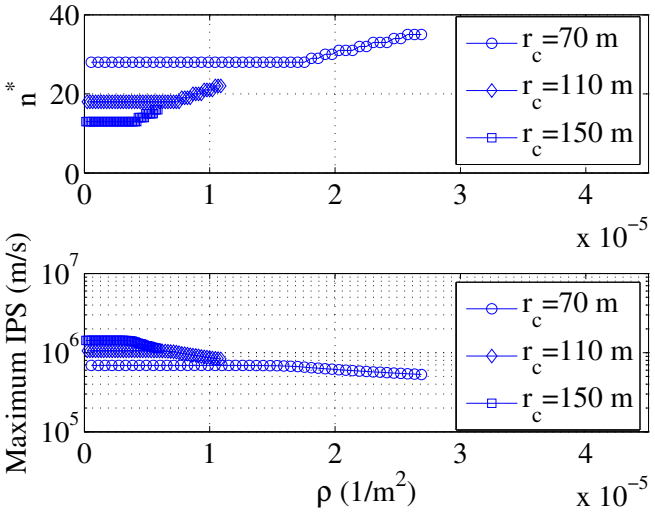
Figure 5.8: Optimal one-hop distance d^* and theoretical maximum IPS for the network IPS case



(a) Exponential distribution service time

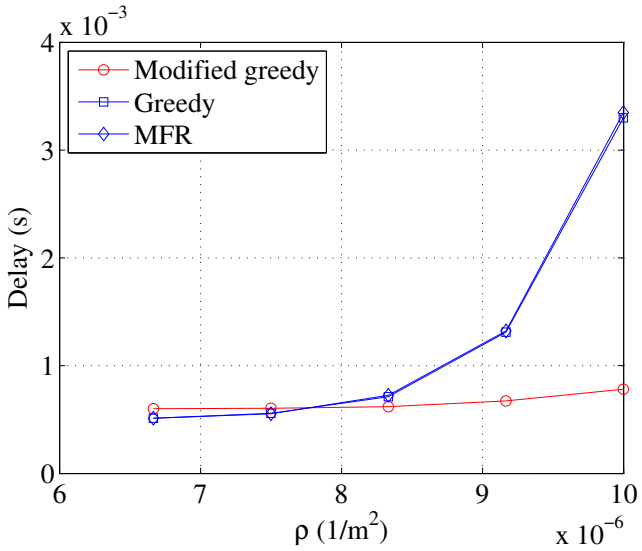


(b) Uniform distribution service time

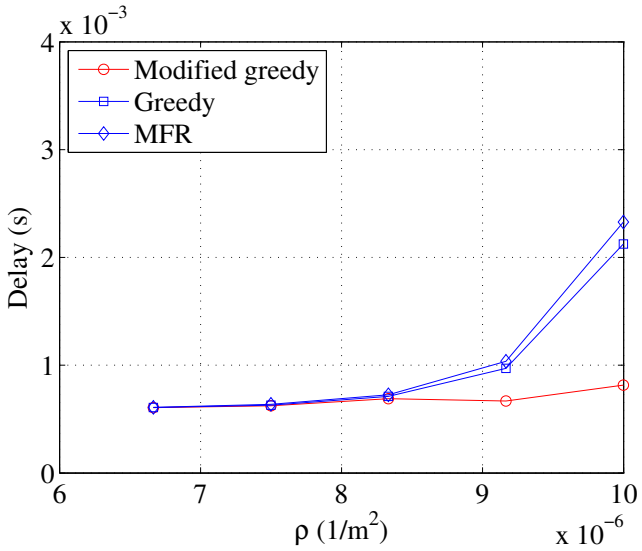


(c) Constant service time

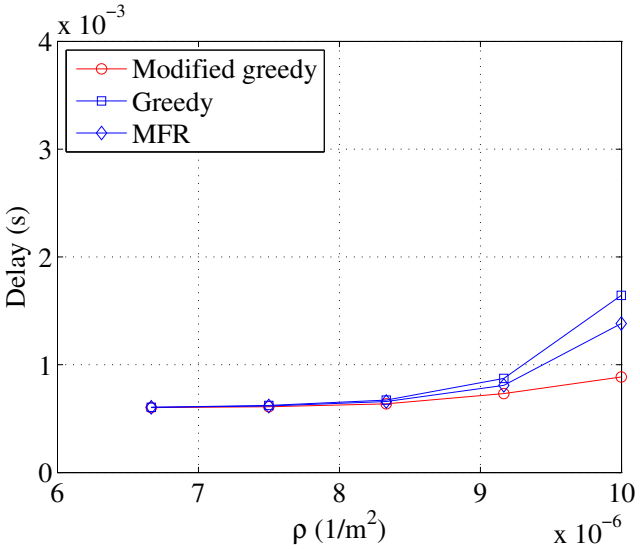
Figure 5.9: Optimal number of relay SU nodes n^* and theoretical maximum IPS for the flow IPS case



(a) Exponential distribution service time



(b) Uniform distribution service time



(c) Constant service time

Figure 5.10: Delay performance of different routing schemes

Chapter 6

Convergecast Delay Analysis of Large Scale Sensor Networks Coexisting with WiFi Networks

6.1 Introduction

The de facto radio technology for modern sensor networks is IEEE 802.15.4 (ZigBee) [24] due to its low power feature. However, a ZigBee radio operates in ISM bands, which are being shared by many other radio technologies. Thus, ZigBee and these radio technologies can cause mutual interference to each other and create coexistence issues between sensor networks and other types of networks in ISM bands.

Among these coexistence issues, the one between WiFi networks and sensor networks is the most pronounced. This is because WiFi networks use IEEE 802.11 radio technology [25] and are the most popular network type that operates in ISM bands. WiFi's wide deployment makes it very likely that sensor networks are co-located with WiFi networks. In addition, the differences between their underlying ZigBee and IEEE 802.11 technologies further aggravate the coexistence issue between WiFi and sensor networks. While WiFi's underlying IEEE 802.11 technology is designed for high data rate, relatively long range and reliable data communications, sensor networks' underlying ZigBee technology is aimed for low power, low data rate, and relatively short range communications. To achieve high data rate communications, WiFi transmission power is typically 100 times stronger than sensor node transmission power. Thus, sensor packets are less likely to be detected by WiFi transmitters [23]. As a result, WiFi's CSMA mechanism cannot prevent WiFi packets from colliding with ZigBee traffic. Hence, as demonstrated by experimental studies [49, 21, 36, 67], high power WiFi packets often corrupt a large amount of ongoing sensor packets. Even if sensor signals can somehow be detected by WiFi transmitters, the sensor transceiver's response time is 16

times longer than WiFi transceiver's response time [67]. Therefore, sensor transmission is often preempted by WiFi transmission due to its slow responses.

The above problem of WiFi's interference and preemption of sensor network packets can potentially have very serious impact on the packet delivery delay for large-scale sensor networks, where every sensor in a network needs to send its sensing data through potentially multiple hops to reach a sink. Such convergecast delay performance is an important design requirement for many large-scale sensor network applications, because an outdated packet not only renders its carried data useless but also wastes precious network resources such as energy and bandwidth. Therefore, analysis of convergecast delay in sensor networks coexisting with WiFi networks is a very important design problem. Unfortunately, while there are some existing works [55, 12, 53, 13] on delay estimation and bound analyses of convergecast in sensor networks, none of them takes WiFi interference into consideration. Hence, in this chapter, we fill this technical void.

In this work, we model the delay of sensor networks coexisting with WiFi networks as a two priority preemptive repeat identical queueing system. Using queueing theory and probabilistic theory, we derive the packet convergecast delay of sensor networks coexisting with WiFi networks. Further, we develop a simplified linear approximation model to estimate the expected delay. The correctness of both models is validated by simulations.

The remaining part of this chapter is organized as follows. Related work is summarized in Section 6.2. In Section 6.3, the network model is presented and the delay analysis problem is formulated as a two priority M/G/1 preemptive repeat identical queueing system. The expected delay of sensor packets in a sensor network coexisting with WiFi nodes is analyzed and derived in Section 6.4. An example routing protocol is used in our analysis in Section 6.5. A simplified linear approximation model is proposed in Section 6.6. Our analytical results are validated by NS2 simulations in Section 6.7. Finally, Section 6.8 concludes the chapter.

6.2 Related Work

There are two bodies of works related to this work. The first one is coexistence between WiFi and ZigBee radios. These papers proposed new protocols and schemes to improve the performance of ZigBee radios when ZigBee and WiFi coexist. None of them provides theoretical analysis on network delay performance when ZigBee-based large-scale sensor networks and WiFi networks coexist. For example, in [23], the authors showed that WiFi transmitters cannot detect ZigBee signals through real system experiments. They then proposed a new protocol, called WISE, that enables ZigBee traffic to use the white space between WiFi transmissions, and reschedule sensor traffic. In [68], the authors proposed a new mechanism, called Cooperative Busy Tone (CBT), to enhance the visibility of ZigBee signals to WiFi transmitters. The basic idea of CBT is to designate another ZigBee node as a signaler

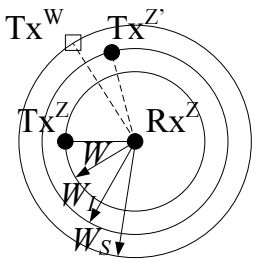


Figure 6.1: One hop distance W, sensing range W_S and interference range W_I

emitting the busy tone. In [36], the authors designed *BuzzBuzz* to counteract the WiFi interference. The basic idea is to use multiple ZigBee headers to gain multiple opportunities to detect WiFi packets.

Another body of work is on the convergecast delay analysis in single-tier networks, where no outside interference from other networks are considered. In [55], the authors analyzed the convergecast delay of WiFi mesh networks with RTS/CTS mechanism. In [12] and [13], the authors analyzed data collection delay in slotted and synchronized sensor networks equipped with both directional and omnidirectional antennas. In the case of omnidirectional antennas, only line topologies are analyzed. In [53], the authors analyzed both grid and random sensor networks, and derived the delay performance when random forwarding routing and slot reservation MAC are used. Our work differs from all these existing convergecast work in our practical consideration of WiFi's interference on sensor networks.

6.3 Network Model and Problem Formulation

6.3.1 Network Model

We consider a large-scale sensor network coexisting with many WiFi nodes. The WiFi nodes' operating channel and the sensor networks' operation channel overlap in the ISM bands. There is a sink node in the sensor network, collecting data from sensor nodes. Sensor nodes generate fixed length data packets following a Poisson arrival process. Due to limited communication range, each sensor node may also relay data packets generated by other sensor nodes. To preserve secrecy of other nodes' data and also guarantee timely delivery, we do not assume data aggregation at relay sensor nodes. WiFi nodes are formed into pairs. Each pair consists of one transmitter and one receiver. We also assume WiFi packet arrival follows a Poisson arrival process. We assume sensor transceivers cannot transmit and receive at the same time, and each sensor data packet will be acknowledged by an ACK packet.

To model the impact of WiFi traffic on sensor network's delay, note that a WiFi node's transmission power (typical value is 100 mW [25]) is much higher than a sensor node's transmission power (typical value is 1 mW [24]). Thus, to guarantee reliable packet trans-

mission, a sensor transmitter needs to sense whether there are active WiFi transmitters in its neighborhood before its transmission. Thus, a sensor's sensing range for WiFi transmitters is related to how WiFi traffic affects sensor transmissions. As shown in Figure 6.1, suppose sensor transmitter T_x^Z is sending data to sensor receiver R_x^Z, the distance between the sensor transmitter and receiver is W, and the nearby WiFi transmitter T_x^W is interfering with this sensor transmission. We denote a sensor node R_x^Z's sensing range for WiFi node T_x^W as W_s, and assume that outside of W_s any WiFi transmitter T_x^W cannot corrupt the sensor node transmission from from T_x^Z to R_x^Z. In addition to the interference from WiFi transmitters, interference from nearby sensor transmitters may also corrupt an ongoing sensor transmission. Therefore, before data transmission, the sensor transmitter also needs to detect whether there are active sensor transmitters in its neighborhood. As shown in Figure 6.1, we denote this interference range within which another sensor transmitter $T_x^{Z\prime}$ can corrupt an ongoing sensor transmission between T_x^Z and R_x^Z as W_I. Since a packet transmission between a pair of sensor nodes requires bidirectional communications (DATA-ACK handshake), both sensor transmitter and receiver check active WiFi transmitters and sensor transmitters in its neighborhood.

To guarantee successful sensor transmission, the interference from any WiFi transmitter or sensor transmitter to a sensor receiver should be below a certain threshold. We assume that when the desired received signal power is T_{CP} times higher than the received interference power, the desired signal can be successfully received.

Thus, considering the interference from other sensor transmitters to a receiving sensor receiver, the interference range among sensor nodes, W_I, must satisfy the following equation:

$$\frac{\frac{P_t^Z}{W^\alpha}}{\frac{P_t^Z}{W_I^\alpha}} = \frac{W_I^\alpha}{W^\alpha} = T_{CP}, \tag{6.1}$$

where P_t^Z is the sensor transmission power, $\alpha \geq 2$ is the path attenuation exponent, and W is the distance between the communicating sensor transmitter and receiver. Equation (6.1) can be converted to

$$W_I = C_1 W, \tag{6.2}$$

where $C_1 = (T_{CP})^{\frac{1}{\alpha}}$.

To avoid the interference from a WiFi transmitter to a sensor receiver, the sensor's minimum sensing range W_S must satisfy

$$\frac{\frac{P_t^Z}{W^\alpha}}{\frac{P_t^W}{W_S^\alpha}} = \frac{P_t^Z}{P_t^W}\frac{W_S^\alpha}{W^\alpha} = T_{CP}, \tag{6.3}$$

where P_t^W is the WiFi transmission power. Thus, we have

$$W_S = C_2 W, \tag{6.4}$$

where $C_2 = (T_{CP}\frac{P_t^W}{P_t^Z})^{\frac{1}{\alpha}}$.

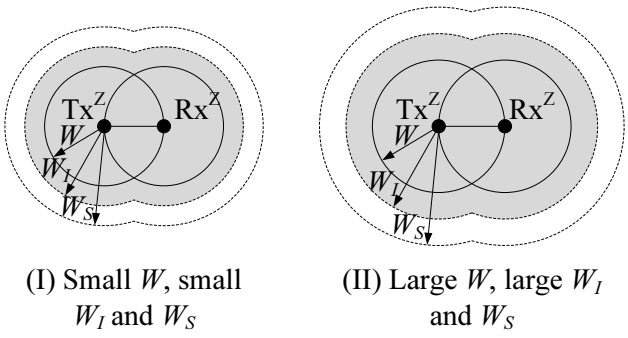


Figure 6.2: Impact of different one hop distance W

6.3.2 Problem Formulation

The goal of this work is to analyze the convergecast delay in a sensor network under the influence of coexisting WiFi networks. We quantify the convergecast delay of a sensor network by its packet convergecast delay, which is the expected time that a sensor that is l distance away from the sink takes to deliver a packet to the sink. We denote this expected packet convergecast delay as $T(l)$.

In a multihop sensor network, sensor packets are forwarded to the sink via multiple hops. Thus, the convergecast delay of a packet equals the summation of delays along each hop on its path to the sink. Therefore, to derive convergecast delay, we need to figure out two things: the number of hops that a packet may take to reach the sink and the delay that it may experience at each hop. To do this, we need to consider the following factors.

First, according to our network model, sensor packet transmissions may be preempted by WiFi packet transmissions from neighboring WiFi nodes, and may also compete the channel with packet transmission from other neighboring sensor nodes. As shown in Figure 6.2, a long per hop distance W will result in longer sensing range W_S and interference range W_I according to (6.2) and (6.4). This will induce more WiFi traffic and competing sensor traffic within the sensing region and interference region. Therefore, it is expected that per hop delay is a function of per hop distance, which is determined by path topology. In addition, the path delay is not necessarily a monotonic function of per hop distance, since a path with a longer per hop distance may take few hops to reach the sink but on the other hand take longer time to finish each hop.

Second, a sensor node often needs to relay packets from other sensor nodes that are further away from the sink. The relaying traffic not only competes with a sensor's locally generated traffic for transmission time, but also increases this sensor's chance for competing transmission time with its neighboring sensors. Hence, relaying traffic introduces additional delay. To make things more complicated, since all the sensor traffic are relayed to the central sink, it is expected that the relaying traffic intensity is not a constant. Instead, it increases as one gets closer to the sink node. Therefore, per hop delay is also a complex function of a hop's

distance to the sink node.

Finally, in a multihop path that a packet travels to reach the sink, the distance of each hop will affect the final total hop count as well as the later hops' relaying traffic intensity. Thus, all the factors that we have considered for per hop delay intertwine over multihop paths and depend on the exact topology of the networks and path that a packet takes.

To derive the above complex relationship mathematically, we start our formulation of the convergecast delay as the following equation:

$$T(l) = \sum_{m=\lceil \frac{l}{r_c} \rceil}^{\infty} \tau(m)P(m), \tag{6.5}$$

where r_c is the sensor maximum communication range constrained by sensor hardware design, $\tau(m)$ is the expected delay of a path with path length m, and $P(m)$ is the probability that the path from the source to sink has length m.

Essentially, equation (6.5) considers that for different sources at l distance away from the sink, they may take different numbers of hops and different paths to reach the sink. Thus, equation (6.5) groups them by their hop count to decompose the expected convergecast delay problem into two subproblems. The first subproblem is to compute $P(m)$, the probability that the path from a source to sink is m hop, given that the source is l distance away from the sink. The second subproblem is finding out $\tau(m)$, which will be determined by the stochastic distributions of the m-hop path's per hop distance and its resulting per hop delay. In the following, we will show how we solve these two subproblems.

6.4 Delay Analysis

In this section, we solve equation (6.5) by computing both $\tau(m)$ and $P(m)$.

6.4.1 Compute $P(m)$

Given that a path to the sink travels distance l, the probability that the path is m hop, i.e., $P(m)$, equals the probability that both of the following two conditions hold. First, the aggregate progress to the sink from the first $m-2$ hops plus the maximum sensor communication range r_c is less than the distance l. Second, the aggregate progress from the first $m-1$ hops plus the maximum sensor communication range r_c is greater than or equal to the distance l. Define Y_i as the i-th hop's progress to the sink, and assume all Y_i are independent and identically distributed random variables. Mathematically, we have

$\sum_{i=1}^{m-2} Y_i + r_c < l$ and $\sum_{i=1}^{m-1} Y_i + r_c \geq l$. Denote $\sum_{i=1}^{m-2} Y_i = Z$, υ as the expected value of Y_i, and σ^2 as the variance of Y_i. By the central limit theorem, Z can be approximated as Gaussian distribution with mean $(m-2)\upsilon$ and variance $(m-2)\sigma^2$ [53], i.e.,

$$f_Z(z) = \frac{1}{\sigma\sqrt{2\pi(m-2)}} e^{-\frac{(z-(m-2)\upsilon)^2}{2(m-2)\sigma^2}}. \tag{6.6}$$

Therefore, we have

$$\begin{aligned} P(m) &= \Pr\{\sum_{i=1}^{m-2} Y_i + r_c < l, \sum_{i=1}^{m-1} Y_i + r_c \geq l\} \\ &= \Pr\{Z + r_c < l, Z + Y_{m-1} + r_c \geq l\} \\ &= \Pr\{Z < l - r_c, Y_{m-1} \geq l - r_c - Z\} \\ &= \frac{\int_{-\infty}^{l-r_c} e^{-\frac{[z-(m-2)\upsilon]^2}{2(m-2)\sigma^2}} [1 - F_Y(l - r_c - z)]dz}{\sigma\sqrt{2\pi(m-2)}}, \end{aligned} \tag{6.7}$$

where $F_Y(y)$ is the CDF of Y. $F_Y(y)$ depends on which routing protocol is used, and will be derived later in Section 6.5.

6.4.2 Compute $\tau(m)$

Before computing $\tau(m)$, the expected delay of a path of hop count m, we first formulate the one-hop delay analysis problem as a queueing model, based on which the expected one hop delay can be derived to be a function of the traffic arrival rates of both sensors and WiFi nodes. We then show how to compute these two arrival rates. Finally, by summing delay of all hops, we get the expected delay $\tau(m)$.

A Queueing Model for the Channel

Recall that since sensor nodes and WiFi nodes share the same channel, sensor nodes have to yield to WiFi transmissions, and ongoing sensor transmissions can be interrupted by WiFi interference. To derive the expected one-hop delay of sensor packets, we can model the shared channel between sensor traffic and WiFi traffic as a M/G/1 priority preemptive repeat identical queueing system. In this model, WiFi traffic is modeled as a high priority traffic to access the channel, and sensor traffic is modeled as a low priority traffic to access the channel. When WiFi traffic comes, sensor traffic is preempted. Only when current WiFi traffic transmission is finished, the same sensor traffic is repeated. The repeated sensor traffic transmission is successful, only if it is not preempted by other WiFi traffic. Based on this model, we derive the expected delay using queueing theory analysis.

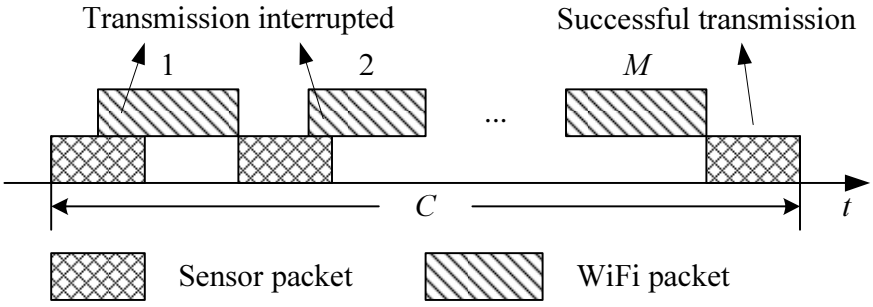


Figure 6.3: Completion time

In our analysis, we use subscript 1 to denote low priority (sensor) traffic, and subscript 2 to denote high priority (WiFi) traffic. We denote traffic load of priority i traffic as $\rho_i = \frac{\lambda_i}{\mu_i} = \lambda_i \overline{X}_i$, where λ_i is the arrival rate of priority i traffic, μ_i is the service rate of priority i traffic, X_i is the service time of priority i traffic, and $\overline{X_i}$ is the mean value of X_i. By [8], the expected time that a sensor packet spent in the system, i.e., waiting time plus service time, is

$$g_1 = \frac{\lambda_1 E[C^2]}{2(1-\lambda_1 E[C])} + \frac{\lambda_2 E[B_2^2]}{2(1-\lambda_2 E[B_2])} + E[C], \tag{6.8}$$

where C is the sensor packet's completion time. The completion time is defined as the duration of time period between the instant when the transmission of a sensor packet begins and the instant when service of the next sensor packet may begin (it does not begin when this packet is still present). Figure 6.3 illustrates an example of C. C includes the service time of the sensor packet, the sum of service time of M WiFi packets preempting sensor packets, and the sum of unsuccessful sensor service time of the preempted sensor packet transmissions. It is shown in [11] that

$$E[C] = (E[e^{\lambda_2 X_1}] - 1)(E[B_2] + \frac{1}{\lambda_2}) \tag{6.9}$$

and

$$\begin{aligned} E[C^2] = {} & 2(E[B_2] + \frac{1}{\lambda_2})^2 E[e^{\lambda_2 X_1} - 1]^2 + (E[B_2^2] + \frac{2E[B_2]}{\lambda_2} + \frac{2}{\lambda_2^2})(E[e^{\lambda_2 X_1}] - 1) \\ & - 2(E[B_2] + \frac{1}{\lambda_2}) E[X_1 e^{\lambda_2 X_1}] \end{aligned} \tag{6.10}$$

The symbol B_2 is the busy period duration of WiFi traffic, and we have

$$E[B_2] = \frac{1}{\mu_2(1-\rho_2)}, \tag{6.11}$$

and

$$E[B_2^2] = \frac{E[X_2^2]}{(1-\rho_2)^3} \tag{6.12}$$

Combining equations (6.8), (6.9), (6.10), (6.11) and (6.12), we have

$$g_1 = \frac{1}{2(1-\frac{\lambda_1(e^{\frac{\lambda_2}{\mu_1}}-1)}{\lambda_2(1-\rho_2)})}\lambda_1[\frac{2(e^{\frac{\lambda_2}{\mu_1}}-1)^2}{\lambda_2^2(1-\rho_2)^2}+(\frac{E[X_2^2]}{(1-\rho_2)^3}+\frac{2}{\lambda_2\mu_2(1-\rho_2)}+\frac{2}{\lambda_2^2})(e^{\frac{\lambda_2}{\mu_1}}-1)$$
$$-\frac{2e^{\frac{\lambda_2}{\mu_1}}}{\lambda_2\mu_1(1-\rho_2)}]+\frac{\lambda_2 E[X_2^2]}{2(1-2\rho_2)(1-\rho_2)^2}+\frac{e^{\frac{\lambda_2}{\mu_1}}-1}{\lambda_2(1-\rho_2)} \tag{6.13}$$

From (6.13), we can see that g, the per-hop delay at a sensor node, is a function of λ_1 and λ_2, which are the arrival rates of the sensor traffic and the WiFi traffic in the neighborhood of the sensor. In the next step, we show how we derive these two arrival rates.

Compute the Traffic Arrival Rates

In (6.13), the sensor traffic arrival rate λ_1 is a function of the current sensor node's distance to the sink node, current sensor link hop distance W, and sensor traffic arrival rate per unit area. Assume WiFi traffic is uniformly generated. Then, the WiFi traffic arrival rate λ_2 is a function of current sensor link hop distance W, and the WiFi traffic arrival rate per unit area.

To derive λ_1 and λ_2, we need to first compute the neighborhood region of a sensor, where the WiFi traffic and sensor traffic in this region compete the shared channel resource with the sensor. Denote U_W as the region in which WiFi transmitters may interfere with a sensor link, as shown by the region inside of two larger circles in Figure 6.2. Denote U_Z as the region in which sensor transmitters may interfere with a sensor link, as shown by the grey region in Figure 6.2. Only when there are no active WiFi transmitter in U_W and no active sensor transmitter in U_Z, the current sensor link can successfully finish data transmission. It can be shown that the areas of U_W and U_Z are $A_W(W) = C_W W^2$ and $A_Z(W) = C_Z W^2$, respectively, where

$$C_W = 2\pi C_2^2 - 2C_2^2 \arccos\frac{1}{2C_2} + \sqrt{C_2^2 - \frac{1}{4}} \tag{6.14}$$

and

$$C_Z = 2\pi C_1^2 - 2C_1^2 \arccos\frac{1}{2C_1} + \sqrt{C_1^2 - \frac{1}{4}}. \tag{6.15}$$

Define λ_W as the WiFi traffic arrival rate per unit area. Then, the WiFi traffic arrival rate within region U_W is

$$\lambda_2 = C_W \lambda_W W^2, \tag{6.16}$$

and the WiFi traffic load competing the channel with the sensor link is $\rho_2 = \frac{\lambda_2}{\mu_2} = \frac{\lambda_W A_W(W)}{\mu_2} = \frac{C_W \lambda_W W^2}{\mu_2}$, where $\mu_2 = \frac{1}{X_2}$ is the WiFi traffic service rate.

The total amount of traffic going through a sensor node includes the traffic generated by this node and the relaying traffic forwarded by this node. Define the sensor traffic generation rate per unit area as λ_Z, and sensor node density as γ_1. Thus, it follows that the expected data rate generated by a sensor node is $\frac{\lambda_Z}{\gamma_1}$. Since all sensor traffic is forwarded to the sink, the aggregate traffic rate each sensor node forwards is not uniform and is related the sensor's distance to the sink. We define $\Lambda(l')$ as the expected relaying traffic arrival rate of a sensor node whose distance to the sink is l'. Since we have assumed that sensor nodes cannot transmit and receive at the same time, relaying a unit of traffic at a sensor requires two units of time, one for transmission and one for reception. The total number of sensor nodes sharing the wireless medium with the current sensor link is $A_Z(W)\gamma_1$, where γ_1 is the sensor node density. Hence, the amount of competing sensor traffic sharing the same channel can be estimated as:

$$\begin{aligned} \lambda_1 &= A_Z(W)\gamma_1\left(\frac{\lambda_Z}{\gamma_1} + 2\Lambda(l')\right) \\ &= C_Z W^2 \lambda_Z + 2C_Z W^2 \gamma_1 \Lambda(l'), \end{aligned} \tag{6.17}$$

where $\mu_1 = \frac{1}{X_1}$ is the sensor traffic service rate.

To compute λ_1, we next derive the relaying traffic arrival rate $\Lambda(l')$. We divide the disk region centered at the sink node into multiple rings with ring thickness $\frac{r_c}{2}$, where r_c is the maximum transmission range of a sensor. In this way, given the connectivity of the network, any node in a ring can communicate with some other nodes in its neighbor rings. The number of nodes in a ring with inner radius l' can be approximated as

$$\begin{aligned} N(l') \approx& 2\pi\left(l' + \frac{r_c}{4}\right)\frac{r_c}{2}\gamma_1 \\ =& \pi\left(l' + \frac{r_c}{4}\right) r_c \gamma_1. \end{aligned} \tag{6.18}$$

Note that all sensors that are located further away from the sink than the ring have to relay their traffic through nodes on the ring. Therefore, the average amount of traffic that each sensor on the ring has to relay is

$$\begin{aligned} \Lambda(l') &= \frac{\text{Total traffic going through the ring}}{N(l')} \\ &\approx \frac{\lambda_S \pi (R^2 - l'^2)}{\pi(l' + \frac{r_c}{4}) r_c \gamma_1} \\ &= \frac{\lambda_S (R^2 - l'^2)}{\gamma_1 (l' + \frac{r_c}{4}) r_c}. \end{aligned} \tag{6.19}$$

Substituting (6.19) back into (6.17), we can derive λ_1's expressions in (6.17). Assuming that μ_1 and μ_2 are available, we can now compute the expected one-hop delay of a sensor

node whose one-hop distance is W and whose distance to the sink is l' by (6.13). Note that (6.13)'s expression assumes that the relaying sensor traffic arrival rate is Poisson. This Poisson approximation has been shown to be fairly accurate [53, 72].

Compute Delay for a Given Path of Length m

With the per-hop delay computed, in this section, we finally can compute $\tau(m)$, which is the delay of a m-hop paths by aggregating the delay along all the m hops.

To compute the aggregation, note that when the sensor packet has already traveled X distance, i.e., the current forwarder is $l' = l - X$ distance to the sink, and the next hop distance is W, the expected one-hop delay can be computed according to (6.13), (6.16), (6.17) and (6.19). Since based on (6.13), (6.16), (6.17) and (6.19), g_1 is a function of X and W, we denote it as $g_1(X, W)$. Recall that after k hops the total progress, i.e., X, to the sink can be approximated as a Gaussian distribution variable with mean kv and variance $k\sigma^2$. Then, the expected delay of a packet at the $(k+1)$th hop after travelling k hops is

$$\tau_1(k) = \begin{cases} \dfrac{\int_{-\infty}^{\infty}\int_0^{r_c} g_1(x,w) f_W(w) e^{-\frac{(x-kv)}{2k\sigma^2}} dw dx}{\sigma\sqrt{2\pi k}}, k > 0, \\ \displaystyle\int_0^{r_c} g_1(0,w) f_W(w) dw, k = 0, \end{cases} \tag{6.20}$$

where $f_W(w)$ is the PDF of one-hop distance W and is a function that is determined by the selection of sensor routing protocol. Therefore, the expected delay for a m-hop path is

$$\tau(m) = \sum_{k=0}^{m-1} \tau_1(k). \tag{6.21}$$

Combining (6.7), (6.20) and (6.21), we can compute the expected delay of a sensor node which is l distant from the sink.

6.5 Analysis of Routing Schemes

Note that in Section 6.4's analysis of delay, we assumed that the distributions of the one-hop progress, i.e., Y, to the sink and one-hop distance, i.e., W, are known since they can be derived based on the specific routing schemes used in sensor networks. To demonstrate this point, we next show how to derive the one-hop distance and one-hop progress of a routing scheme using an example. This example routing protocol is Most Forwarding progress within Radius (MFR) [47], which is a popular geographic routing scheme in sensor networks. In MFR, the current node greedily forwards packets to the neighbor whose progress along the straight line from the current node to the destination node is the largest. As shown in

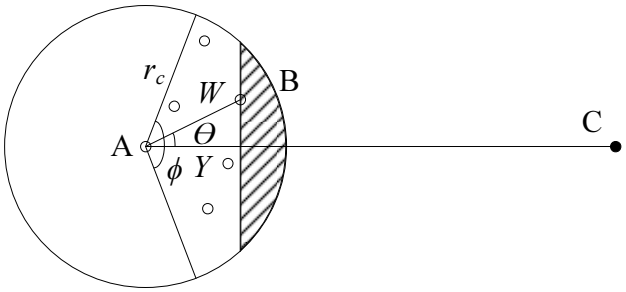


Figure 6.4: Most forwarding progress within radius

Figure 6.4, node A is sending packets to sink C, and by MFR it forwards packets to node B which has the longest progress to node C along the straight line from node A to node C.

Define the forwarding sector as the sector region within which the current forwarding sensor looks for the next hop relay node. Denote the angle of forwarding sector of MFR as ϕ, and the angle between the forwarding direction and the straight line between the current forwarding node and sink node as Θ. Without loss of generality, we assume $0 < \phi \leq \pi$. Denote the progress of one-hop MFR as $0 \leq Y \leq r_c$. Since MFR chooses the node with the most forwarding progress as the relaying node, there should be no sensor nodes in the shadowed region in Figure 6.4. The area of this shadowing region can be computed as follows

$$A_{MFR}(Y) = \begin{cases} \frac{r_c^2\phi}{2} - Y^2 \tan\frac{\phi}{2}, 0 \leq Y \leq r_c \cos\frac{\phi}{2} \\ r_c^2 \arccos\frac{Y}{r_c} - Y\sqrt{r_c^2 - Y^2}, r_c\cos\frac{\phi}{2} < Y \leq r_c. \end{cases} \tag{6.22}$$

Its derivative is

$$A'_{MFR}(Y) = \begin{cases} -2y\tan\frac{\phi}{2}, 0 \leq Y \leq r_c\cos\frac{\phi}{2} \\ -2\sqrt{r_c^2 - Y^2}, r_c\cos\frac{\phi}{2} < Y \leq r_c. \end{cases} \tag{6.23}$$

Then, the probability that $Y \leq y$ can be expressed as:

$$\begin{aligned} \Pr\{Y < y\} =& e^{-\gamma_1 A_{MFR}(y)} \\ =& \begin{cases} e^{-\gamma_1(\frac{r_c^2\phi}{2} - y^2\tan\frac{\phi}{2})}, 0 \leq y \leq r_c\cos\frac{\phi}{2} \\ e^{-(r_c^2\arccos\frac{y}{r_c} - y\sqrt{r_c^2-y^2})}, r_c\cos\frac{\phi}{2} < y \leq r_c. \end{cases} \end{aligned} \tag{6.24}$$

It follows that the PDF of Y is

$$\begin{aligned} f_Y(y) =& \frac{-\gamma_1 A'_{MFR}(y) e^{-\gamma_1 A_{MFR}(y)}}{\Pr\{Y < r_c\} - \Pr\{Y < 0\}} \\ =& \begin{cases} \frac{2\gamma_1 y\tan\frac{\phi}{2} e^{-\gamma_1(\frac{r_c^2\phi}{2} - y^2\tan\frac{\phi}{2})}}{\Pr\{Y<r_c\}-\Pr\{Y<0\}}, 0 \leq y \leq r_c\cos\frac{\phi}{2} \\ \frac{2\gamma_1\sqrt{r_c^2-y^2} e^{-\gamma_1(r_c^2\arccos\frac{y}{r_c} - y\sqrt{r_c^2-y^2})}}{\Pr\{Y<r_c\}-\Pr\{Y<0\}}, r_c\cos\frac{\phi}{2} < y \leq r_c. \end{cases} \end{aligned} \tag{6.25}$$

The mean and variance of Y can be computed as:

$$E\{Y\} = \int_0^{r_c} y f_Y(y) dy \tag{6.26}$$

$$E\{Y^2\} = \int_0^{r_c} y^2 f_Y(y) dy \tag{6.27}$$

$$Var\{Y\} = E\{Y^2\} - (E\{Y\})^2. \tag{6.28}$$

Note that the one-hop distance $W = \frac{Y}{\Theta}$, where Θ is the angle between the straight line from the current node to the next hop node and the straight line from current node to the sink node. It is easy to shown that Θ is uniformly distributed within $[-\frac{\phi}{2}, \frac{\phi}{2}]$. Then, the probability that $W \leq w$ is

$$\begin{aligned}
\Pr\{W \leq w\} =& \iint\limits_{\frac{y}{\cos\theta} \leqslant w} \frac{1}{\phi} f_Y(y) d\theta dy \\
=& 2 \int_{w\cos(\frac{\phi}{2})}^{w} \int_0^{\arccos(\frac{y}{w})} \frac{1}{\phi} f_Y(y) d\theta dy + 2 \int_0^{w\cos(\frac{\phi}{2})} \int_0^{\frac{\phi}{2}} \frac{1}{\phi} f_Y(y) d\theta dy \\
=& \frac{2}{\phi} \int_{w\cos(\frac{\phi}{2})}^{w} \arccos(\frac{y}{w}) f_Y(y) dy + \int_0^{w\cos(\frac{\phi}{2})} f_Y(y) dy.
\end{aligned} \tag{6.29}$$

Its derivative is

$$\Pr\{W \leq w\}' = \frac{2}{\phi} \int_{w\cos(\frac{\phi}{2})}^{w} \frac{y}{w\sqrt{w^2 - y^2}} f_Y(y) dy. \tag{6.30}$$

Thus, the PDF of W is

$$f_W(w) = \frac{\Pr\{W \leq w\}'}{\int_0^{r_c} \Pr\{W \leq w\}' dw}. \tag{6.31}$$

Equation (6.25), (6.26), (6.27), (6.28) and (6.31) essentially are expressions that can be used to compute the distribution of one-hop progress and one-hop distance for MFR routing protocol. Plugging these distribution expressions into the delay analysis results in Section 6.4, we can obtain the convergecast delay under MFR routing systems. Other routing protocols can also be analyzed in similar ways as shown in [53].

6.6 Simplified Linear Approximation Model

We have shown how to compute expected delay $T(l)$ based on all possible path probabilities and path delays by (6.5). Since this computation involves multiple integrations and may be computationally intense, we next show a simplified linear approximation model that can be used to estimate the delay.

The basic idea is to decompose the expected delay $T(l)$ into delays of multiple hops. The total number of hops and hop distance estimation is based on the expected hop progress $E\{Y\}$ to compute the expected delay $T(l)$. Different from the probabilistic model where the distribution function of hop progress are used to estimate the delay, in this simplified linear model only expected hop progress is used. In high-density sensor networks, the variance of hop progress is expected to be small; hence, the approximation is likely to be accurate.

Note that in MFR, the expected progress of each hop is $E\{Y\}$ except that of the last hop, and in the last hop MFR forwards the packet directly to the sink. Therefore, the expected delay $T(l)$ can be approximated as the sum of the last hop delay and all the previous hop delays. Each previous hop progress is $E\{Y\}$, and the last hop progress is $l - \lfloor \frac{l}{E\{Y\}} \rfloor E\{Y\}$. Therefore, we have

$$T(l) = \sum_{i=1}^{\lfloor \frac{l}{E\{Y\}} \rfloor} g_1(l - iE\{Y\}, E\{Y\}) + g_1(\left\lfloor \frac{l}{E\{Y\}} \right\rfloor E\{Y\}, l - \left\lfloor \frac{l}{E\{Y\}} \right\rfloor E\{Y\}), \quad (6.32)$$

where $g_1(x, w)$ is the expected next hop delay when the progress from the source node to the sink node is x and the next hop distance is w.

6.7 Simulations

In our previous mathematical analysis, we have made several approximations that may affect the accuracy of our analysis. When computing the path probability $P(m)$ in (6.7), the central limit theorem is used. The central limit theorem approximation is guaranteed to be accurate only when the distance l is much larger than sensor communication range r_c. Also, when computing the sensor traffic arrival rate λ_1, we underestimate it by assuming all the sensor node traffic arrival rate is the same and equals the traffic arrival rate of the sender of the current hop. This assumption is accurate when the sender of current hop is far away form the sink, but there is a chance that its approximation accuracy may decrease when the sender of current hop is close to the sink. Therefore, it is expected that the gap between the theoretical estimation and the actual delay becomes small, when the network becomes large and when distance l becomes large. To understand how these approximation may affect the accuracy of our analysis, in this section, we do simulations in NS 2.34 [60], and compare the theoretical curves with simulation curves.

We do two sets of simulations. In the first set of simulations, the simulation region is a $400 \times 400m^2$ square region. 128 WiFi nodes are uniformly deployed in the square region. Each WiFi node is paired with another WiFi node within its communication range. WiFi nodes' transmission power is 100 mW, communication range is set to 100 m, interference capture threshold T_{CP} is 10, and channel bandwidth is set to 1 Mbps. The physical layer WiFi data packet length is set to 572 bytes. The WiFi arrival process follows a Poisson arrival process. We simulate 3 s and 5 s expected WiFi packet arrival interval, respectively. The sensor sink node is placed in the center of the region. 960 sensor nodes are uniformly deployed in a disk region centered at the sink node with radius 200 m. Each non-sink sensor node finds its next hop by MFR, and the forwarding sector angle $\phi = \pi$. Sensors' transmission power is set to 1 mW, transmission range is set to 25 m, interference capture threshold T_{CP} is 10, and the channel bandwidth is set to 250 Kbps. The packet generation process of each sensor node follows a Poisson arrival process, where the expected packet generation interval is set to 80 s. We simulate 50 bytes and 65 bytes physical layer sensor packet length, respectively. We simulate 10 random WiFi topologies overlapped with 10 random sensor topologies. Each simulation runs 100 s. For each sensor node, we record its distance to the sensor sink, and the delay of packets generated by this sensor node. We collect the delay data for all sensor nodes for all possible topologies. We plot the delay as a function of distance to the sink node. To capture the delay's relationship with distance, we divide the simulated sensor network region into multiple rings with a thickness of 12.5 m and the center at the sink node, and compute the average delay of nodes in each of the rings. The simulation results are shown in Figure 6.5, 6.6, 6.7 and 6.8.

In Figure 6.5 and 6.6, we compare the convergecast delay derived from our probabilistic theoretical model with simulation results. The convergecast delay for 3 s expected WiFi packet arrival interval are shown in Figure 6.5, and the convergecast delay for 5 s expected WiFi packet arrival interval are shown in Figure 6.6. As shown in both figures, the gap between theoretical results and simulation results is fairly small, validating the correctness of our analyses.

In Figure 6.7 and 6.8, we compare the convergecast delays derived from the simplified linear approximation model with simulation results. The convergecast delays for 3 s expected WiFi packet arrival interval are shown in Figure 6.7, and that for 5 s expected WiFi packet arrival interval are shown in Figure 6.8. Because only the expected progress per hop is used in the simplified model, the theoretical estimation may not be as accurate as the probabilistic model. However, as shown in both figures, the simplified linear approximation model serves as an acceptable estimation for the simulated delay.

Next, to validate our analyses in a relatively larger network, we do the second set of simulation in another setting. In the second setting, we change the simulation region to a $450 \times 450m^2$ square region. Sensor nodes are uniformly deployed in a disk region centered at the sink node with radius 225 m. The numbers of WiFi nodes and sensor nodes are increased to 162 and 1100, respectively. For WiFi traffic, we simulate 4 s and 6 s expected WiFi packet arrival intervals. For sensor traffic, we simulate 50 bytes and 60 bytes physical

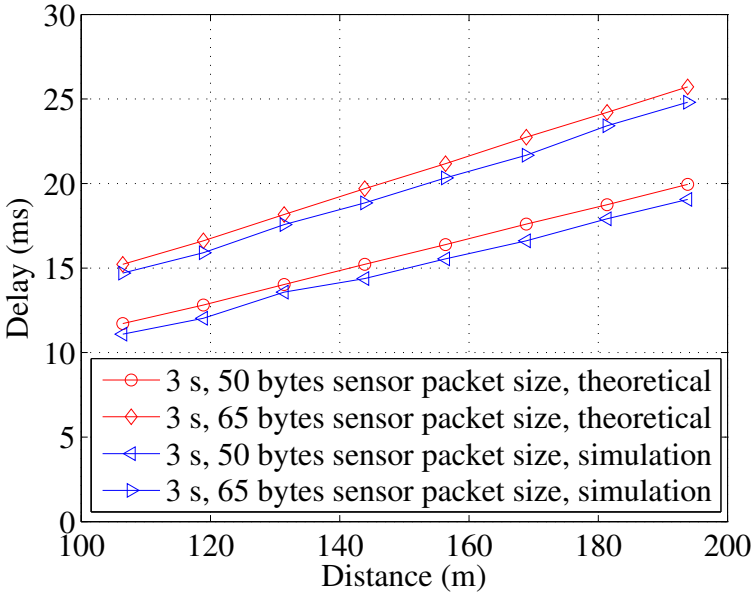


Figure 6.5: Comparison of theoretical delay and simulated delay when 960 sensor nodes and 128 WiFi nodes coexist, WiFi packet arrival interval is 3 s

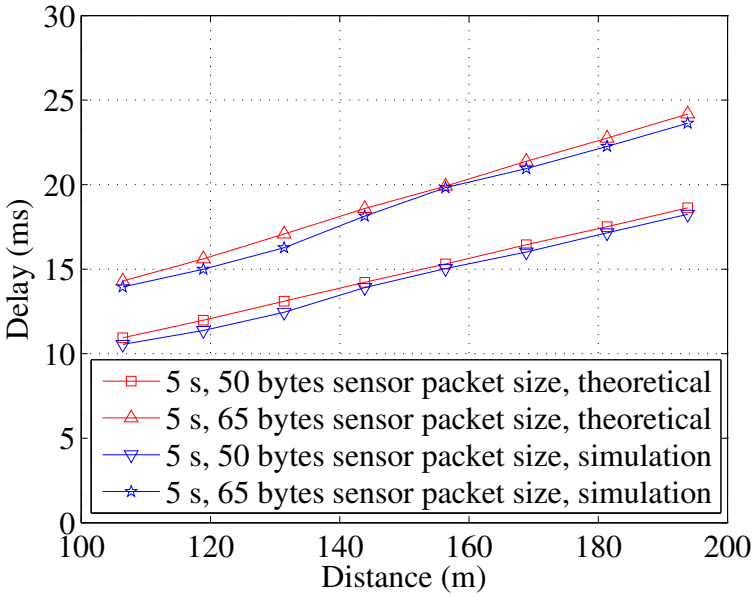


Figure 6.6: Comparison of theoretical delay and simulated delay when 960 sensor nodes and 128 WiFi nodes coexist, WiFi packet arrival interval is 5 s

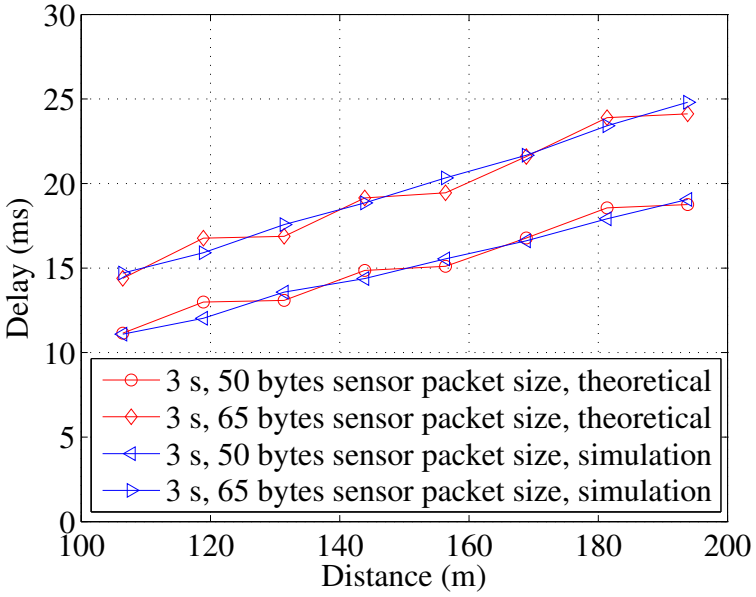


Figure 6.7: Comparison of simplified theoretical delay and simulation delay when 960 sensor nodes and 128 WiFi nodes coexist, WiFi packet arrival interval is 3 s

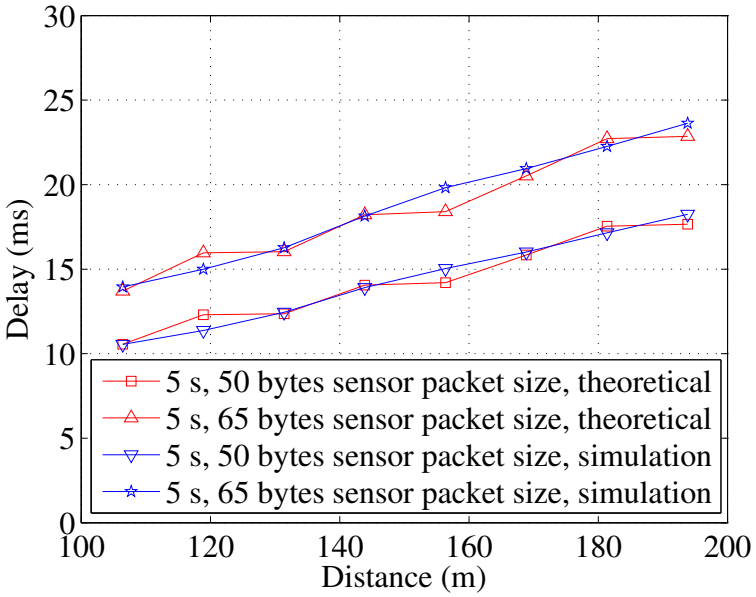


Figure 6.8: Comparison of simplified theoretical delay and simulation delay when 960 sensor nodes and 128 WiFi nodes coexist, WiFi packet arrival interval is 5 s

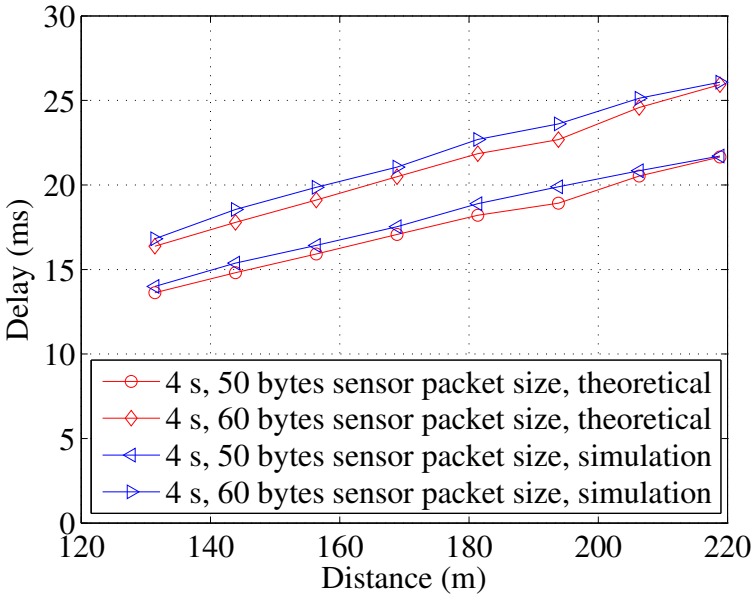


Figure 6.9: Comparison of theoretical delay and simulated delay when 1100 sensor nodes and 162 WiFi nodes coexist, WiFi packet arrival interval is 4 s

layer sensor packet length. Other parameter settings remain unchanged. The simulation results are shown in Figure 6.9, 6.10, 6.11 and 6.12.

In Figure 6.9 and 6.10, we compare the convergecast delay derived from the probabilistic theoretical model with simulation results. The convergecast delay for 4 s expected WiFi packet arrival interval are shown in Figure 6.9, and the convergecast delay for 6 s expected WiFi packet arrival interval are shown in Figure 6.10. As shown in both figures, the gap between theoretical results and simulation results is still fairly small, validating the correctness of our analyses. Compared with the performance of 960 sensor nodes, the gap of 1100 nodes is even smaller. This shows that our model becomes more accurate, when the network scale increases. As expected, when the distance l becomes bigger, the gap between theoretical results and simulation results becomes smaller as shown in Figure 6.9 and 6.10.

In Figure 6.11 and 6.12, we compare the convergecast delays derived from the simplified linear approximation model with simulation results. The convergecast delays for 4 s expected WiFi packet arrival interval are shown in Figure 6.11, and that for 6 s expected WiFi packet arrival interval are shown in Figure 6.12. Because only the expected progress per hop is used in the simplified model, the theoretical estimation may not be as accurate as the probabilistic model. However, as shown in both figures, the simplified linear approximation model serves as an acceptable estimation for the simulated delay.

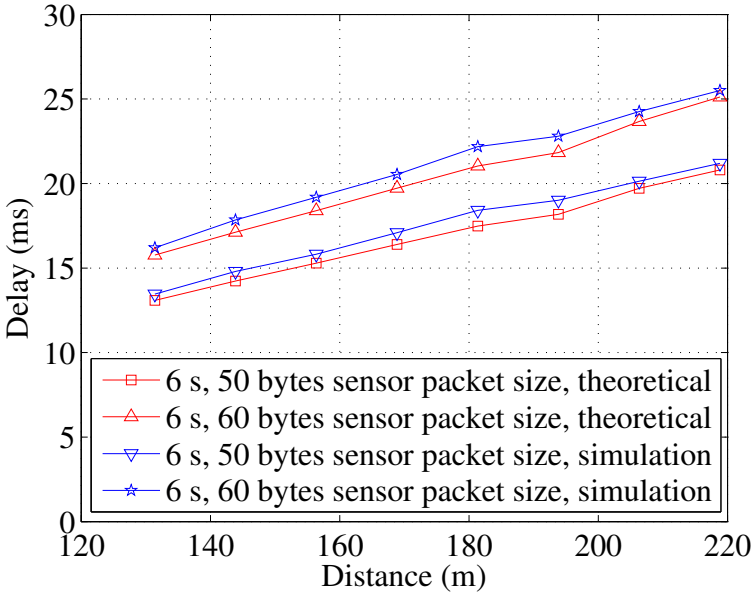


Figure 6.10: Comparison of theoretical delay and simulated delay when 1100 sensor nodes and 162 WiFi nodes coexist, WiFi packet arrival interval is 6 s

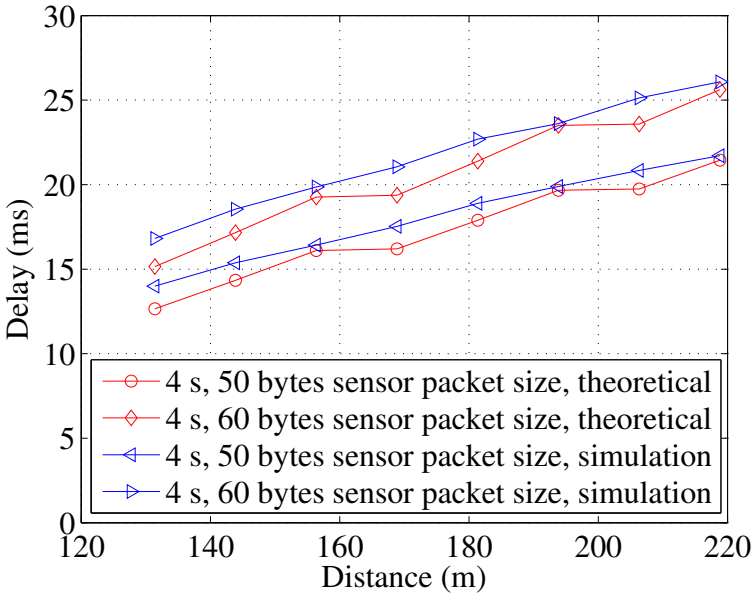


Figure 6.11: Comparison of simplified theoretical delay and simulation delay when 1100 sensor nodes and 162 WiFi nodes coexist, WiFi packet arrival interval is 4 s

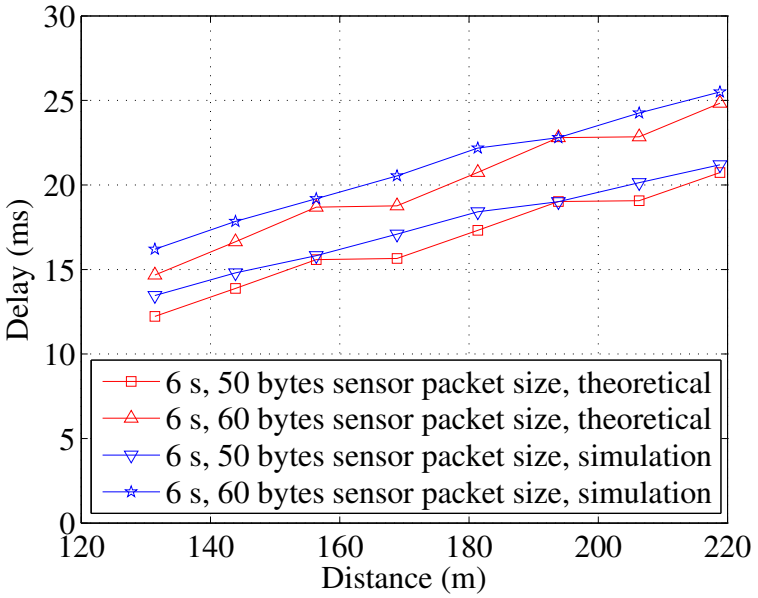


Figure 6.12: Comparison of simplified theoretical delay and simulation delay when 1100 sensor nodes and 162 WiFi nodes coexist, WiFi packet arrival interval is 6 s

6.8 Conclusions

In this chapter, we study the delay performance of sensor networks in the presence of WiFi interference. We model the delay analysis problem as a M/G/1 two priority preemptive repeat identical queueing system. Using queueing theory and probabilistic analysis, we derive the expected packet forwarding delay for a given distance from a certain sensor node to the sink. Further, we develop a simplified linear delay model, which is computationally less intensive and can also serve as delay estimation with acceptable accuracy. NS2 simulations show that our theoretical results are close to the simulation results, validating the correctness of our work.

Bibliography

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, pp. 2127–2159, 2006.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Comput. Netw. ISDN Syst.*, vol. 47, no. 4, pp. 445 – 487, 2005.

[3] E. Baccelli, P. Jacquet, B. Mans, and G. Rodolakis, "Information propagation speed in bidirectional vehicular delay tolerant networks," in *Proc. of IEEE INFOCOM*, 2011.

[4] R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel, "A survey on routing metrics," ETH Zürich, Tech. Rep. TIK Report 262, 2007.

[5] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*. Wiley-Interscience, 2006.

[6] J. Bondy and U. Murty, *Graph Theory*. Springer, 2008.

[7] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, pp. 483 – 502, 2002.

[8] W. Chang, "Preemptive priority queue," *Operations Research*, vol. 13, pp. 820–827, 1965.

[9] R.-R. Chen and X. Liu, "Delay performance of threshold policies for dynamic spectrum access," *IEEE Transaction on Wireless Communications*, vol. 10, pp. 2283–2293, 2011.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. Cambridge, MA, USA: MIT Press, 2001.

[11] J. D. P. Gaver, "A waiting line with interrupted service, including priorities," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 42, pp. 73–90, 1962.

[12] C. Florens, M. Franceschetti, and R. J. McEliece, "Lower bounds on data collection time in sensory networks," *IEEE JSAC*, vol. 22, pp. 1110–1120, 2004.

[13] C. Florens, M. Sharif, and R. J. McEliece, "Random sensory networks: A delay analysis," *IEEE Transactions on Information Theory*, vol. 55, pp. 1650–1664, 2009.

[14] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 5, pp. 66 – 77, 1983.

[15] A. Gibbons, *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[16] Q. Gu, P. Liu, W.-C. Lee, and C.-H. Chu, "KTR: An efficient key management scheme for secure data access control in wireless broadcast services," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, pp. 188 – 201, 2009.

[17] S. Guo and O. W. Yang, "Energy-aware multicasting in wireless ad hoc networks: A survey and discussion," *Computer Communications*, vol. 30, pp. 2129 – 2148, 2007.

[18] H. Haanpää, A. Schumacher, and P. Orponen, "Distributed algorithms for lifetime maximization in sensor networks via Min-Max spanning subgraphs," *Wireless Networks*, vol. 16, pp. 875 – 887, 2010.

[19] C. Han and Y. Yang, "The information propagation speed upper bound in cognitive radio networks," in *Proc. of IEEE GLOBECOM*, 2010.

[20] P. Helman, B. M. E. Moret, and H. D. Shapiro, "An exact characterization of greedy structures," *SIAM Journal on Discrete Mathematics*, vol. 6, pp. 274 – 283, 1993.

[21] J. Hou, B. Chang, D.-K. Cho, and M. Gerla, "Minimizing 802.11 interference on ZigBee medical sensors," in *Proc. of BodyNets*, 2009.

[22] C.-J. Huang, W. K. Lai, Y.-T. Chuang, and S.-Y. Hsiao, "A dynamic alternate path QoS enabled routing scheme in mobile ad hoc networks," *International Journal of Wireless Information Networks*, vol. 14, pp. 1 – 16, 2007.

[23] S. Huang, X. Liu, and Z. Ding, "Distributed power control for cognitive user access based on primary link control feedback," in *Proc. of IEEE INFOCOM*, 2010.

[24] IEEE Computer Society, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Std., 2006.

[25] ——, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std., 2007.

[26] P. Jacquet, B. Mans, P. Muhlethaler, and G. Rodolakis, "Opportunistic routing in wireless ad hoc networks: upper bounds for the packet propagation speed," *IEEE JSAC*, vol. 27, pp. 1192–1202, 2009.

[27] P. Jacquet, B. Mans, and G. Rodolakis, "Information propagation speed in mobile and delay tolerant networks," *IEEE Transactions on Information Theory*, vol. 56, pp. 5001–5015, 2010.

[28] D. B. Johnson and D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic Publishers, 1996, ch. 5, pp. 153–181.

[29] D. Jungnickel, *Graphs, Networks and Algorithms*. Springer, 2008.

[30] I. Kang and R. Poovendran, "Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks," *Mob. Netw. Appl.*, vol. 10, no. 6, pp. 879 – 896, 2005.

[31] J. Kleinberg and va Tardos, *Algorithm Design*. Addison Wesley, 2005.

[32] W. K. Lai, S.-Y. Hsiao, and Y.-C. Lin, "Adaptive backup routing for ad-hoc networks," *Computer Communications*, vol. 30, pp. 453 – 464, 2007.

[33] A. Laourine, S. Chen, and L. Tong, "Queuing analysis in multichannel cognitive spectrum access: A large deviation approach," in *Proc. of IEEE INFOCOM*, 2010.

[34] S.-J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," in *Proc. IEEE WCNC*, 2000.

[35] B. Liang and Z. J. Haas, "Optimizing route-cache lifetime in ad hoc networks," in *Proc. of Infocom*, 2003.

[36] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi interference in low power ZigBee networks," in *Proc. ACM SenSys*, 2010.

[37] M. Lu and J. Wu, "Opportunistic routing algebra and its applications," in *Proc. IEEE INFOCOM*, 2009.

[38] J. Moy, "OSPF Version 2," RFC 2328, April 1998.

[39] A. Nasipuri, R. Castaeda, and S. R. Das, "Performance of multipath routing for on-demand protocols in mobile ad hoc networks," *Mobile Networks and Applications*, vol. 6, pp. 339 – 349, 2001.

[40] A. Navarra, I. Caragiannis, M. Flammini, C. Kaklamanis, and R. Klasing, *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless, and Ad Hoc Networks*. Springer Berlin Heidelberg, 2010, ch. Energy Consumption Minimization in Ad Hoc Wireless and Multi-interface Networks, pp. 335–355.

[41] I. Papadimitriou and L. Georgiadis, "Minimum-energy broadcasting in multi-hop wireless networks using a single broadcast tree," *Mobile Networks and Applications*, vol. 11, pp. 361 – 375, 2006.

[42] G. Parissidis, M. Karaliopoulos, R. Baumann, T. Spyropoulos, and B. Plattner, *Guide to Wireless Mesh Networks*. Springer London, 2009, ch. Routing Metrics for Wireless Mesh Networks, pp. 199–230.

[43] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1997.

[44] C. E. Perkins, Ed., *Ad hoc networking*. Addison Wesley, 2001.

[45] P. Popovski, H. Yomo, K. Nishimori, R. D. Taranto, and R. Prasad, "Opportunistic interference cancellation in cognitive radio systems," in *Proc. of IEEE DySPAN*, 2007.

[46] J. Qadir, C. T. Chou, A. Misra, and J. G. Lim, "Minimum latency broadcasting in multiradio, multichannel, multirate wireless meshes," *IEEE Transactions on Mobile Computing*, vol. 8, pp. 1510 – 1523, 2009.

[47] S. Rhrup, *Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*. Bentham Science, 2009, ch. Theory and Practice of Geographic Routing.

[48] Y. Shi, C. Jiang, Y. T. Hou, and S. Kompella, "On capacity scaling law of cognitive radio ad hoc networks," in *Proc. IEEE ICCCN*, 2011.

[49] A. Sikora and V. F. Groza, "Coexistence of IEEE 802.15.4 with other systems in the 2.4 GHz-ISM-Band," in *Proc. of Instrumentation and Measurement Technology Conference*, 2005.

[50] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet," in *Proc. IEEE INFOCOM*, 2001.

[51] ——, "Network routing with path vector protocols: theory and applications," in *Proc. ACM SIGCOMM*, 2003.

[52] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *International Journal of Network Management*, vol. 11, pp. 3 – 30, 2001.

[53] R. Subramanian and F. Fekri, "Analysis of latency and related tradeoffs in distributed sensor networks," in *Proc. IEEE SECON*, 2007.

[54] L. Sun and W. Wang, "On distribution and limits of information dissemination latency and speed in mobile cognitive radio networks," in *Proc. of IEEE INFOCOM*, 2011.

[55] S. Waharte and R. Boutaba, "Tree-based wireless mesh network architecture: Topology analysis," in *Proc. of the First International Workshop on Wireless Mesh Networks (MeshNets)*, 2005.

[56] S. Wang, J. Zhang, and L. Tong, "Delay analysis for cognitive radio networks with random access: A fluid queue view," in *Proc. of IEEE INFOCOM*, 2010.

[57] W. Wang and M. Zhao, "Joint effects of radio channels and node mobility on link dynamics in wireless networks," in *Proc. of Infocom*, 2008.

[58] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proc. IEEE INFOCOM*, 2000.

[59] wiki, "ns-2." [Online]. Available: http://nsnam.isi.edu/nsnam/

[60] ——, "NS2," Nov. 2011. [Online]. Available: http://nsnam.isi.edu/nsnam

[61] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz, "Primary user behavior in cellular networks and implications for dynamic spectrum access," *IEEE Communications Magazine*, vol. 47, pp. 88–95, 2009.

[62] Y. Xu and W. Wang, "The limit of information propagation speed in large-scale multi-hop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, pp. 209–222, 2011.

[63] Y. Yang and J. Wang, "Design guidelines for routing metrics in multihop wireless networks," in *Proc. IEEE INFOCOM*, 2008.

[64] C. Yin, L. Gao, and S. Cui, "Scaling laws for overlaid wireless networks: A cognitive radio network versus a primary network," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1317–1329, 2010.

[65] D. Yuan, J. Bauer, and D. Haugland, "Minimum-energy broadcast and multicast in wireless networks: An integer programming approach and improved heuristic algorithms," *Ad Hoc Networks*, vol. 6, pp. 696 – 717, 2008.

[66] R. Zhang and Y.-C. Liang, "Exploiting hidden power-feedback loops for cognitive radio," in *Proc. of IEEE DySPAN*, 2008.

[67] X. Zhang and K. G. Shin, "A case for the coexistence of heterogeneous wireless networks," in *Proc. of the 3rd ACM Wireless Student Workshop*, 2011.

[68] ——, "Enabling coexistence of heterogeneous wireless systems: Case for ZigBee and WiFi," in *Proc. ACM MobiHoc*, 2011.

[69] G. Zhao, G. Y. Li, and C. Yang, "Proactive detection of spectrum opportunities in primary systems with power control," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 4815–4823, 2009.

[70] M. Zhao and W. Wang, "A unified mobility model for analysis and simulation of mobile wireless networks," *ACM-Springer Wireless Networks (WINET)*, vol. 15, pp. 365 – 389, 2007.

[71] R. Zheng, "Asymptotic bounds of information dissemination in power-constrained wireless networks," *IEEE Transaction on Wireless Communications*, vol. 7, pp. 251–259, 2008.

[72] J. Zhou and K. Mitchell, "A scalable delay based analytical framework for csma/ca wireless mesh networks," *Computer Networks*, vol. 52, pp. 304–318, 2010.

Appendix A

Proof of Lemma 9

Lemma 9. *The function*

$$f(n_i) = n_i(1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}) \tag{A.1}$$

is monotonically increasing with respect to n_i.

Proof. We relax the original function to a continuous function of n_i, and then calculate its derivative.

$$\begin{aligned} f'(n_i) &= (1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}) - n_i[\sum_{k=1}^{i-1} \frac{n_k}{(n_k + n_i)^2} \prod_{j=0, j\neq k}^{i-1} \frac{n_j}{n_j + n_i}] \\ &= 1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} - \sum_{k=1}^{i-1} \frac{n_i}{n_k + n_i} \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i}. \end{aligned} \tag{A.2}$$

To complete the proof, we need to show that

$$\begin{aligned} 1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} - \sum_{k=1}^{i-1} \frac{n_i}{n_k + n_i} \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} > 0, &\Longleftrightarrow 1 + \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} > \sum_{k=1}^{i-1} \frac{n_i}{n_k + n_i} \prod_{j=0}^{i-1} \frac{n_j}{n_j + n_i} \\ &\Longleftrightarrow \prod_{j=0}^{i-1} (1 + \frac{n_i}{n_j}) + 1 > \sum_{k=0}^{i-1} \frac{n_i}{n_k + n_i}. \end{aligned}$$

The above inequality is true since

$$\prod_{j=0}^{i-1} (1 + \frac{n_i}{n_j}) + 1 > 1 + \sum_{j=0}^{i-1} \frac{n_i}{n_j} + 1 > \sum_{j=0}^{i-1} \frac{n_i}{n_j} > \sum_{k=0}^{i-1} \frac{n_i}{n_k + n_i}.$$

Therefore, $f'(n_i) > 0$. That is $f(n_i)$ is a monotonically increasing function of n_i. □

Appendix B

Proof of Lemma 10

Lemma 10. *Using the optimal policies in Propositions 1 and 2, the numerator of the objective function in* (4.16) *is an increasing, discrete, concave function with respect to path cache size* k.

Proof. Denote the k paths in $\mathcal{I}$ as $\{p_1, p_2, \cdots, p_k\}$ in an increasing order of their hop counts. According to Lemma 3, the numerator of the objective function in (4.16) is

$$E[T_{\mathcal{I}}^{\max}]|_{|\mathcal{I}|=k} = E\big[\max_{0\le i\le k}\{T_i\}\big]. \tag{B.1}$$

Now increase the number of cached paths to $k+1$ by adding an additional path p_{k+1} in $\mathcal{I}$. Note that the hop counts of paths p_{k+1} and p_k satisfy $n_{k+1} \ge n_k$. $E[T_{\mathcal{I}}^{\max}]$ becomes

$$\begin{aligned} E[T_{\mathcal{I}}^{\max}]|_{|\mathcal{I}|=k+1} &= E\big[\max\{\max_{0\le i\le k}\{T_i\}, T_{k+1}\}\big] \\ &= a_{k+1}E\Big[\big(T_{k+1} - \max_{0\le i\le k}\{T_i\}\big)^+\Big] + E\big[\max_{0\le i\le k}\{T_i\}\big], \end{aligned} \tag{B.2}$$

where $a_{k+1} = P\{T_{k+1} > \max_{0\le i\le k}\{T_i\}\}$ and $(f(x))^+$ denotes

$$(f(x))^+ = \begin{cases} f(x), & f(x) \ge 0, \\ 0, & f(x) < 0. \end{cases} \tag{B.3}$$

Define $U(k+1) := E[T_{\mathcal{I}}^{\max}]|_{|\mathcal{I}|=k+1} - E[T_{\mathcal{I}}^{\max}]|_{|\mathcal{I}|=k}$. We have

$$U(k+1) = a_{k+1}E\Big[\big(T_{k+1} - \max_{0\le i\le k}\{T_i\}\big)^+\Big]. \tag{B.4}$$

Clearly, $U(k+1) > 0$. Hence, $E[T_{\mathcal{I}}^{\max}]$ is an increasing function of k. In addition, $E[T_{\mathcal{I}}^{\max}]$ is a concave function if we can show that $U(k+1)$ is a decreasing function.

Based on (4.15), $a_{k+1} = \prod_{j=0}^{k} \frac{n_j}{n_j+n_{k+1}}$. Since $n_k \leq n_{k+1}$, a_{k+1} decreases when k increases. In addition, note that

$$\begin{aligned} E\Big[\big(T_{k+1} - \max_{0\leq i\leq k}\{T_i\}\big)^+\Big] =& E\Big[\big(T_{k+1} - \max\{T_k, \max_{0\leq i\leq k-1}\{T_i\}\}\big)^+\Big] \\ <& E\Big[\big(T_{k+1} - \max_{0\leq i\leq k-1}\{T_i\}\big)^+\Big] \\ \leq& E\Big[\big(T_k - \max_{0\leq i\leq k-1}\{T_i\}\big)^+\Big]. \end{aligned} \tag{B.5}$$

Therefore, $E\Big[\big(T_{k+1} - \max_{0\leq i\leq k}\{T_i\}\big)^+\Big]$ also decreases as k increases. Hence, based on (B.4), we can conclude that $U(k)$ decreases as k increases. Hence, the numerator of the objective function in (4.16) is an increasing, discrete, concave function with respect to k. □

Appendix C

Proof of Lemma 11

Lemma 11. *Using the optimal policies in Propositions 1 and 2, the denominator of the objective function in* (4.16) *can be approximated as a non-decreasing, discrete, convex function with respect to the path cache size k.*

Proof. Consider increasing the path cache size from k to $k+1$ by adding a path p_{k+1}. Note that the path lengths of paths p_{k+1} and p_k satisfy $n_{k+1} \geq n_k$. This addition of p_{k+1} increases the denominator of the objective function in (4.16) by

$$V(k+1) = n_{k+1}(1 + \prod_{j=1}^{k} \frac{n_j}{n_j + n_{k+1}}) \approx n_{k+1}. \tag{C.1}$$

The approximation holds for large k since $\prod_{j=1}^{k} \frac{n_j}{n_j+n_{k+1}} \ll 1$ for large k. In addition, since $n_{k+1} \geq n_k$, $V(k+1)$ in (C.1) can be approximated as an non-decreasing function with respect to k. Hence, the denominator or the objective function can be approximated as a non-decreasing, discrete, convex function. □

Appendix D

Validation of Proposition 3

The proof of Proposition 3 is based on Lemma 11 in the Appendix, which uses some approximation to derive its result. In this section, we validate that this approximation does not affect the correctness of Proposition 3 by experiments.

In the experiments, the total number of nodes is 200, and the path lifetime parameter λ is calculated by (4.8). We set $\bar{v} = 10$ m/s and $R = 250$ m. Note that the specific values of the average speed $\bar{v}$ and communication range R do not affect the general trend of the objective function value. The curves are calculated by (4.16). To validate the correctness of the objective function in (4.16), we also do numerical calculation of the objective. In the calculation, the path lifetime follows the exponential distribution with the same parameters as those used in the theoretical formulas. However, the numerical calculation of the objective function does not have the approximations in Lemma 11. The numerical calculated values are averaged over 10^4 trials.

We perform two experiments: the random hop count experiment, and the fixed hop count experiment. For the random hop count experiment, the path hop count is uniformly distributed within $[1, 20]$. The value of the objective function in (4.16) and its numerator and denominator with respect to the path cache size $|\mathcal{I}|$ are shown in Figure D.1. In the figures, the theoretical curves and the numerical curves matches perfectly, validating the correctness of our theoretical formulas. Also, as expected, the numerator is an increasing concave function and the denominator is an increasing convex function. The objective function value with respect to $|\mathcal{I}|$ is first increasing in a concave shape and then decreasing. Its shape is the same as Figure 4.2(b), validating the correctness of Proposition 3. For the fixed hop count experiment, the hop counts of all paths discovered by flooding-based route discovery are fixed at 2, 3, and 4 respectively. The experiment results are shown in Figure D.2. As shown in the figure, the theoretical curves matches perfectly with the numerical curves, validating the correctness of our theoretical formula for the objective function. Also, the objective function is concave and monotonically increasing with respect to $|\mathcal{I}|$. The shape is the same as Figure 4.2(a), validating the correctness of Proposition 3.

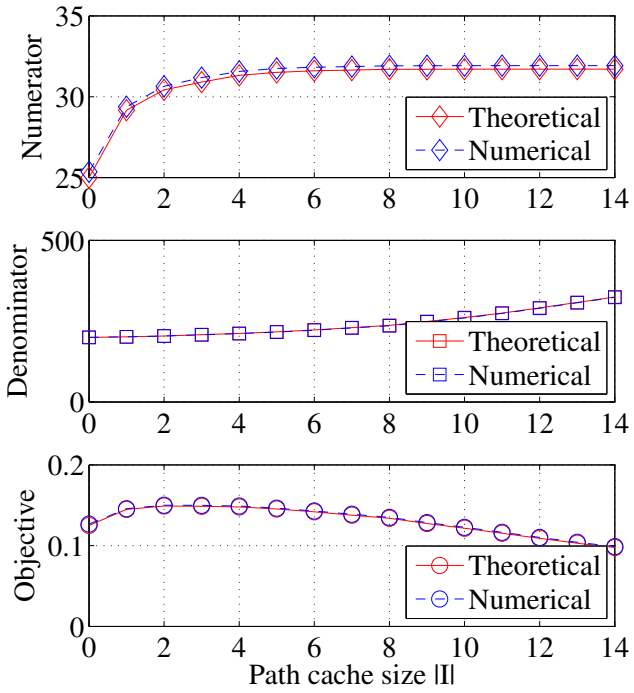


Figure D.1: Objective function value when the hop count is random

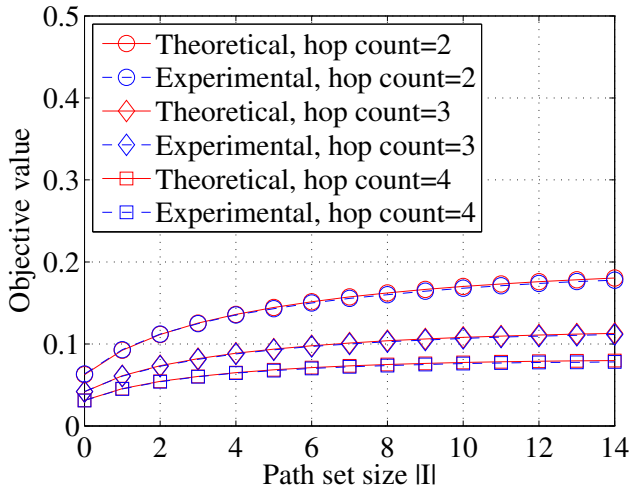


Figure D.2: Objective function value when the hop count is fixed