Optimization for Resource-Constrained Wireless Networks

by

Xi Fang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved January 2013 by the
Graduate Supervisory Committee:

Guoliang Xue, Chair
Sik-Sang Yau
Jieping Ye
Junshan Zhang

ARIZONA STATE UNIVERSITY

May 2013

ABSTRACT

Nowadays, wireless communications and networks have been widely used in our daily lives. One of the most important topics related to networking research is using optimization tools to improve the utilization of network resources. In this dissertation, we concentrate on optimization for resource-constrained wireless networks, and study two fundamental resource-allocation problems: 1) distributed routing optimization and 2) anypath routing optimization.

The study on the distributed routing optimization problem is composed of two main thrusts, targeted at understanding distributed routing and resource optimization for multihop wireless networks. The first thrust is dedicated to understanding the impact of full-duplex transmission on wireless network resource optimization. We propose two provably good distributed algorithms to optimize the resources in a full-duplex wireless network. We prove their optimality and also provide network status analysis using dual space information. The second thrust is dedicated to understanding the influence of network entity load constraints on network resource allocation and routing computation. We propose a provably good distributed algorithm to allocate wireless resources. In addition, we propose a new subgradient optimization framework, which can provide find-grained convergence, optimality, and dual space information at each iteration. This framework can provide a useful theoretical foundation for many networking optimization problems.

The study on the anypath routing optimization problem is composed of two main thrusts. The first thrust is dedicated to understanding the computational complexity of multi-constrained anypath routing and designing approximate solutions. We prove that this problem is NP-hard when the number of constraints

is larger than one. We present two polynomial time K-approximation algorithms. One is a centralized algorithm while the other one is a distributed algorithm. For the second thrust, we study directional anypath routing and present a cross-layer design of MAC and routing. For the MAC layer, we present a directional anycast MAC. For the routing layer, we propose two polynomial time routing algorithms to compute directional anypaths based on two antenna models, and prove their optimality based on the packet delivery ratio metric.

Dedicated to my family

ACKNOWLEDGEMENTS

First of all, I would like to express my sincerest gratitude to my advisor, Dr. Guoliang Xue, for his professional guidance, patience, support, and encouragement over the course of my PhD study. His immense knowledge and enthusiastic attitude towards research quality have always inspired me. I greatly appreciate his invaluable guidance in my research and his warm-hearted help in my personal life. I am very fortunate to have him as my advisor without whom I could not have my current achievement.

I would also like to thank all my committee members, Dr. Stephen Yau, Dr. Jieping Ye, and Dr. Junshan Zhang for their invaluable advice. I learned a lot from my committee members about research either by doing research with them or taking their classes.

I also thank my colleagues and friends: Dejun Yang, Dr. Satyajayant Misra, Lingjun Li, Xinxin Zhao, Dr. Jin Zhang, Jun Shen, Pritam Gundecha, Vishnu Kilari, Gabriel Silva, Xiang Zhang, and Ziming Zhao for the pleasant and inspiring discussion. Their friendship is a valuable experience for me.

Last but not least, I would like to thank my mother Xiaoling Chen, my father Zhi Fang, and my wife Xinhui Hu, who give me endless love and constant encouragement during my study at ASU and throughout my whole life. This dissertation is dedicated to them!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Nowadays, wireless communications and networks have been widely used in our daily lives. One of the most important topics related to networking research is using optimization tools to improve the utilization of network resources. In this dissertation, we concentrate on optimization for resource-constrained wireless networks and study two fundamental resource optimization problems for such wireless networks: 1) distributed routing optimization problem and 2) anypath routing optimization problem.

1.1 Optimizing Distributed Routing for Multihop Wireless Networks

1.1.1 Motivation

The fact that the wireless spectrum is a limited resource motivates us to investigate how to use wireless resources effectively. This part is composed of the following two main thrusts, targeted at understanding distributed routing and resource optimization for multihop wireless networks.

The first thrust is dedicated to understanding the impact of full-duplex transmission on wireless network resource optimization. A basic assumption in wireless communications is that a radio cannot transmit and receive on the same frequency at the same time. A recent research breakthrough on radio design challenges the half-duplex constraint, which has been used in wireless communication and network research for several decades. Choi *et al.* [17] and Jain *et al.* [43] implemented practical single-channel full-duplex radio systems. This encouraging breakthrough calls for new research efforts in the design of network layer protocols

and algorithms for wireless networks, and motivates us to re-think the network design, relaxing the basic assumption of half-duplexing, which has been used in wireless communication research during the last decade. Therefore we will study routing problems in emerging multihop wireless networks with full-duplex radios.

The second thrust is dedicated to understanding the influence of network entity load constraints on network resource allocation and routing computation. Most existing wireless routing algorithms aim at maximizing overall performance of a network (total throughput, total energy consumption, etc.) without carefully addressing the resource and social constraints, which is one of the main focuses of this thrust. In addition, routing in multihop wireless networks is challenging due to the impact of interference [43]. Traditional routing protocols such as AODV [83] and DSR [48] essentially follow the design methodology of wired networks by abstracting wireless links as wired links and look for path(s) with either the shortest delay or the least cost for a user between a pair of source and destination nodes. Recent advances on wireless routing took into account the impact of interference and MAC layer resource allocation, which led to a few interference-aware routing metrics and algorithms [82, 99, 101]. However, most of them are heuristic algorithms that cannot provide any performance guarantees, which are very important for real-time or mission-critical applications. Some optimization techniques (such as linear programming) were applied to design centralized routing algorithms in [3, 101], which may not be suitable for multihop wireless networks due to the lack of a fixed infrastructure or a central control node. Recently, optimization frameworks [13, 26, 62, 63] have been developed to solve routing and related resource allocation problems in multihop wireless networks in a distributed way. However, the state of the art cannot provide accurate convergence analysis when the subproblems in the iterative process are NP-hard. Therefore, it is desirable

to have a general optimization framework with fine-grained convergence analysis per iteration, which can deal with NP-hard subproblems and can lead to simple, fast, and fully distributed routing algorithms.

1.1.2 Contribution

The main contribution of this part is summarized in the following.

For the first thrust, we study the following two problems in Chapter 2.

1. *User profit optimization problem*: how to optimize the total profit of multiple simultaneous users in a full-duplex wireless network using multipath routing subject to node constraints.

2. *Network power consumption optimization problem*: how to optimize the network power consumption in a full-duplex wireless network using multipath routing subject to minimum user rate demands and node constraints.

Our contribution is two-fold.

1. We propose a collision-free full-duplex broadcast MAC and prove the necessary and sufficient conditions for successful collision-free broadcasts.

2. We formulate the two problems studied as convex programming systems and present two distributed iterative algorithms to solve them, respectively. Our algorithms compute the optimized user information flow (i.e. user behavior) for the network layer and the optimized node broadcast rate (i.e. node behavior) for the MAC layer. In each iteration, Lagrange multipliers are updated in a distributed manner according to the current user/node behaviors, and then each user and each node individually adjusts its own behavior based on the updated Lagrange multipliers. We provide bounds on

the amount of feasibility violation and the gap between our solution and the optimal solution at each iteration. We also use the dual space information to analyze the node load constraint violation at each iteration.

For the second thrust, we study the problem of allocating network resources to maximize the total user utility for a load-constrained wireless network in Chapter 3. Our contribution is two-fold.

1. The first contribution is a theoretical optimization framework, which may be applicable for many networking optimization problems. We propose a new subgradient optimization framework that has the following property. Given an approximation/optimal algorithm for solving the subproblem at each iteration, the framework leads to a result that can provide the following bounds at each iteration: (a) the bounds on the Lagrangian multipliers; (b) the bound on the amount of feasibility violation of the generated primal solutions; and (c) the upper and lower bounds on the gap between the optimal solution and the generated primal solutions. Note that standard subgradient methods require one to solve a sub-problem optimally at each iteration. However, in practice this sub-problem could be an NP-hard problem, which makes efficiently computing an optimal solution impossible. This problem was studied by Lin and Shroff [63] and Chen *et al.* [13]. However, the focus of these works is more on the asymptotic behavior of the primal sequences. Our framework further advances this research by providing fine-grained convergence, optimality, and dual space information *at each iteration*. We believe that this framework can provide a useful theoretical foundation for many networking optimization problems.

2. We formulate the resource allocation problem as a convex programming sys-

tem. Based on our α-approximation dual subgradient algorithm, we present a distributed iterative algorithm, which allows each user and each node to individually adjust its own behavior in each iteration period. This feature is of great importance for network scalability and self-organization. We prove the bounds on the amount of feasibility violation and the gap between our solution and the optimal solution *at each iteration*. We provide the bounds on node queue lengths, user utility deficits, and node load violation ratios *at each iteration* using dual space information.

1.2 Optimizing Anypath Routing for Multihop Wireless Networks

1.2.1 Motivation

Traditional routing algorithms and protocols for wireless networks often follow the design methodology for wired networks by abstracting the wireless links as wired links and looking for the shortest delay, least cost, or widest bandwidth path(s) between a pair of source and destination nodes [113]. However, for unreliable wireless networks, due to the broadcast nature of the wireless medium, it is usually less costly to transmit a packet to one of the nodes in a set of neighbors than to one specific neighbor. This observation motivated the emergence of a new technology, known as *opportunistic routing*, which takes advantage of the intermediate nodes overhearing the transmissions. It has been shown that opportunistic routing can help improve the performance of wireless networks [6, 9]. Dubois-Ferrière [25] generalized opportunistic routing and introduced the concept of *anypath routing*, which was subsequently studied in [24, 58, 59, 94, 116]. In anypath routing, each packet is broadcast to a forwarding set composed of several neighbors (called forwarders), and the packet is retransmitted only if none of the forwarders in this set receives it. As long as one of the forwarders receives this packet, it will be forwarded.

This part is composed of the following two main thrusts, targeted at understanding anypath routing for multihop wireless networks.

The first thrust is dedicated to understanding the computational complexity of multi-constrained anypath routing and designing approximate solutions. Previous works on anypath routing are focused on computing a shortest delay or least cost anypath, with only one QoS metric taken into account for route selection [24, 25, 58, 59]. However, many applications are associated with multiple QoS constraints. For instance, usually the energy consumption affects the network lifetime or the cost of the pair of source-destination nodes charged by the intermediate nodes providing forwarding service. Obviously, both delay and energy consumption should be taken into account when we are computing a route between a pair of source-destination nodes. There have been extensive studies on multi-constrained single path routing. Since the problem is NP-hard [105], many heuristics and approximation algorithms have been proposed [14, 42, 66, 105, 107, 109]. However, to the best of our knowledge, multi-constrained anypath routing has not been studied.

The second thrust is dedicated to cross-layer design for directional anypath routing. Prior works along the line of research on anypath routing were mainly focused on networks equipped with omnidirectional antennas. However, Yi *et al.* [110] proved that wireless networks can achieve a capacity gain when directional antennas are exploited. Authors of [18, 34, 76, 90, 98, 100] have proposed various schemes to improve performance of wireless networks by using directional communications. Thus it is of interest to design an optimally combined use of directional communications and anypath routing.

1.2.2 Contribution

The main contribution of this part is summarized in the following.

For the first thrust, we study the problem of anypath routing subject to multiple (K) constraints in Chapter 4. Our contribution is two-fold.

1. We show that this problem is NP-hard when the number of constraints is larger than one.
2. We present two polynomial time K-approximation algorithms. One is a centralized algorithm while the other one is a distributed algorithm.

For the second thrust, we study directional anypath routing and present a cross-layer design of MAC and routing layers in Chapter 5, which represents the first attempt towards the practical design for directional anypath routing.

Our contribution is two-fold:

1. For the MAC layer, we present a directional anycast MAC (DAM). DAM is an enhancement to the 802.11 MAC [41], and reduces to the 802.11 MAC when there is only one forwarder for each node and only omnidirectional antenna is being used.
2. For the routing layer, we propose two polynomial time routing algorithms to compute directional anypaths based on two antenna models, and prove their optimality based on the packet delivery ratio metric.

Chapter 2

PATHBOOK: Distributed Algorithms for Multipath Routing in Full-duplex Wireless Networks

2.1 Introduction

2.1.1 Motivation

A basic assumption in wireless communications is that a radio cannot transmit and receive on the same frequency at the same time. A recent research breakthrough challenges this assumption. Choi *et al.* [17] combined antenna, RF, and digital interference cancellation technologies, and designed the first practical single channel wireless full-duplex system. Jain *et al.* [44] then presented a full-duplex radio design using signal inversion and adaptive cancellation. This new design, unlike the prior work [17], supports wideband and high power systems. In theory, this new design has no limitation on bandwidth or power. Therefore, building full-duplex wireless networks (e.g. full-duplex 802.11n wireless networks) becomes possible.

These two fundamental works are mainly focused on the physical and MAC layer design. This encouraging breakthrough further calls for new research efforts in studying the impact of full-duplex radios on the network layer and designing higher layer protocols and algorithms for wireless networks to harvest the benefit brought by this technology. It motivates us to re-think of the network design from a theoretical perspective, relaxing this basic assumption that has been used in wireless communication research during the last decade. In this work, we concentrate on cross-layer optimization of the MAC and network layers for resource allocation in full-duplex wireless networks, comprehensively considering various resource and social constraints. We study resource allocation taking into

account three network entity roles: *users*, *nodes*, and *network* itself, which will be described respectively in the following.

From the perspective of users, each user obtains an amount of *utility* if a certain information rate is allocated to it. Since the intermediate nodes provide forwarding service for the user, they may charge the user a service fee (i.e. cost). Therefore, each user tries to maximize its *profit* (i.e. utility minus cost) by choosing an optimized *user behavior* (i.e. a route). Furthermore, when there exists more than one user in a wireless network, we need to solve the resource competition among users in order to maximize the total user profit, since individual user behaviors usually may not lead to a global optimum.

From the perspective of nodes, each node may also have its own *node constraints*, and hence may have its own *node behavior*. In this work, we consider the following two node constraints, which characterize a node's individual requirement and social requirement, respectively. The first node constraint is the *node max load constraint*, specifically, the maximum load this node is willing to carry. This constraint is important, since each node may have its own energy consumption concern, or computation and transmission capacity limits. The second node constraint is *the node load balance constraint*. Load balancing is an important issue in network design [30, 74, 84–86, 117]. Without considering the node load balance, the traditional routing design methodology, i.e., the shortest route is always chosen, could result in congestion on the center of a network or hotspots which drain the energy from the nodes in these areas much faster [57, 74]. Security may also be an issue if node load balance is not taken into account [74]. For instance, if a large number of messages go through a small number of nodes, then radio jamming can be a vicious attack. In contrast, it would be less effective and more expensive to jam a large number of nodes. Therefore, a node may set a maximum

allowed load difference compared with other nodes (such as H-hop neighbors). This constraint guarantees that a node will not carry too much more load than other nodes and hence balances the loads among different nodes. In summary, the node max load constraint reflects a node's own load requirement and the node load balance constraint shapes the load relationships among different nodes.

From the perspective of a network, network energy efficiency is of great importance, since wireless nodes are usually energy-constrained devices and in some scenarios it is even infeasible to recharge or replace wireless nodes. Hence, it is important to optimize the network energy efficiency.

We hence study the following two problems in this work.

1. *User profit optimization problem* (UPOP): how to optimize the total profit of multiple simultaneous users in a full-duplex wireless network using multipath routing subject to node constraints.

2. *Network power consumption optimization problem* (NPCOP): how to optimize the network power consumption in a full-duplex wireless network using multipath routing subject to minimum user rate demands and node constraints.

In this work, we realize multipath routing using an opportunistic routing (OR) protocol, called MORE, that was proposed by Chachulski *et al.* in [9]. OR has proved useful and effective for improving the performance of wireless networks by exploiting opportunistic forwarding capacity of intermediate nodes that overhear packet transmissions [6, 9]. Therefore, OR allows any node overhearing a packet to participate in forwarding it instead of deterministically choosing the next hop before transmitting a packet. Such OR-based multipath routing is called

opportunistic multipath routing (OMR).

2.1.2 Contribution

Our contributions are as follows: We first propose a collision-free full-duplex broadcast MAC and prove the necessary and sufficient conditions for successful collision-free broadcasts. We then formulate UPOP and NPCOP as convex programming systems and present two distributed iterative algorithms to solve UPOP and NPCOP, respectively. Our algorithms compute the optimized user information flow (i.e. user behavior) for the network layer and the optimized node broadcast rate (i.e. node behavior) for the MAC layer. In each iteration, Lagrange multipliers are updated in a distributed manner according to the current user/node behaviors, and then each user and each node individually adjusts its own behavior based on the updated Lagrange multipliers. We provide bounds on the amount of feasibility violation and the gap between our solution and the optimal solution at each iteration. We also use the dual space information to analyze the node load constraint violation at each iteration. To the best of our knowledge, this is the first work to study cross-layer optimization of the MAC and network layers for resource allocation in full-duplex wireless networks.

2.1.3 Related Work

2.1.3.1 Full-Duplex Transmission

The current solution to the full-duplex transmission is designed based on the interference cancellation technology. Digital cancellation has been extensively used in the literature, such as ZigZag [33] and successive interference cancellation [38]. Radunović *et al.* [88] suggested RF interference cancellation. Choi *et al.* [17] combined antenna, RF and digital interference cancellation technologies, and designed the first practical single channel wireless full-duplex system. This line of

research is further advanced by [23,92]. Recently, Jain *et al.* [44] presented a full-duplex radio design using signal inversion and adaptive cancellation. This new design, unlike [17], supports wideband and high power systems. In theory, this new design has no limitation on bandwidth or power. Therefore, it is possible to build full-duplex 802.11n devices. The difference between our work and the works above is that we concentrate on the joint optimization of higher layers (specifically the network layer and the MAC layer), rather than the physical layer, to exploit full-duplex transmissions.

2.1.3.2 Opportunistic Multipath Routing

Traditional routing chooses the nexthop before transmitting a packet. However, wireless links are usually unreliable and lossy. When link quality is poor, the probability that the chosen nexthop receives the packet is low. Opportunistic routing (OR), as a new technology for improving the performance of wireless networks, exploits spatial diversity and the broadcast nature of the wireless media. In OR, once a node that is closer to the destination overhears the transmission, it is allowed to participate in packet forwarding if needed. Biswas and Morris [6] proposed the ExOR protocol, and demonstrated that throughput can be significantly improved by using this more relaxed choice of nexthop. Chachulski *et al.* [9] further proposed MORE that integrates network coding into OR. Katti *et al.* [50] proposed a more aggressive system, MIXIT, which exploits a basic property of mesh networks: even when no node receives a packet correctly, any given bit is likely to be received correctly by some node. In order to solve the resource allocation problem in network coding based OR, Zhang and Li [115] introduced a game theory framework named Dice to find a Nash bargaining point of user rates. Koutsonikolas *et al.* [54] proposed CCACK, a new efficient network coding based OR protocol, which exploits a novel cumulative coded acknowledgment scheme.

Rozner *et al.* [91] presented a simple opportunistic adaptive routing protocol incorporating adaptive forwarding path selection, priority timer-based forwarding, local loss recovery, and adaptive rate control. The authors of [113,114] studied OR in multirate and multichannel wireless networks, respectively. In order to identify the essential properties of OR using mathematical language, Lu and Wu [69] designed a new OR algebra, based on the routing algebra. Dubois-Ferrière [25] introduced the concept of anypath routing, which has been extensively studied in [24, 27, 46, 58]. Radunović *et al.* [87] presented an optimization framework for computing optimal flow control, routing, scheduling, and rate adaptation schemes, and designed a distributed heuristic algorithm. Our work differs from the above works in that we are the first to integrate full-duplex transmission into OR, and the first to present a distributed resource allocation algorithm for full-duplex wireless networks based on OR.

2.1.3.3 Network Resource Optimization

Optimizing network utility is an important objective in networking applications [13, 26, 28, 47, 63, 64, 67, 68, 80, 81, 106] (see [16, 95] for more discussions on this subject). Load balancing is also an important issue in network design. Gao and Zhang [31] studied wireless network routing with the aim of achieving good performance in terms of both stretch factor and load-balancing ratio. The proposed algorithms can achieve constant competitive factors for both measures when all the nodes are located in a narrow strip. Pham and Perreau [84] showed that multi-path routing provides better congestion and traffic balancing. Further work by Ganjali and Keshavarzian [30] showed that multipath routing can balance load only if a very large number of paths are used. Popa *et al.* [85] showed that an optimum routing scheme based on the shortest paths can be computed by using linear programming. Zorzi and Rao [117] solved the energy-efficiency issues by

balancing the load reactively. Energy efficiency optimization is also an important research topic. Abdulla *et al.* [1] proposed a solution to extend the network lifetime of wireless sensor networks through a hybrid approach that combines two routing strategies, flat multi-hop routing and hierarchical multi-hop routing. Li *et al.* [61] addressed the problem of energy efficient reliable routing for wireless ad hoc networks in the presence of unreliable communication links, devices, or lossy wireless link layers by integrating the power control techniques into the energy efficient routing. Ma *et al.* [72] proposed an interference aware metric, called network allocation vector count, and showed that network lifetime can be notably prolonged when this metric is employed to conduct transmit power control. Mao *et al.* [73] studied energy efficient opportunistic routing in wireless sensor networks and investigated how to select and prioritize forwarder list to minimize energy consumptions. Singh *et al.* [97] studied a case for using new power-aware metrics for determining routes in wireless ad hoc networks. Chang and Tassiulas used linear programming to capture the issue of power consumption in [10], and proposed a centralized algorithm to determine the maximum lifetime in [11]. Sankar and Liu [93] studied the routing problem with the goal of maximizing the network lifetime. Rao *et al.* [89] studied the tradeoff between energy consumption and network performance in real-time wireless sensor networks by investigating the interaction between the network performance optimization and network lifetime maximization problems. Liu *et al.* [65] proposed an energy-aware routing protocol for sensor networks, which improves lifetime by minimizing energy consumption for in-network communications and balancing the load among all the nodes. Unlike the classic network optimization problems discussed above, we comprehensively take into account various resource and social requirements of users, nodes, and network itself, which makes this work novel even if we do not consider full-duplex transmission. In addition, we provide optimality and dual space infor-

mation analysis for each iteration rather than only providing asymptotic analysis.

2.2 Model

2.2.1 Preliminaries

In this section, we review MORE [9]. The basic operation of our OMR protocol is based on MORE.

Source node The traffic coming from this source node is divided into a number of batches each with M packets (called *native packets*) in it. The source continuously generates coded packets from each batch using random linear network coding. Specifically, each coded packet is $m'_j = \sum_i c_{ji} m_i$, where c_{ji} is a random coefficient, and m_i is a native packet from the current batch. $\{c_{j1}, ..., c_{ji}, ..., c_{jM}\}$ is called the *code vector* of packet m'_j. The source node keeps generating and sending coded packets from a batch until it receives the ACK for this batch from the destination node.

Intermediate node The sender includes in the *forwarder list* the nodes which are closer (in ETX metric [22]) to the destination than itself, ordered according to their proximity to the destination. Whenever an intermediate node v hears a packet, this node checks whether it is a forwarder by looking for its ID in the forwarder list. If node v is a forwarder for this packet, node v then checks whether the packet contains new information (i.e. whether it is *an innovative packet*) using simple algebra, such as Gaussian Elimination. A packet is innovative if it is linearly independent from the packets the node has received from this batch. If this packet is an innovative packet, node v generates a new packet – a random linear combination of the coded packets it has received from the same batch, and broadcasts this new packet. Suppose, for example, that intermediate node v has

coded packets of the form $m'_j = \sum_i c_{ji} m_i$. It generates more coded packets by computing a linear combination of these coded packets as follows: $m'' = \sum_j a_j m'_j$, where a_j's are random numbers. Obviously, m'' is also a linear combination of the native packets (i.e. $m'' = \sum_i (\sum_j a_j c_{ji}) m_i$). Since wireless channels are unreliable, MORE uses a heuristic algorithm to compute resource allocation. This heuristic algorithm computes the expected number of transmissions an intermediate node must make once it receives an innovative packet. Compared with this heuristic algorithm, in order to optimize the resource allocation, our algorithms compute the optimized user information flow (i.e. user behavior) for the network layer and the optimized node broadcast rate (i.e. node behavior) for the MAC layer.

Destination node The destination node keeps all the received innovative packets until M innovative packets from the current batch are received. It then decodes the original whole batch using matrix inversion and sends an ACK back to the source node.

2.2.2 System Model

We consider a full-duplex wireless network where there are K simultaneous users. Each user maintains a session from a source node to a destination node, and uses zero or more intermediate nodes to forward the packets. Our work is particularly suitable for applications with long user sessions (e.g. large file transfer) in static or slowly changing wireless networks (e.g. [2]).

2.2.2.1 Opportunistic Multipath Routing Sub-model

In this subsection, we model OMR in full-duplex wireless networks. We assume that all the nodes are equipped with single channel full-duplex transceivers. There-

fore, a node in our model is capable of transmitting and receiving on the same frequency simultaneously.

A wireless network is modeled as a directed graph $G = (V, E)$, where E is the set of edges and V is the set of vertices. We use the following terms interchangeably: edge and link, vertex and node. Unlike the traditional unit-disk model, the *transmission range* is defined as the distance where the reception probability falls below a small given threshold, as in [115]. There exists a directed edge (u, v) in G if v is in u's transmission range. Since the transmission range is defined as the distance where the reception probability falls below a small threshold, as in [115] we assume that the *interference range* equals to the transmission range. As in [9], each user performs a distributed node pre-selection procedure to add intermediate nodes into its forwarder list so that each forwarder is closer (in ETX metric [22]) to the destination than its predecessors. Let $G(V, E)$ denote the resulting topology involved in the session of user k, where V and E are the set of nodes and the set of directed links, respectively.

Collision-Free Full-Duplex Broadcast MAC We consider a *collision-free full-duplex broadcast MAC* based on slotted scheduling: a broadcast transmission from node u is collision-free if and only if all the other transmitters that are transmitting packets are outside the range of any intended downstream receiver of node u. Consider the example shown in Fig.2.1, where v_3 and v_4 are the intended downstream receivers of node v_1. If no collision occurs at v_3, the transmitters of nodes v_4 and v_5 should not transmit. If no collision occurs at v_4, the transmitters of nodes v_2, v_3, v_5 and v_6 should not transmit. Thus a collision-free broadcast transmission from node v_1 means that no collision occurs at both v_3 and v_4. Since full-duplexing is supported, each node is able to receive packets while transmitting.

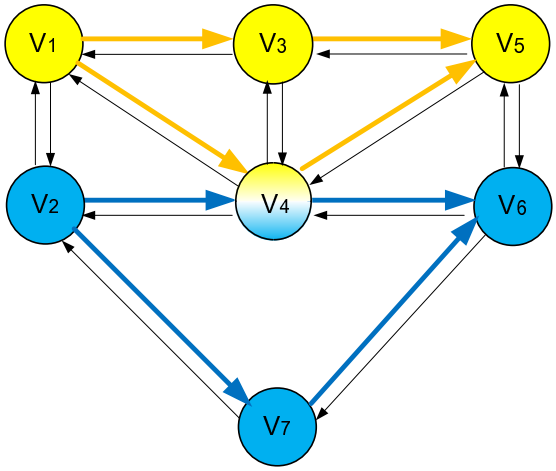


Figure 2.1: An example network: if a node is located in another node's transmission (or interference) range, there is a directed edge between them. There are two users in this network. User 1 transmits data from v_1 to v_5 through intermediate nodes v_3 and v_4, and user 2 transmits data from v_2 to v_6 through v_4 and v_7.

For example, v_7 can transmit the previously received packets to v_6 while receiving packets from v_2. Note that although v_3 (or v_4) is also capable of transmitting and receiving simultaneously, when it is receiving packets from v_1, it should not transmit because that transmission will interfere with v_4 (or v_3).

Let $R(u)$ denote the set of nodes whose transmission (interference) can be heard by node u. For example, in the network shown in Fig.2.1, $R(v_1)$=$\{v_2, v_3, v_4\}$, $R(v_2)$=$\{v_1, v_4, v_7\}$, and $R(v_4)$=$\{v_1, v_2, v_3, v_5, v_6\}$. We use $B_k^{(t)}(u)$ to denote a binary variable indicating whether node u transmits user k's data in slot t. We assume that a node can only transmit one user's data in one time slot. *A necessary and sufficient condition for collision-free broadcasts* is that *for any* $k \in [1, K]$, *the following inequality holds*:

$$\sum_{m\in[1,K]} \sum_{v\in R(u)} B_m^{(t)}(v) \leq 1, \forall u \in V \setminus s_k. \tag{2.1}$$

This inequality means that any node u allows the broadcast transmission from at most one transmitter within its range (excluding its transmit antenna). For

user k, its source s_k is excluded, since s_k does not need to receive data from other nodes. Before proving this necessary and sufficient condition, let us use the example shown in Fig.2.1 for an illustration. Assume that user 1 transmits packets from v_1 to v_5 through v_3 and v_4, and user 2 transmits packets from v_2 to v_6 through v_4 and v_7. Thus $V = \{v_1, v_3, v_4, v_5\}$ and $V = \{v_2, v_4, v_6, v_7\}$. The corresponding necessary and sufficient conditions can be expressed as:
for $k = 1$,

$$\text{for } v_3, \quad B_1^{(t)}(v_1) + B_1^{(t)}(v_4) + B_2^{(t)}(v_4) \leq 1; \tag{2.2}$$

$$\text{for } v_4, \quad B_1^{(t)}(v_1) + B_1^{(t)}(v_3) + B_2^{(t)}(v_2) \leq 1; \tag{2.3}$$

$$\text{for } v_5, \quad B_1^{(t)}(v_3) + B_1^{(t)}(v_4) + B_2^{(t)}(v_4) \leq 1; \tag{2.4}$$

and for $k = 2$,

$$\text{for } v_4, \quad B_1^{(t)}(v_1) + B_1^{(t)}(v_3) + B_2^{(t)}(v_2) \leq 1; \tag{2.5}$$

$$\text{for } v_6, \quad B_1^{(t)}(v_4) + B_2^{(t)}(v_4) + B_2^{(t)}(v_7) \leq 1; \tag{2.6}$$

$$\text{for } v_7, \quad B_2^{(t)}(v_2) \leq 1, \tag{2.7}$$

where $B_k^{(t)}(u) \in \{0, 1\}$ for $k = 1$ or 2 and $u \in V$. Note that $B_1^{(t)}(v_2) = B_1^{(t)}(v_5) = B_1^{(t)}(v_6) = B_1^{(t)}(v_7) = B_2^{(t)}(v_1) = B_2^{(t)}(v_3) = B_2^{(t)}(v_5) = B_2^{(t)}(v_6) = 0$. Inequality (2.2) means that if v_3 can receive packets of user 1 from v_1 without collision, then v_4 cannot transmit. Note that since full-duplexing is used, v_3 does not have to care whether its radio is transmitting. Although Inequalities (2.3) and (2.5) are the same, they convey different meanings. Inequality (2.3) means that if v_4 can receive packets of user 1 from v_1 without collision, then v_2 and v_3 cannot transmit. Inequality (2.5) means that if v_4 can receive packets of user 2 from v_2 without collision, then v_1 and v_3 cannot transmit. Likewise, since full-duplexing is used, v_4 does not have to care whether its radio is transmitting. Note that although Inequalities (2.3) and (2.5) convey different meanings, in real computation, such

redundant inequalities can be removed to reduce computational complexity. In addition, we do not have the inequalities for v_1 and v_2 since they are sources and do not need to receive data from other nodes.

We now prove this necessary and sufficient condition. On one hand, if for any $k \in [1, K]$, Inequality (2.1) holds, then for any node u, no more than one of its intended upstream transmitters is broadcasting. Due to the full-duplex operation, no matter whether node u is transmitting, node u can hear this broadcasting without collision. On the other hand, if condition (2.1) does not hold for some $k \in [1, K]$ and some node $u \in V \setminus s_k$, then $\sum_{m\in[1,K]} \sum_{v\in R(u)} B_m^{(t)}(v) \geq 2$. Recall that a node can only transmit one user's data in one time slot. This means that at least two of u's upstream transmitters are broadcasting, which results in a collision at node u.

Assuming that the period of a schedule is T, for any $k \in [1, K]$ we thus have

$$\frac{C}{T} \sum_{t\in[1,T]} \sum_{m\in[1,K]} \sum_{v\in R(u)} B_m^{(t)}(v) \leq C, \forall u \in V \setminus s_k,$$

where C is the MAC layer capacity, which is the maximum broadcast rate of a node when no interference presents. Since the *average broadcast rate* of node u for user m (i.e. *node u's behavior for user m*) can be computed as $b_m(u) = \lim_{T\to\infty} \frac{C}{T} \sum_{t\in[1,T]} B_m^{(t)}(u)$, for any $k \in [1, K]$ we have

$$\sum_{m\in[1,K]} \sum_{v\in R(u)} b_m(v) \leq C, \forall u \in V \setminus s_k. \tag{2.8}$$

Since we transformed an integer variable $B_m^{(t)}(u)$ into continuous $b_m(u)$ by averaging, constraint (2.8) is necessary, but not sufficient.

Remark: We note that full-duplex transmission could lead to other routing designs such as wormhole routing [17]. As the first attempt towards the routing

optimization for full-duplex wireless networks, this work mainly focuses on routing optimization based on this collision-free broadcast MAC.

Information Flow Considering the flow conservation principle, for any $u \in V$ we have

$$\sum_{(u,v)\in E} r_k(u,v) - \sum_{(w,u)\in E} r_k(w,u) = h_k(u), \forall k \in [1,K], \tag{2.9}$$

where $h_k(u)$ is equal to λ_k if $u = s_k$, $-\lambda_k$ if $u = d_k$, and 0 otherwise. s_k and d_k denote user k's source and destination, respectively. λ_k and $r_k(u,v)$ denote user k's *total information rate* and *link information rate* on link (u,v), respectively. The link information rate vector $\overrightarrow{r_k}$ represents *user k's behavior* (i.e. *user k's route*).

Network Coding Constraint Our OMR is realized using network coding based OR (see the operations of MORE in Section 2.2.1). Although Lun *et al.* [71] made an exact characterization of the coding model, due to an exponential number of constraints, it makes the problem intractable. To balance the model accuracy and the computation tractability, we adopt the following model proposed in [115]:

$$b_k(u)p(u,v) \geq r_k(u,v), \forall (u,v) \in E, \tag{2.10}$$

where $p(u,v)>0$ is the packet delivery ratio of link (u,v). This is not a tight bound. However, as discussed in [115], this is an effective approximation to the behavior of an actual wireless network using network coding. It includes the useful information that users and nodes can exploit to induce a better performance.

2.2.2.2 User Profit Sub-model

User k obtains a utility of $\mathcal{U}_k(\lambda_k)$ if it achieves a total information rate of λ_k. We assume that $\mathcal{U}_k(\cdot)$ is an increasing, concave and continuously differentiable

function, as in [51]. Since intermediate nodes provide forwarding services for a user, each forwarder charges this user a service fee which is proportional to the service rate (i.e. the broadcast rate) for this user. Let $\sigma_k(u)$ denote the unit service price charged by node u for user k. User k hence pays $\sigma_k(u)b_k(u)$ ($\sigma_k(u) \geq 0$) for node u's forwarding service. User k's profit is thus defined by

$$\mathcal{P}_k(\lambda_k, \overrightarrow{b}_k) = \mathcal{U}_k(\lambda_k) \quad - \sum_{u \in V \setminus d_k} \sigma_k(u) b_k(u). \tag{2.11}$$

2.2.2.3 Node Constraint Sub-model

Node Max Load Constraint Due to the energy consumption concern, or computation and transmission capacity limits, each node u needs to consider the maximum average broadcast rate (denoted by $\mathcal{M}^{max}(u)$) that it is willing to provide. As a result, node u sets an upper bound on its average broadcast rate:

$$b(u) = \sum_{k \in [1,K]} b_k(u) \leq \mathcal{M}^{max}(u), \mathcal{M}^{max}(u) \in (0, \Phi]. \tag{2.12}$$

Node Load Balance Constraint We define a *load balance area* (i.e. a set of nodes) for each node. We use $v \in A(u)$ to denote that v is in u's load balance area. That is to say,

$$b(u) - b(v) \leq \mathcal{B}^{max}(u, v), \mathcal{B}^{max}(u, v) \in (0, \Theta], \forall v \in A(u), \tag{2.13}$$

where $\mathcal{B}^{max}(u, v)$ is a parameter set by node u. $A(u)$ and $\mathcal{B}^{max}(u, v)$ are used by node u to achieve a controllable balanced load with other nodes. In this work, we consider a symmetric load balance (i.e. $v \in A(u) \iff u \in A(v)$, and $\mathcal{B}^{max}(u, v) = \mathcal{B}^{max}(v, u)$). Although in practice the load balance constraint could be asymmetric, there would be no big difference for our mathematical analysis and algorithm design. We hence concentrate on this simpler definition, which makes our expressions clearer.

2.2.2.4 Network Power Consumption Sub-model

Based on our slotted MAC, we assume that a node's power consumption is proportional to its broadcast rate, since in most cases transmission operations dominate the energy consumption. Specifically, node u's average power consumption is computed as $l(u)b(u)$, where $l(u)$ is the *power consumption ratio*. The network power consumption is computed as:

$$\mathcal{P}(\overrightarrow{b}) = \sum_{k\in[1,K]} \sum_{u\in V\setminus d_k} l(u)b_k(u). \tag{2.14}$$

2.3 Problem Formulation

2.3.1 User Profit Optimization Problem (UPOP)

By our user profit submodel (2.11), OMR submodel (2.8)-(2.10), and node constraint submodel (2.12), (2.13), the user profit optimization problem (UPOP) is formulated as:

System $1(\overrightarrow{r}, \overrightarrow{b})$:

$$\max \quad \sum_{k\in[1,K]} \Big(\mathcal{U}_k(\lambda_k) - \sum_{u\in V\setminus d_k} \sigma_k(u) b_k(u)\Big),$$

$$\text{s.t.} \sum_{(u,v)\in E} r_k(u,v) - \sum_{(w,u)\in E} r_k(w,u) = h_k(u), \forall u \in V, k \in [1,K]; \tag{2.15}$$

$$\sum_{m\in[1,K]} \sum_{v\in R(u)} b_m(v) \le C, \forall u \in V \setminus s_k, k \in [1,K]; \tag{2.16}$$

$$b_k(u)p(u,v) \ge r_k(u,v), \forall (u,v) \in E, k \in [1,K]; \tag{2.17}$$

$$\sum_{k\in[1,K]} b_k(u) \le \mathcal{M}^{max}(u), \forall u \in V; \tag{2.18}$$

$$\sum_{k\in[1,K]} b_k(u_1) - \sum_{k\in[1,K]} b_k(u_2) \le \mathcal{B}^{max}(u_1,u_2), \forall u_2 \in A(u_1), u_1 \in V; \tag{2.19}$$

$$\text{over} \quad r_k(u,v) \in [0,C], \forall (u,v) \in E, k \in [1,K],$$

$$b_k(u) \in [0,C], \forall u \in V \setminus d_k, k \in [1,K].$$

2.3.2 Network Power Consumption Optimization Problem (NPCOP)

Each user k has a *minimum user rate demand* Λ_k. By our network power consumption submodel (2.14), OMR submodel (2.8)-(2.10), node constraint submodel (2.12), (2.13), and minimum user rate demand, the network power consumption optimization problem (NPCOP) is formulated as System 2. We will discuss how to select Λ_k in Section 2.4.4, since if some of them are too large, the network resource may not be able to satisfy them and as a result System 2 may not have a feasible solution.

$System2(\overrightarrow{r},\ \overrightarrow{b}\ )$:

$$\begin{aligned} \min \quad & \sum_{k\in[1,K]}\sum_{u\in V\setminus d_k} l(u)b_k(u), \\ \text{s.t.} \quad & (2.15)-(2.19); \\ & \lambda_k \geq \Lambda_k, k\in[1,K]; \qquad (2.20) \\ \text{over} \quad & r_k(u,v)\in[0,C], \forall (u,v)\in E, k\in[1,K], \\ & b_k(u)\in[0,C], \forall u\in V\setminus d_k, k\in[1,K]. \end{aligned}$$

2.4 Distributed Optimization Algorithms

Although Systems 1 and 2 can be solved using traditional centralized convex programming techniques [7], distributed algorithms are preferable for the purpose of practical implementations. We combine Lagrangian decomposition with the approximate dual subgradient method (ADSM) proposed by Nedić and Ozdaglar in [78], and propose distributed algorithms for solving Systems 1 and 2 (denoted by *Pathbook-I* and *Pathbook-II*, respectively). ADSM can be summarized as follows.

The *primal problem* is the following:

$$\min\ f(\vec{x}), \qquad \text{s.t. } g(\vec{x})\preceq 0, \qquad \text{over } \vec{x}\in\vec{X}, \tag{2.21}$$

where $f(\cdot)$:$\mathbb{R}^N\mapsto\mathbb{R}$ is a convex function ($\mathbb{R}^N$ is an N-dimensional vector space), $g(\cdot)$=$(g_1(\cdot),...,g_\rho(\cdot))^T$ and each $g_j(\cdot):\mathbb{R}^N\mapsto\mathbb{R}$ is a convex function, and $\vec{X}\in\mathbb{R}^N$ is a nonempty compact convex set. $(\cdot)^T$ denotes the transpose of $(\cdot)$.

The *dual problem* is the following:

$$\max\ q(\vec{\delta}) = \inf_{\vec{x}\in\vec{X}}\big(L(\vec{x},\vec{\delta})\big), \quad \text{s.t. } \vec{\delta}\succeq 0, \ \text{ over } \vec{\delta}\in\mathbb{R}^\rho,$$

where $L(\vec{x},\vec{\delta}) = f(\vec{x})+\vec{\delta}^T g(\vec{x})$ is the Lagrangian of the primal problem, and $\vec{\delta}$ is the vector of Lagrange multipliers.

ADSM iteratively computes $\vec{\delta}^{(1)}$, $\vec{x}^{(1)}$, $\vec{\delta}^{(2)}$, $\vec{x}^{(2)}$, $\cdots$ from a starting point $\{\vec{\delta}^{(0)},\vec{x}^{(0)}\} \in \mathbb{R}^{\rho} \times \vec{X}$ as follows:

for $i \geq 0$,

$$\vec{\delta}^{(i+1)} = [\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)})]^{+}, \text{where } \vec{x}^{(i)} \in \text{argmin}_{\vec{x} \in \vec{X}} \left(f(\vec{x}) + (\vec{\delta}^{(i)})^{T} g(\vec{x}) \right), \quad (2.22)$$

where η is a constant stepsize. For a vector $\vec{x} \in \mathbb{R}^{N}$, $\vec{x}^{+}$ is the projection of $\vec{x}$ onto the nonnegative orthant in $\mathbb{R}^{N}$, i.e. $\vec{x}^{+}=(\max\{0, x_1\}, ..., \max\{0, x_N\})^{T}$ for $\vec{x}=(x_1, ..., x_N)^{T}$. The primal solution is approximated using averaging:

$$\hat{x}^{(i)} = \frac{1}{i} \sum_{m=0}^{i-1} \vec{x}^{(m)}, i \geq 1. \quad (2.23)$$

Our distributed algorithms use subgradient methods in the dual space and exploit the subgradient information to produce near-feasible and near-optimal primal solutions. The algorithmic framework of Pathbook-I and Pathbook-II is designed based on ADSM. We now briefly outline this framework. Let $\vec{\delta}^{(i)}$ denote the vector of Lagrange multipliers in the i^{th} iteration, and $\vec{x}^{(i)}$ denote the vector $((\overrightarrow{r}^{(i)})^{T}, (\overrightarrow{b}^{(i)})^{T})^{T}$. According to ADSM, from a starting point we compute $\vec{\delta}^{(1)}$, and then compute $\vec{x}^{(1)}$ based on $\vec{\delta}^{(1)}$. In the second iteration, we compute $\vec{\delta}^{(2)}$ based on $\vec{\delta}^{(1)}$ and $\vec{x}^{(1)}$, and then compute $\vec{x}^{(2)}$ based on $\vec{\delta}^{(2)}$. This procedure continues until a stopping criterion is reached. Using Lagrangian decomposition, the update operations for both $\vec{\delta}^{(i)}$ and $\vec{x}^{(i)}$ can be done in a distributed manner.

2.4.1 Distributed Optimization Algorithm Pathbook-I for UPOP

After some simple mathematical manipulations and removing redundant inequalities, the Lagrangian of System 1 subject to constraint (2.15) can be expressed

as:

$$
\begin{aligned}
&L(\overrightarrow{r}, \overrightarrow{b}, \overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{\mu}, \overrightarrow{\omega}) \\
&= -\sum_{k\in[1,K]} \mathcal{U}_k(\lambda_k) + \sum_{k\in[1,K]}\sum_{u\in V} \sigma_k(u) b_k(u) \\
&\quad + \sum_{u\in V} \alpha(u)\Big(\sum_{k\in[1,K]}\sum_{v\in R(u), u\neq s_k} b_k(v) - C\Big) \\
&\quad + \sum_{k\in[1,K]}\sum_{(u,v)\in E} \beta_k(u,v)\big(r_k(u,v) - b_k(u)p(u,v)\big) \\
&\quad + \sum_{u\in V} \mu(u)\Big(\sum_{k\in[1,K]} b_k(u) - \mathcal{M}^{max}(u)\Big) \\
&\quad + \sum_{u_1,u_2\in V: u_2\in A(u_1)} \omega(u_1,u_2)\Big(\sum_{k\in[1,K]}\big(b_k(u_1) - b_k(u_2)\big) - \mathcal{B}^{max}(u_1,u_2)\Big). \qquad (2.24)
\end{aligned}
$$

In the i^{th} iteration, according to formula (2.22) we update Lagrange multipliers as follows:

$$
\begin{aligned}
&\forall u \in V, \alpha^{(i)}(u) = [\alpha^{(i-1)}(u) + \eta \mathcal{M}_u^{(i-1)}]^+, \\
&\forall (u,v) \in E, \beta_k^{(i)}(u,v) = [\beta_k^{(i-1)}(u,v) + \eta \mathcal{C}_{(k,u,v)}^{(i-1)}]^+, \\
&\forall u \in V, \mu^{(i)}(u) = [\mu^{(i-1)}(u) + \eta \mathcal{G}_u^{(i-1)}]^+, \\
&\forall u \in V \text{and } v \in A(u), \omega^{(i)}(u,v) = [\omega^{(i-1)}(u,v) + \eta \mathcal{H}_{(u,v)}^{(i-1)}]^+,
\end{aligned}
$$

where

$$
\begin{aligned}
\mathcal{M}_u^{(i-1)} &= \sum_{k\in[1,K]}\sum_{v\in R(u), u\neq s_k} b_k^{(i-1)}(v) - C, \\
\mathcal{C}_{(k,u,v)}^{(i-1)} &= r_k^{(i-1)}(u,v) - b_k^{(i-1)}(u)p(u,v),
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{G}_u^{(i-1)} &= \sum_{k\in[1,K]} b_k^{(i-1)}(u) - \mathcal{M}^{max}(u), \\
\mathcal{H}_{(u,v)}^{(i-1)} &= \sum_{k\in[1,K]} \big(b_k^{(i-1)}(u) - b_k^{(i-1)}(v)\big) - \mathcal{B}^{max}(u,v).
\end{aligned}
$$

After the Lagrange multipliers are computed, according to formula (2.22), we need to find $\overrightarrow{r}^{(i)}$ and $\overrightarrow{b}^{(i)}$ so as to minimize $L(\overrightarrow{r}, \overrightarrow{b}, \overrightarrow{\alpha}^{(i)}, \overrightarrow{\beta}^{(i)}, \overrightarrow{\mu}^{(i)}, \overrightarrow{\omega}^{(i)})$.

After some mathematical manipulations, Equation (2.24) leads to the following:

$$
\begin{aligned}
&L(\overrightarrow{r}, \overrightarrow{b}, \overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{\mu}, \overrightarrow{\omega}) \\
=&-\left(\sum_{k\in[1,K]} \mathcal{U}_k(\lambda_k) + \sum_{k\in[1,K]}\sum_{(u,v)\in E} \beta_k(u,v) r_k(u,v)\right) \\
&+\Bigg(\sum_{k\in[1,K]}\sum_{u\in V} \sigma_k(u) b_k(u) + \sum_{k\in[1,K]}\sum_{u\in V}\sum_{v\in R(u), v\neq s_k} \alpha(v) b_k(u) \\
&-\sum_{k\in[1,K]}\sum_{(u,v)\in E} \beta_k(u,v) b_k(u) p(u,v) + \sum_{u\in V}\mu(u)\sum_{k\in[1,K]} b_k(u) \\
&+\sum_{u_1,u_2\in V: u_2\in A(u_1)} \omega(u_1,u_2)\sum_{k\in[1,K]} \big(b_k(u_1) - b_k(u_2)\big)\Bigg) - C\sum_{u\in V}\alpha(u) \\
&-\mathcal{B}^{max}(u_1,u_2)\sum_{u_1,u_2\in V: u_2\in A(u_1)} \omega(u_1,u_2) - \sum_{u\in V}\mu(u)\mathcal{M}^{max}(u). \qquad (2.25)
\end{aligned}
$$

Fortunately, by Equation (2.25) this minimization operation can be decomposed to users' behaviors and nodes' behaviors. Specifically, in the i^{th} iteration, user k solves the following problem:

$$
\begin{aligned}
\min \quad & -\mathcal{U}_k(\lambda_k^{(i)}) + \sum_{(u,v)\in E} \beta_k^{(i)}(u,v) r_k^{(i)}(u,v), \qquad (2.26)\\
\text{s.t.} \quad & \sum_{(u,v)\in E} r_k^{(i)}(u,v) - \sum_{(w,u)\in E} r_k^{(i)}(w,u) = h_k^{(i)}(u), u\in V, \\
\text{over} \quad & r_k^{(i)}(u,v)\in[0,C], \forall (u,v)\in E,
\end{aligned}
$$

and node u solves the following problem:

$$
\begin{aligned}
\min \quad & \Big(\sigma_k(u) + \sum_{v\in R(u), v\neq s_k} \alpha^{(i)}(v) - \sum_{(u,v)\in E} \beta_k^{(i)}(u,v) p(u,v) \\
& +\mu^{(i)}(u) + \sum_{v\in A(u)} \omega^{(i)}(u,v) - \sum_{v\in A(u)} \omega^{(i)}(v,u)\Big) b_k^{(i)}(u), \qquad (2.27)\\
\text{over} \quad & b_k^{(i)}(u)\in[0,C], u\in V\setminus d_k.
\end{aligned}
$$

Obviously, Problem (2.27) can be solved by each node u. Now we describe how to solve Problem (2.26). Based on the flow-path formulation technique in

[115], we consider an equivalent problem:

$$\begin{aligned} \min \quad & -\mathcal{U}_k(\sum_{\pi\in P}\gamma_k(\pi)) + \sum_{\pi\in P}\kappa_k(\pi)\gamma_k(\pi), \\ \text{over} \quad & \sum_{\pi\in P}\gamma_k(\pi) \in [0, C], \end{aligned}$$

where P is a set of single paths, $\gamma_k(\pi)$ is the flow rate of user k following path π, and $\kappa_k(\pi)$ is equal to $\sum_{(u,v)\in\pi}\beta_k^{(i)}(u,v)$. Solving this equivalent problem will ensure that the min-cost single path (with respect to $\beta_k^{(i)}(u,v)$) is always chosen. We use Γ_k to denote the value of this single path flow, and hence transform the problem above to

$$\min \ -\mathcal{U}_k(\Gamma_k) + \kappa_k^{min}\Gamma_k, \text{ over } \Gamma_k \in [0, C], \tag{2.28}$$

where κ_k^{min} is the cost of the min-cost path (with respect to $\beta_k^{(i)}(u,v)$). Therefore, each user k can use a distributed shortest path algorithm to compute κ_k^{min}, and then solve Problem (2.28). $r_k^{(i)}(u,v)$ is set to the solution to Problem (2.28) if (u,v) is on this min-cost path, and 0 otherwise.

In the i^{th} iteration, according to Formula (2.23) users and nodes can approximate the primal solutions in the following way: for $i \geq 1$

$$\hat{r}_k^{(i)}(u,v) = \frac{1}{i}\sum_{m=0}^{i-1} r_k^{(m)}(u,v), \hat{b}_k^{(i)}(u) = \frac{1}{i}\sum_{m=0}^{i-1} b_k^{(m)}(u). \tag{2.29}$$

After the i^{th} iteration, user k uses $\hat{r}_k^{(i)}(u,v)$ as its user behavior on link (u,v) and node u uses $\hat{b}_k^{(i)}(u)$ as its node behavior for user k until the next iteration. Equation (2.29) tells each user its optimized information flow. However, transmissions in our OMR are opportunistic. To realize this information flow, we adopt a *transmission credit mechanism* (see Section 2.4.4).

We now analyze what information is needed in each iteration. We first analyze the update operation for Lagrange multipliers. For $\alpha^{(i)}(u)$, node u needs

its one-hop neighbors' broadcast rates. For $\beta_k^{(i)}(u,v)$, node u needs $b_k^{(i)}(u)$ and $r_k^{(i)}(u,v)$. For $\mu^{(i)}(u)$, node u needs its own broadcast rate $b_k^{(i)}(u)$. For $\omega^{(i)}(u,v)$, node u needs the broadcast rates of the nodes located in its load balance area. We then analyze the update operation of user and node behaviors when the Lagrange multipliers have been computed. Each user can update its user behavior by using a distributed shortest path algorithm. For node behaviors, each node needs the information from its one-hop neighbors and the nodes in its load balance area. This analysis indicates that no global information is needed as long as no node includes all the nodes in its load balance area. In practice, a node can consider the set of its 1 or 2-hop neighbors as its load balance area. Note that a large load balance area will lead to many information exchanges. We will discuss this issue in Section 2.4.4.

2.4.2 Distributed Optimization Algorithm Pathbook-II for NPCOP

As in Pathbook-I, we first compute the Lagrangian of System 2 subject to constraint (2.15) as follows:

$$
\begin{aligned}
& L(\overrightarrow{r}, \overrightarrow{b}, \overrightarrow{\alpha}, \overrightarrow{\beta}, \overrightarrow{\mu}, \overrightarrow{\omega}, \overrightarrow{y}) \\
= & \sum_{k\in[1,K]} \sum_{u\in V\setminus d_k} l(u) b_k(u) \\
+ & \sum_{u\in V} \alpha(u) \Big(\sum_{k\in[1,K]} \sum_{v\in R(u), u\neq s_k} b_k(v) - C \Big) \\
+ & \sum_{k\in[1,K]} \sum_{(u,v)\in E} \beta_k(u,v) \big(r_k(u,v) - b_k(u) p(u,v) \big) \\
+ & \sum_{u\in V} \mu(u) \Big(\sum_{k\in[1,K]} b_k(u) - \mathcal{M}^{max}(u) \Big) \\
+ & \sum_{u_1,u_2\in V: u_2\in A(u_1)} \omega(u_1,u_2) \Big(\sum_{k\in[1,K]} \big(b_k(u_1) - b_k(u_2) \big) - \mathcal{B}^{max}(u_1,u_2) \Big) \\
+ & \sum_{k\in[1,K]} y_k(\Lambda_k - \lambda_k). \qquad (2.30)
\end{aligned}
$$

In the i^{th} iteration, according to Formula (2.22) we first need to update the Lagrange multipliers. For $\alpha^{(i)}(u)$, $\beta_k^{(i)}(u,v)$, $\mu^{(i)}(u)$, and $\omega^{(i)}(u,v)$, we can update them as in Pathbook-I. For $y_k^{(i)}$, we update it as follows: $y_k^{(i)} = [y_k^{(i-1)} + \eta Y_k^{(i-1)}]^+$, where $Y_k^{(i-1)} = \Lambda_k - \lambda_k^{(i-1)}$.

After the Lagrange multipliers are computed, according to Formula (2.22), we need to find $\overrightarrow{r}^{(i)}$ and $\overrightarrow{b}^{(i)}$ so as to minimize $L(\overrightarrow{r}, \overrightarrow{b}, \overrightarrow{\alpha}^{(i)}, \overrightarrow{\beta}^{(i)}, \overrightarrow{\mu}^{(i)}, \overrightarrow{\omega}^{(i)}, \overrightarrow{y}^{(i)})$. As in Pathbook-I, after some manipulations, we decompose this minimization operation to the following users' behaviors and nodes' behaviors.

In the i^{th} iteration, user k solves the following problem:

$$\begin{aligned} \min \quad & -y_k^{(i)}\lambda_k^{(i)} + \sum_{(u,v)\in E} \beta_k^{(i)}(u,v) r_k^{(i)}(u,v), \\ \text{s.t.} \quad & \sum_{(u,v)\in E} r_k^{(i)}(u,v) - \sum_{(w,u)\in E} r_k^{(i)}(w,u) = h_k^{(i)}(u), u \in V, \\ \text{over} \quad & r_k^{(i)}(u,v) \in [0,C], \forall (u,v) \in E, \end{aligned} \tag{2.31}$$

and node u solves the following problem:

$$\begin{aligned} \min \quad & \Big(l(u) + \sum_{v\in R(u), v\neq s_k} \alpha^{(i)}(v) - \sum_{(u,v)\in E} \beta_k^{(i)}(u,v) p(u,v) \\ & + \mu^{(i)}(u) + \sum_{v\in A(u)} \omega^{(i)}(u,v) - \sum_{v\in A(u)} \omega^{(i)}(v,u) \Big) b_k^{(i)}(u), \\ \text{over} \quad & b_k^{(i)}(u) \in [0,C], u \in V \setminus d_k. \end{aligned} \tag{2.32}$$

Problem (2.32) can be easily solved by each node u. For user problem (2.31), as before we consider an equivalent problem:

$$\begin{aligned} \min \quad & -y_k^{(i)} \sum_{\pi\in P} \gamma_k(\pi) + \sum_{\pi\in P} \kappa_k(\pi)\gamma_k(\pi), \\ \text{over} \quad & \sum_{\pi\in P} \gamma_k(\pi) \in [0,C]. \end{aligned}$$

Solving this equivalent problem will ensure that the min-cost path is always chosen. Thus this problem is transformed to

$$\min \ -y_k^{(i)}\Gamma_k + \kappa_k^{min}\Gamma_k, \text{ over } \Gamma_k \in [0, C], \tag{2.33}$$

where κ_k^{min} is the cost of the min-cost path (with respect to $\beta_k^{(i)}(u,v)$). Therefore, each user k can use a distributed shortest path algorithm to find the min-cost path (with respect to $\beta_k^{(i)}(u,v)$) so as to compute κ_k^{min}, and then solve Problem (2.33). $r_k^{(i)}(u,v)$ is set to the solution to Problem (2.33) if (u,v) is on this min-cost path, and 0 otherwise.

Users and nodes can approximate the primal solutions using (2.29). After the i^{th} iteration, user k uses $\hat{r}_k^{(i)}(u,v)$ as its user behavior on link (u,v) and node u uses $\hat{b}_k^{(i)}(u)$ as its node behavior for user k until the next iteration. We now analyze what information is needed in each iteration. Similar to UPOP, the update operations for $\overrightarrow{r}^{(i)}$, $\overrightarrow{b}^{(i)}$, $\overrightarrow{\alpha}^{(i)}$, $\overrightarrow{\beta}^{(i)}$, $\overrightarrow{\mu}^{(i)}$, and $\overrightarrow{\omega}^{(i)}$ do not require global information. For $y_k^{(i)}$, obviously it can be updated by user k itself. Thus, no global information is required in the iterations.

2.4.3 Algorithm Analysis

2.4.3.1 Optimality Analysis

Theorem 1 states the optimality of our distributed algorithms. Let $\hat{x}^{(i)}$ denote the primal solution approximated in the i^{th} iteration. Note that we reformulate Systems 1 and 2 to the standard form (2.21) for brevity. Here we need to introduce a lemma given by [78], which shows a bound on the Euclidean norms of the Lagrange multipliers generated by ADSM.

Lemma 1. *If 1) the Euclidean norms of the subgradients are bounded by a constant $\mathcal{L}$ (i.e. $\mathcal{L} = \max_{\vec{x}\in\vec{X}} ||g(\vec{x})|| < \infty$) and 2) there exists a vector $\vec{x^s}$ that*

satisfies Slater's condition [7], then for all $i \geq 1$, $||\vec{\delta}^{(i)}||$ *is bounded by a constant* $\varrho = \frac{2}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)})) + \max\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)}) + \frac{\eta\mathcal{L}^2}{2}) + \eta\mathcal{L}\}$. *Note that* $\varpi > 0$ *since* $\vec{x^s}$ *is a Slater vector.* □

Theorem 1. *The following properties hold for all* $i \geq 1$*:*

1. *An upper bound on the amount of constraint violation of the vector* $\hat{x}^{(i)}$ *for System 1 (System 2) is given by* $||g(\hat{x}^{(i)})^+|| \leq \frac{\varrho}{i\eta}$, *where* $\varrho = \frac{2}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)})) + \max\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)}) + \frac{\eta\mathcal{L}^2}{2}) + \eta\mathcal{L}\}$. *For System 1* $\mathcal{L} \leq (|V| + |E|)KC + |V|\Phi + |V|^2\Theta$ *and for System 2* $\mathcal{L} \leq (|V| + |E| + 1)KC + |V|\Phi + |V|^2\Theta$.

2. *An upper bound on* $f(\hat{x}^{(i)})$ *is given by* $f(\hat{x}^{(i)}) \leq f^* + \frac{||\vec{\delta}^{(0)}||^2}{2i\eta} + \frac{\eta\mathcal{L}^2}{2}$, *where* f^* *is the primal optimal value of System 1(System 2).*

3. *A lower bound on* $f(\hat{x}^{(i)})$ *is given by* $f(\hat{x}^{(i)}) \geq f^* - \frac{\varrho^2}{i\eta}$. □

Interpretation. Property 1) in Theorem 1 states that the amount of constraint violation of our primal solution $\hat{x}^{(i)}$ diminishes to zero at the rate $\frac{1}{i}$ as the number of iterations i increases. Properties 2) and 3) imply that as i increases, the primal value of System 1 (System 2) using our solution $\hat{x}^{(i)}$ converges to f^* within an error level $\frac{\eta\mathcal{L}^2}{2}$ at the rate of $\frac{1}{i}$.

Proof. First, we will prove the Slater's condition holds for Systems 1 and 2. For System 1, since $\mathcal{M}^{max}(u) > 0, \mathcal{B}^{max}(u, v) > 0, p(u, v) > 0$ and $C > 0$, we can easily construct a *strictly* feasible solution by allocating each user a very small amount of information flow so that all the constraints are strictly satisfied. For System 2, we need to take care of the additional constraint: $\lambda_k \geq \Lambda_k, k \in [1, K]$. In Section 2.4.4, we discuss how to compute a feasible minimum user rate. If we

set Λ_k to a value that is strictly smaller than the corresponding minimum user rate computed in Section 2.4.4, the Slater's condition also holds for System 2.

Second, we will construct an upper bound on the norms of the subgradients. For brevity, we use $g_1(\overrightarrow{r}, \overrightarrow{b})$, $g_2(\overrightarrow{r}, \overrightarrow{b})$, $g_3(\overrightarrow{r}, \overrightarrow{b})$, $g_4(\overrightarrow{r}, \overrightarrow{b})$, and $g_5(\overrightarrow{r}, \overrightarrow{b})$ to denote Inequalities (2.16), (2.17), (2.18), (2.19), and (2.20), respectively. Note that $g_1(\overrightarrow{r}, \overrightarrow{b})$ consists of $\sum_{k=1}^{K} |V \backslash s_k|$ inequalities, $g_2(\overrightarrow{r}, \overrightarrow{b})$ consists of $\sum_{k=1}^{K} |E|$ inequalities, $g_3(\overrightarrow{r}, \overrightarrow{b})$ consists of $|V|$ inequalities, $g_4(\overrightarrow{r}, \overrightarrow{b})$ consists of $\sum_{v \in V} |A(v)|$ inequalities, and $g_5(\overrightarrow{r}, \overrightarrow{b})$ consists of K inequalities. Due to $b_m(v) \geq 0$, we know $||g_1(\overrightarrow{r}, \overrightarrow{b})|| \leq \sum_{k=1}^{K} |V \setminus s_k| C \leq K|V|C$. Due to $b_k(v) \leq C$, $r_k(u,v) \geq 0$ and $p(u,v) \in (0,1]$, $||g_2(\overrightarrow{r}, \overrightarrow{b})|| \leq \sum_{k=1}^{K} |E|C \leq K|E|C$. Due to $b_k(v) \geq 0$, $||g_3(\overrightarrow{r}, \overrightarrow{b})|| \leq \sum_{u \in V} \mathcal{M}^{max}(u)$. Obviously, $||g_4(\overrightarrow{r}, \overrightarrow{b})|| \leq \sum_{v \in V} \sum_{u \in A(v)} \mathcal{B}^{max}(v,u)$, and $||g_5(\overrightarrow{r}, \overrightarrow{b})|| \leq KC - \sum_{k=1}^{K} \Lambda_k$. Therefore, for System 1 we have

$$
\begin{aligned}
&||g(\overrightarrow{r}, \overrightarrow{b})|| \leq \sum_{j=1}^{4} ||g_j(\overrightarrow{r}, \overrightarrow{b})|| \\
&\leq (|V| + |E|)KC + \sum_{u \in V} \mathcal{M}^{max}(u) + \sum_{v \in V} \sum_{u \in A(v)} \mathcal{B}^{max}(v,u) \\
&\leq (|V| + |E|)KC + |V|\Phi + |V|^2\Theta,
\end{aligned}
$$

and for System 2 we have

$$
\begin{aligned}
&||g(\overrightarrow{r}, \overrightarrow{b})|| \leq \sum_{j=1}^{5} ||g_j(\overrightarrow{r}, \overrightarrow{b})|| \\
&\leq (|V| + |E|)KC + \sum_{u \in V} \mathcal{M}^{max}(u) + \sum_{v \in V} \sum_{u \in A(v)} \mathcal{B}^{max}(v,u) + KC - \sum_{k=1}^{K} \Lambda_k \\
&\leq (|V| + |E| + 1)KC + |V|\Phi + |V|^2\Theta.
\end{aligned}
$$

Therefore, for System 1 we have $\mathcal{L} \leq (|V| + |E|)KC + |V|\Phi + |V|^2\Theta$ and for System 2 we have $\mathcal{L} \leq (|V| + |E| + 1)KC + |V|\Phi + |V|^2\Theta$.

Now we are ready to prove the three properties. This proof follows the proof framework used in [78] to prove the performance of ADSM.

We first prove property 1). According to our algorithm, we have

$$\vec{\delta}^{(i+1)} = [\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)})]^+ \geq \vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)}) \text{ for } i \geq 0.$$

We hence have $\eta g(\vec{x}^{(i)}) \leq \vec{\delta}^{(i+1)} - \vec{\delta}^{(i)}$ for $i \geq 0$. Summing them up over until $i-1$, we have

$$\eta \sum_{m=0}^{i-1} g(\vec{x}^{(m)}) \leq \vec{\delta}^{(i)} - \vec{\delta}^{(0)} \leq \vec{\delta}^{(i)} \text{ for } i \geq 1.$$

Since each of $g_j(\cdot)$ is convex, we have for $i \geq 1$,

$$g(\hat{x}^{(i)}) = g(\frac{1}{i}\sum_{m=0}^{i-1} \vec{x}^{(m)}) \leq \frac{1}{i}\sum_{m=0}^{i-1} g(\vec{x}^{(m)}) \leq \frac{\vec{\delta}^{(i)}}{\eta i}.$$

Considering $\vec{\delta}^{(i)} \geq 0$, we know $g(\hat{x}^{(i)})^+ \leq \frac{\vec{\delta}^{(i)}}{\eta i}$, for all $i \geq 1$. By Lemma 1, we hence have

$$||g(\hat{x}^{(i)})^+|| \leq \frac{||\vec{\delta}^{(i)}||}{\eta i} \leq \frac{\varrho}{i\eta}, \text{ for } i \geq 1.$$

We then prove property 2). Since $f(\cdot)$ is convex, we know

$$\begin{aligned}
& f(\hat{x}^{(i)}) \\
& \leq \frac{1}{i}\sum_{m=0}^{i-1} f(\vec{x}^{(m)}) \\
& = -\frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) + \frac{1}{i}\sum_{m=0}^{i-1} \Big(f(\vec{x}^{(m)}) + \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) \Big) \\
& = -\frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) + \frac{1}{i}\sum_{m=0}^{i-1} q(\vec{\delta}^{(m)}) \\
& \leq -\frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) + q^* \qquad (2.34)
\end{aligned}$$

Additionally, for $m \geq 0$ we know

$$\begin{aligned}
& ||\vec{\delta}^{(m+1)}||^2 \\
& = ||[\vec{\delta}^{(m)} + \eta g(\vec{x}^{(m)})]^+||^2 \\
& \leq ||\vec{\delta}^{(m)} + \eta g(\vec{x}^{(m)})||^2 \\
& = ||\vec{\delta}^{(m)}||^2 + ||\eta g(\vec{x}^{(m)})||^2 + 2\eta \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}).
\end{aligned}$$

It implies

$$-\vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) \leq \frac{||\vec{\delta}^{(m)}||^2 + ||\eta g(\vec{x}^{(m)})||^2 - ||\vec{\delta}^{(m+1)}||^2}{2\eta}.$$

Summing this inequality over $m \in [0, i-1]$ for $i \geq 1$, we have

$$-\frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) \leq \frac{||\vec{\delta}^{(0)}||^2 - ||\vec{\delta}^{(i)}||^2}{2i\eta} + \frac{\eta \sum_{m=0}^{i-1} ||g(\vec{x}^{(m)})||^2}{2i}.$$

Substituting this into Inequality (3.18), we have

$$\begin{aligned} f(\hat{x}^{(i)}) &\leq q^* + \frac{||\vec{\delta}^{(0)}||^2 - ||\vec{\delta}^{(i)}||^2}{2i\eta} + \frac{\eta \sum_{m=0}^{i-1} ||g(\vec{x}^{(m)})||^2}{2i} \\ &\leq q^* + \frac{||\vec{\delta}^{(0)}||^2}{2i\eta} + \frac{\eta \mathcal{L}^2}{2} = f^* + \frac{||\vec{\delta}^{(0)}||^2}{2i\eta} + \frac{\eta \mathcal{L}^2}{2}. \end{aligned} \tag{2.35}$$

Note that Inequality (2.35) holds because of the Slater's condition.

We finally prove property 3). Given a dual optimal solution $\vec{\delta}^*$, we have

$$f(\hat{x}^{(i)}) = f(\hat{x}^{(i)}) + \vec{\delta}^* \cdot g(\hat{x}^{(i)}) - \vec{\delta}^* \cdot g(\hat{x}^{(i)}) \geq q(\vec{\delta}^*) - \vec{\delta}^* \cdot g(\hat{x}^{(i)}).$$

Additionally, since $\vec{\delta}^* \geq 0$ and $g(\hat{x}^{(i)})^+ \geq g(\hat{x}^{(i)})$, we have

$$-\vec{\delta}^* \cdot g(\hat{x}^{(i)}) \geq -\vec{\delta}^* \cdot g(\hat{x}^{(i)})^+ \geq -||\vec{\delta}^*||||g(\hat{x}^{(i)})^+||.$$

Combining the two inequalities above, we have

$$f(\hat{x}^{(i)}) \geq q(\vec{\delta}^*) - ||\vec{\delta}^*||||g(\hat{x}^{(i)})^+|| \geq q^* - \frac{\varrho^2}{i\eta} = f^* - \frac{\varrho^2}{i\eta}.$$

■

2.4.3.2 Dual Space Information Analysis

Property 1) in Theorem 1 indicates that our algorithm cannot guarantee absolute primal feasibility, although the constraint violation keeps decreasing as our algorithm iterates. In this subsection, we specifically analyze the load constraint violation using dual space information.

Next we define two metrics to quantify the node max load violation and the node load balance violation.

For node u, we define the *node max load ratio* by

$$\mathcal{R}_m^{(i)}(u) = \frac{\sum_{m=0}^{i-1} \sum_{k\in[1,K]} b_k^{(m)}(u)}{i\mathcal{M}^{max}(u)}, \tag{2.36}$$

and the *node load balance ratio* by

$$\mathcal{R}_b^{(i)}(u) = \frac{\sum_{m=0}^{i-1} \sum_{k\in[1,K]} (b_k^{(m)}(u) - b_k^{(m)}(v))}{i\mathcal{B}^{max}(u,v)}. \tag{2.37}$$

We have the following theorem to bound these two ratios at each iteration using dual space information.

Theorem 2. *For any $i \geq 1$,*

$$\mathcal{R}_m^{(i)}(u) \leq 1 + \frac{\mu^{(i)}(u) + \eta\sum_{k\in[1,K]} b_k^{(0)}(u)}{i\mathcal{M}^{max}(u)\eta}; \tag{2.38}$$

$$\mathcal{R}_b^{(i)}(u) \leq 1 + \frac{\omega^{(i)}(u,v) + \eta \sum_{k\in[1,K]} \left(b_k^{(0)}(u) - b_k^{(0)}(v)\right)}{i\mathcal{B}^{max}(u,v)\eta}. \tag{2.39}$$

□

Interpretation. Theorem 2 shows the bounds on the node max load ratio and the node load balance ratio using dual variables. According to Lemma 1, it is easy to verify that $\mu^{(i)}(u)$ and $\omega^{(i)}(u,v)$ are bounded by $\mathcal{L}$. Therefore, the upper bounds on the node max load ratio and the node load balance ratio approach 1 at the rate of $\frac{1}{i}$.

Proof. We use τ to denote the length of each iteration. Let $\mathcal{N}^{(i)}(u) = \sum_{m=0}^{i-1} \tau \left(\sum_{k\in[1,K]} b_k^{(m)}(u) \ - \mathcal{M}^{max}(u)\right)$ and $A = \eta \left(\sum_{k\in[1,K]} b_k^{(0)}(u) - \mathcal{M}^{max}(u)\right)$. We claim that for $i \geq 1$

$$\mathcal{N}^{(i)}(u) \leq \frac{\tau}{\eta}\mu^{(i)}(u) + \frac{\tau}{\eta}A. \tag{2.40}$$

Since $\mu^{(i)}(u) \geq 0$, Inequality (2.40) trivially holds for $i = 1$. Suppose that Inequality (2.40) holds for some $i \geq 1$. We will prove that Inequality (2.40) also holds for $i + 1$ as follows.

$$\begin{aligned}
\mathcal{N}^{(i+1)}(u) &= \mathcal{N}^{(i)}(u) + \tau \left(\sum_{k\in[1,K]} b_k^{(i)}(u) - \mathcal{M}^{max}(u) \right) \\
&\leq \frac{\tau}{\eta}\mu^{(i)}(u) + \frac{\tau}{\eta}A + \tau \left(\sum_{k\in[1,K]} b_k^{(i)}(u) - \mathcal{M}^{max}(u) \right) \\
&\leq \frac{\tau}{\eta} \left[\mu^{(i)}(u) + \eta \left(\sum_{k\in[1,K]} b_k^{(i)}(u) - \mathcal{M}^{max}(u) \right) \right]^+ + \frac{\tau}{\eta}A \\
&= \frac{\tau}{\eta}\mu^{(i+1)}(u) + \frac{\tau}{\eta}A.
\end{aligned}$$

Therefore, claim (2.40) is proved. This claim further implies that

$$\frac{\sum_{m=0}^{i-1} \sum_{k\in[1,K]} b_k^{(m)}(u)}{i\mathcal{M}^{max}(u)} \leq 1 + \frac{\mu^{(i)}(u) + A}{i\mathcal{M}^{max}(u)\eta}. \tag{2.41}$$

Therefore, Inequality (2.38) is proved. The proof of Inequality (2.39) is similar and thus omitted. ■

2.4.4 Discussion

In this subsection, we discuss some practical issues.

Mixed-duplex networks In this work, we assume that all the nodes are equipped with full-duplex transceivers. In practice, some nodes may be equipped with traditional half-duplex transceivers. We call such wireless networks, where each node is equipped with either a half-duplex transceiver or a full-duplex tran-sceiver, *mixed-duplex networks*. According to the collision-free broadcast model (i.e., Inequality (4) in [115]), we know that for traditional half-duplex wireless networks, the node

average broadcast rate should satisfy

$$\sum_{m\in[1,K]} b_m(u) + \sum_{m\in[1,K]} \sum_{v\in R(u)} b_m(v) \leq C, \forall u \in V \setminus s_k. \tag{2.42}$$

Combining Inequalities (2.42) and (2.8), we know that for a mixed-duplex network, the node average broadcast rate should satisfy

$$\sum_{m\in[1,K]} D(u) b_m(u) + \sum_{m\in[1,K]} \sum_{v\in R(u)} b_m(v) \leq C, \forall u \in V \setminus s_k, \tag{2.43}$$

where $D(u)$ is a binary indicator variable, which is equal to 1 if u is equipped with a half-duplex transceiver, and 0 otherwise. Therefore, to solve UPOP and NPCOP for mixed-duplex networks, Constraint (2.16) in System 1 and System 2 should be replaced by Constraint (2.43). We can use distributed algorithms that are similar to Pathbook-I and Pathbook-II to solve these two optimization systems for mixed-duplex networks, respectively.

Transmission credit mechanism For each user k, our solution tells each node the optimized information rate $r_k(u)=\sum_{(u,v)\in E} r_k(u,v)$ and the optimized broadcast rate $b_k(u)$. We assume that a node should be triggered to transmit only when it receives a packet, and should perform the transmission only when the MAC permits. We need a transmission credit $TX_k(u)$, computed as $TX_k(u)=\frac{b_k(u)}{r_k(u)}$ if $r_k(u)>0$ and 0 otherwise. Once node u receives an innovative packet (defined in Section 2.2.1) for user k from an upstream node, node u increments the associated *credit counter* for user k by $TX_k(u)$. When a transmission is allowed by the MAC, node u selects a user to serve by using a weighted round-robin algorithm [49] with each user k having a weight of $\frac{r_k(u)}{\sum_{i\in[1,K]} r_i(u)}$. Node u then checks whether the credit counter associated with this user is positive. If positive, node u generates a coded packet, broadcasts this packet, and decrements the credit counter by one. Otherwise, it picks a different user. If all the credit counters are non-positive,

node u keeps silent. Since a full-duplex MAC is still an open research area, our transmission credit mechanism also provides a possible way to integrate our work into some future full-duplex MACs.

Computing feasible minimum user rates The minimum user rates should be carefully selected so that no user will request a minimum user rate demand that is too large. Otherwise, the resources in the network may not satisfy these users' requirements. We propose the following two computation methods.

When a network is being deployed, the network administrator needs to initially analyze this network and compute the max-min user rate $\hat{\Lambda}$. The following system provides a way to compute the max-min user rate.

$System3(\overrightarrow{r}, \overrightarrow{b})$:

$$\begin{aligned} \max \quad & \hat{\Lambda}, \\ \text{s.t.} \quad & (2.15) - (2.19) \ \text{ and } \ \lambda_k \geq \hat{\Lambda}, \forall k \in [1, K]. \end{aligned}$$

It is easy to see that if each node sets its minimum user rate demand to a value less than $\hat{\Lambda}$, System 2 has a feasible solution. Although it seems that this method uses a central network administrator to solve System 3, this mechanism is still practical, since we only need to compute the max-min user rate *once* in the initial phase. Thus users can still adaptively adjust their minimum user rate demands when this network is being used. The nodes can also adjust their constraints as long as these constraints are not tighter than those used in System 3. Therefore, whenever the minimum user rate demands and node constraints change, the run-time optimization for System 2 can still be done in a distributed manner.

Although the first method is very efficient, a central administrator is involved. An alternative method is using System 1 as a starting point to estimate

the feasible minimum user rate demands. Specifically, users and nodes first run Pathbook-I to solve System 1. After enough iterations, a solution is obtained. Based on this solution, users can estimate the feasible minimum user rate demands. In order to obtain fair minimum user rate demands, we can compute a Nash bargaining point [29] of user rates by setting $\mathcal{U}_k(\lambda_k) = \ln \lambda_k$ and $\sigma_k(u) = 0$. The current solution to System 1 is a Nash bargaining point which captures the notion of social efficiency and fairness.

Information exchanges We first consider the information exchanges for maintaining node load balance. To set up a load balance area, each node runs a distributed node discovery algorithm [39] to discover other nodes (such as H-hop neighbors). Through our previous analysis, we know that a large load balance area will lead to many information exchanges. The following scheme can be used to reduce the information exchanges by roughly reducing the overlap among the load balance areas of different nodes. Suppose that node u_1 wants to keep a balanced load with node u_2 (i.e. $|b(u_1) - b(u_2)| \leq \mathcal{B}^{max}(u_1, u_2)$), and that u_1 knows another node u_3 (closer to u_1 than u_2 with respect to hops) has a constraint $|b(u_2) - b(u_3)| \leq \mathcal{B}^{max}(u_2, u_3)$ and $\mathcal{B}^{max}(u_1, u_2) \geq \mathcal{B}^{max}(u_2, u_3)$. Node u_1 hence sets $|b(u_1) - b(u_3)| \leq \mathcal{B}^{max}(u_1, u_3) = \min\{\mathcal{B}^{max}(u_1, u_3), \mathcal{B}^{max}(u_1, u_2) - \mathcal{B}^{max}(u_2, u_3)\}$, and excludes u_2 from its load balance area. The information exchange between u_1 and u_2 is hence not needed any more.

We now consider other information exchanges. In each iteration, each node broadcasts the iteration stepsize, the iteration interval, the stopping criterion, the information of the current iteration (i.e. computed user and node behaviors, and Lagrange multipliers). Other nodes can hence obtain the necessary information for the next iteration by overhearing others' broadcasts. Considering that network

topology and user sessions may change, we can divide the timeline into a sequence of intervals of a constant length. All the nodes and users roughly restart Pathbook-I or Pathbook-II at the beginning of each interval.

Extensions The analytical model and system used in this work may be extensible when new requirements are taken into account. Here we present two extensions.

The fist extension is looking for a Nash bargaining point [29] of user utility $\mathcal{U}_k(\lambda_k)$, which can be formulated as

$System4(\overrightarrow{r}, \overrightarrow{b})$:

$$\max \sum_{k\in[1,K]} \ln \mathcal{U}_k(\lambda_k), \qquad \text{s.t.}(2.15) - (2.19).$$

Note that a Nash bargaining point captures the notion of social efficiency and fairness. In order to solve this system using Pathbook-I, we can *assume* that all the $\sigma_k(u)$'s in System 1 are 0, and that $\mathcal{U}_k(\lambda_k)$ in System 1 actually is equal to $\ln \mathcal{U}_k(\lambda_k)$ in System 4. In this way, we can use Pathbook-I to solve System 4.

The second extension is optimizing the user profit subject to an additional user power consumption constraint $\sum_{u\in V\setminus d_k} l(u)b_k(u) \leq \boldsymbol{c}_k$. This problem can be formulated as

$System5(\overrightarrow{r}, \overrightarrow{b})$:

$$\begin{aligned} \max \quad & \sum_{k\in[1,K]} \Big(\mathcal{U}_k(\lambda_k) - \sum_{u\in V\setminus d_k} \sigma_k(u)b_k(u)\Big), \\ \text{s.t.} \quad & (2.15) - (2.19), \text{ and } \sum_{u\in V\setminus d_k} l(u)b_k(u) \leq \boldsymbol{c}_k. \end{aligned}$$

A distributed algorithm similar to Pathbook-I can be applied to solve this problem and the details are hence omitted.

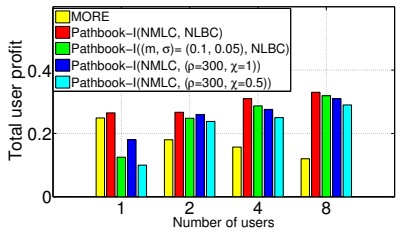
2.5 Performance Evaluation

We implemented OMR and compared its performance with that of an implementation of MORE [9]. Since to the best of our knowledge our work represents the first attempt towards the design of joint optimization algorithms on MAC and routing for full-duplex wireless networks, we assume that the nodes in MORE still adopt half-duplex transmission. We uniformly distributed 20 nodes in a $1000m \times 1000m$ square region. As in [27], we assume that the packet delivery ratio from a node u to a node v is inversely proportional to their distance $d(u, v)$ with a random Gaussian deviation of 0.1. If this packet delivery ratio is greater than 0.1, we say that v is in u's transmission range. We set the channel capacity C to 1Mbps. The number of users was chosen to be 1, 2, 4, and 8. For each user, we randomly chose two different nodes as its source and destination. In Fig.2.2, NMLC and NLBC denote the case where node max load constraints were not considered and the case where node load balance constraints were not considered, respectively. For node max load constraints, we normally generated each node u's $\mathcal{M}^{max}(u)$ with a mean of mMbps and a variance of σMbps. Although u has its maximum load constraint $\mathcal{M}^{max}(u)$, in our experiments we allowed the real broadcast rates to be greater than $\mathcal{M}^{max}(u)$. We computed a *violation ratio* $\frac{b(u)}{\mathcal{M}^{max}(u)}$ for each node to quantify the constraint violation. We chose among all the nodes the worst (largest) violation ratio as the *network violation ratio*. In Fig.2.2, we use ρ to denote the load balance area range of each node. If $d(u, v) \leq \rho$, u(or v) is in v(or u)'s load balance area. In order to quantify the load balance improvement, we used the *fairness index* proposed in [45], defined by $\mathcal{F} = \frac{\left(\sum_{u \in V} b(u)\right)^2}{|V| \sum_{u \in V} b(u)^2}$.

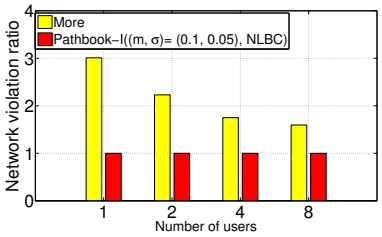
Figs. 2.2a-2.2c compare the results obtained by using MORE and Pathbook-I. We used $\ln(1 + \lambda_k)$ as user k's utility function, and the service price $\sigma_k(u)$ was

set to 0.01 per $Mbps$. If u (or v) was in v (or u)'s load balance area, $\mathcal{B}^{max}(u,v)$ and $\mathcal{B}^{max}(v,u)$ were set to $0.001\chi\times d(u,v)Mbps$, where the value of χ is shown in figures. In the experiments, all the users in MORE selfishly tried to transmit data as much as possible. Fig. 2.2a shows that compared with MORE, Pathbook-I increases total user profit by 6%-175% when no node constraint is considered. Figs. 2.2b-2.2c show that Pathbook-I achieves 37%-67% smaller network violation ratio and 9%-57% higher node load fairness index. This significant performance improvement is as expected, because our analytical system considers joint resource allocation by optimizing both users and nodes behaviors.

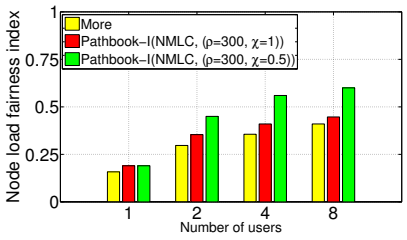
Figs. 2.2d-2.2f compare the results obtained by using MORE and Pathbook-II. The minimum user rate demands were set to 0.01Mbps for all the users. If u (or v) was in v (or u)'s load balance area, $\mathcal{B}^{max}(u,v)$ and $\mathcal{B}^{max}(v,u)$ were set to $0.0001\chi\times d(u,v)Mbps$, where the value of χ is shown in figures. The power consumption ratio $l(u)$ was randomly chosen from (0, 1] for each node u. Note that although full-duplex transmission introduces more operations [17], we can still assume that a node's power consumption ratios are the same in both half-duplex and full-duplex modes. This is because the major power consumption comes from the data transmission, and although two transmit antennas are being used by a full-duplex node [17], each transmit antenna radiates half the power in order to ensure the same total radiated power as with one transmit antenna used in the half-duplex operation. We define a metric, called *network power efficiency*, which is computed as $\sum_{k=1}^{K}\lambda_k/\mathcal{P}(\overrightarrow{b})$. It represents the achieved total user rate per unit power consumption. In Figs. 2.2d-2.2f "MORE-limited" and "MORE-unlimited" represent the case where all the users transmitted data following their minimum user rate demands, and the case where all the users selfishly tried to transmit data as much as possible, respectively. As shown in Figs. 2.2d-2.2f, we observe



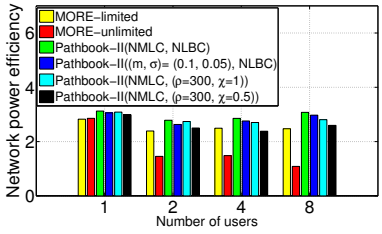
(a) Total user profit (System 1)



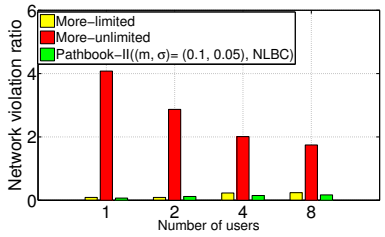
(b) Network violation ratio (System 1)



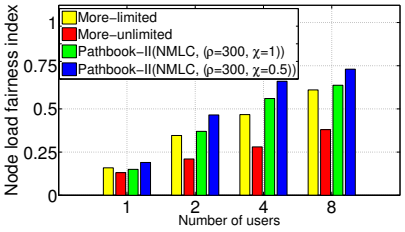
(c) Node load fairness index (System 1)



(d) Network power efficiency (System 2)



(e) Network violation ratio (System 2)



(f) Node load fairness index (System 2)

Figure 2.2: Numerical results

that compared with MORE-limited, Pathbook-II achieves 10%-24% higher network power efficiency (when no node constraint is considered), 21%-35% smaller network violation ratio, and 6%-41% higher node load fairness index (when the number of users is no less than 2). Compared with MORE-unlimited, Pathbook-II achieves 10%-183% higher network power efficiency, much smaller network violation ratio, and 15%-135% higher fairness index.

2.6 Conclusion

In this work, we have studied cross-layer optimization of MAC and routing in full-duplex wireless networks, comprehensively considering various resource and social constraints. We have proposed a collision-free full-duplex broadcast MAC

and proved its necessary and sufficient conditions. We have concentrated on the user profit optimization problem and the network power consumption optimization problem, and have formulated these two problems as convex programming systems. By combining Lagrangian decomposition and subgradient methods, we have proposed distributed iterative algorithms to solve these two systems, which compute the optimized user information flow (i.e. user behavior) for the network layer and the optimized node broadcast rate (i.e. node behavior) for the MAC layer. Our algorithms allow each user and each node to adjust its behavior individually in each iteration. We have analyzed the algorithm convergence, and have provided bounds on the amount of constraint violation, and the gap between our solution and the optimal solution at each iteration. We have also used the dual space information to analyze the node load constraint violation at each iteration. Simulation results show that our algorithm can significantly increase user profit, node load fairness and network power efficiency, and reduce network violation ratio.

Chapter 3

RACER: Resource Allocation in Load-Constrained Multihop Wireless Networks

3.1 Introduction

3.1.1 Motivation

The fact that the wireless spectrum is a limited resource motivates us to investigate how to use wireless resources effectively. A wireless network has two kinds of network entities – *users* and *nodes*, where a user maintains a session (multipath flow) from a source node to a destination node, and uses intermediate nodes to forward the packets.

From the perspective of users, first, a user can obtain an amount of *utility* if a certain information rate is allocated to it. An optimized user transmission rate can not only stabilize a network (without causing network congestion) but also improve the actual obtained utility. Second, when there exist multiple users in a wireless network, we have to solve the resource competition among users in order to improve the total resource utilization, since individual user behaviors may not lead to a global optimum. Third, different users may belong to different quality of service (QoS) classes and may have different *user QoS rate constraints*. One typical requirement is that the achievable information rate by each QoS class must be in a range prescribed in the service level agreement [80]. Therefore, a resource allocation mechanism needs to differentiate users in different QoS classes.

From the perspective of nodes, we consider two kinds of node constraints that will affect node behaviors. These constraints characterize node individual requirements and social requirements, respectively. The first constraint is called *node max load constraint*, *i.e.*, the maximum load this node is willing to carry. This constraint is of importance because each node may have its own transmission

capacity limit or energy consumption concern. The second constraint is called *node load balance constraint*. Load balancing is regarded as an important issue in network design and optimization [28, 30, 74, 84, 85, 117]. Without considering the node load balance, the traditional routing design methodology, that the shortest route is always chosen, may result in congestion on the center of a network or hotspots. A congested area or a hotspot will drain the energy from the nodes in these areas much faster [57, 74]. We introduce the node load balance constraint to control the load relationships among nodes. A node may set a maximum allowed load difference compared with other nodes, such as H-hop neighbors. This constraint ensures that a node will not carry too much more load than other nodes, and hence smooths the load among different nodes.

From the perspective of both users and nodes, a node may set a service level it is willing to provide for a user. This is called *node-user load constraint*. For example, after receiving the bandwidth requests from users, a node decides the largest bandwidth it may provide for each user according to these requests and its own interest. This constraint profiles a social relationship between users and nodes with respect to loads.

These four constraints (user QoS rate constraints, node max load constraints, node load balance constraints, and node-user load constraints) are called the *load constraints* of a wireless network. A multihop wireless network with load constraints is called a *load-constrained multihop wireless network*. Note that we do not require all the network entities to have these constraints. We integrate these constraints into a network for a practical purpose, *i.e.*, they provide heterogeneous network entities with the flexibility to adjust their individual requirements (user QoS rate constraint and node max load constraint) and social requirements (node load balance constraint and node-user load constraint). How-

ever, integrating these constraints makes resource allocation much harder due to the complicated relationships introduced among the network entities.

In this work, we study the problem of allocating network resources to maximize the total user utility in a load-constrained wireless network.

3.1.2 Contribution

Our contribution is two-fold. The first contribution is a theoretical optimization framework, which may be applicable for many networking optimization problems. We propose a new subgradient optimization framework that has the following property. Given an approximation/optimal algorithm for solving the subproblem at each iteration, the framework leads to a result that can provide the following bounds at each iteration: (a) the bounds on the Lagrangian multipliers; (b) the bound on the amount of feasibility violation of the generated primal solutions; and (c) the upper and lower bounds on the gap between the optimal solution and the generated primal solutions. Note that standard subgradient methods require one to solve a sub-problem optimally at each iteration. However, in practice this sub-problem could be an NP-hard problem, which makes efficiently computing an optimal solution impossible. This problem was studied by Lin and Shroff [63] and Chen *et al.* [13]. However, the focus of these works is more on the asymptotic behavior of the primal sequences. Our framework further advances this research by providing fine-grained convergence, optimality, and dual space information *at each iteration*. We believe that this framework can provide a useful theoretical foundation for many networking optimization problems.

Second, we formulate the resource allocation problem as a convex programming system. Based on our α-approximation dual subgradient algorithm, we present a distributed iterative algorithm, which allows each user and each node

to individually adjust its own behavior in each iteration period. This feature is of great importance for network scalability and self-organization. We prove the bounds on the amount of feasibility violation and the gap between our solution and the optimal solution *at each iteration*. We provide the bounds on node queue lengths, user utility deficits, and node load violation ratios *at each iteration* using dual space information.

3.1.3 Related Work

3.1.3.1 Network Resource Optimization

Optimizing network utility is an important objective in networking applications [13, 26, 47, 63, 64, 67, 68, 80, 81, 95, 106] (see Chiang *et al.* [16] and Shakkottai and Srikant [95] for more discussions on this subject). In this work, we extend the scope of resource allocation to load-constrained wireless networks, considering various resource and social requirements.

Note that in literature there are some works focused on network load optimization to improve, for example, energy efficiency, load balance, and network lifetime. Chang and Tassiulas [10] used linear programming to capture the issue of power consumption. Ganjali and Keshavarzian [30] showed that multi-path routing can balance loads only if a very large number of paths are used. Popa *et al.* [85] showed that an optimum routing scheme based on the shortest paths can be computed by using linear programming. Fang *et al.* [28] studied the optimization of opportunistic routing subject to node constraints. Zorzi and Rao [117] solved the energy efficiency issues by balancing the load reactively.

The difference between our work and the above line of research is that we study the utility optimization problem taking into account various network entity resource and social requirements. We present a distributed resource allocation

algorithm with provable bounds on the gap between our solution and the optimal solution, queue lengths, user utility deficits, and node load violation ratios at each iteration.

3.1.3.2 Subgradient Methods

In networking applications, subgradient methods have been used with great success in developing distributed cross-layer resource allocation mechanisms (e.g. [13, 28, 47, 62, 63, 68, 80, 81, 95, 115]). See Nedić and Ozdaglar [78] for more discussions on various schemes in the subgradient method family. Typically, standard subgradient methods require one to *optimally* solve a sub-problem at each iteration. However, in practice solving this sub-problem in a polynomial time may be impossible. Lin and Shroff [63] and Chen *et al.* [13] show such applications: the scheduling component in the optimal cross-layered rate control scheme requires solving at each iteration a global optimization problem that is usually quite difficult.

Taking advantage of the primary solution recovery technique in [78], we present an α-approximation dual subgradient algorithm, in which an approximation to the sub-problem at each iteration is allowed to be used. Our technique enables us to prove a bound on the Lagrange multipliers, a bound on the amount of feasibility violation, and upper and lower bounds on our solution at each iteration. Note that the scheme in [78] is a special case of our algorithm (i.e., $\alpha = 1$).

3.2 Model

3.2.1 Wireless Network Model

We consider a multi-hop wireless network with K users. We model the wireless network as a directed graph $G = (\mathcal{V}, E)$, where E is a set of links and $\mathcal{V}$ is a set of nodes. For each user, a distributed node pre-selection procedure (e.g. [9]) is performed to add the nodes, which are reachable from the source node of this user and can reach its destination node, into its intermediate forwarder set.

Let $G(\mathcal{V}_k, E_k)$ denote the resulting topology for user k, where $\mathcal{V}_k$ and E_k, respectively, are the set of nodes and the set of directed links involved in the session of user k. Let $r_k(u, v)$ (or $r_k(e)$, respectively) denote the rate of user k's flow on link (u, v) (or link e, respectively). We use $R_k(u)$=$\sum_{(u,v)\in E_k} r_k(u, v)$ ($\overline{R}_k(u)$=$\sum_{(v,u)\in E_k} r_k(v, u)$, respectively) to denote the rate of user k's flow outgoing from (incoming to, respectively) node $u \in \mathcal{V}_k$. For notation consistency, $R_k(u)$ and $\overline{R}_k(u)$ are set to 0 if $u \notin \mathcal{V}_k$, and $r_k(u, v)$ is set to 0 if $(u, v) \notin E_k$. Let $R(u)$ denote $\sum_{k=1}^{K} R_k(u)$. Let λ_k, s_k, and d_k denote the information rate, the source, and the destination of user k. We use the following to model the flow of user $k = 1, ..., K$:

$$R_k(u) - \overline{R}_k(u) - h_k(u) \geq 0, \forall u \in \mathcal{V}_k, \tag{3.1}$$

where $h_k(u)$ equals λ_k if $u = s_k$, $-\lambda_k$ if $u = d_k$, and 0 otherwise. For user k, (3.1) specifies the relationship among the rates of incoming flow $\overline{R}_k(u)$, outgoing flow $R_k(u)$, and generating flow $h_k(u)$ at each node $u \in \mathcal{V}_k$. It is similar to the multicommodity flow model for the resource allocation rate constraint used in [13, 62].

3.2.2 Scheduling Model

Due to the wireless link interference, we need to consider the scheduling feasibility problem. In this work, we focus on the following two interference models.

3.2.2.1 General Interference Model (GIM)

We consider two widely used interference models: Protocol Model (PM) [36, 43, 104] and RTS/CTS Model (RCM) [3, 13, 43, 104]. In PM, when there is a single wireless channel, a transmission from node u to node v is successful if and only if 1) v is in u's transmission range and 2) v is not in the interference range of any other transmitting node. In RCM, for every pair of simultaneous communication links (say (u, v) and (w, y)), it should satisfy that 1) they are four distinct nodes; and 2) neither u nor v is in the interference ranges of w or y, and vice versa. In brief, for a successful transmission, PM requires only the receiver to be free of interference, while RCM requires both the sender and the receiver to be free of interference (e.g. in 802.11 MAC).

3.2.2.2 Primary Interference Model (PIM)

PIM, also known as node-exclusive spectrum sharing, has also been widely used in literature [5, 8, 12, 13, 35, 37, 52, 63, 75, 96, 102, 111]. In this model, links that share a common node cannot transmit simultaneously, but links that do not share nodes can do so. PIM, for example, models a wireless network with multiple frequencies/codes available for transmission (using FDMA/CDMA), and enables parallel communications in a neighborhood using such orthogonal FDMA/CDMA channels. See [102] for more discussions.

For PM, RCM, and PIM, we can use standard techniques [13, 43] to con-

struct a *conflict graph* C_G to capture the contention relations among different links. Each vertex in C_G represents a link in the original graph $G(\mathcal{V}, E)$, and an edge between two vertices represents the conflict between the two corresponding links in $G(\mathcal{V}, E)$: *these links cannot transmit at the same time* in the corresponding interference models (i.e. PM, RCM, or PIM). We will use the terms node and link in reference to $G(\mathcal{V}, E)$ while reserving the terms vertex and edge for C_G.

Each *independent set* in C_G represents a set of links in $G(\mathcal{V}, E)$ that can transmit simultaneously without collision. Note that an independent set is a set of vertices which are pairwise non-adjacent. Let $\mathcal{I}$ denote the set of all the independent sets in C_G, and let $\mathbb{C}(e)$ denote the capacity of link $e \in E$. We denote an independent set $I \in \mathcal{I}$ as an $|E|$-dimensional rate vector $\vec{r}^I$, where the e-th entry is $\vec{r}^I(e)=\mathbb{C}(e)$ if link e's corresponding vertex over C_G is in I, and $\vec{r}^I(e)=0$ otherwise. Thus, the feasible region $\prod$ of an $|E|$-dimensional link rate vector $\vec{r}$, whose e-th entry is $\sum_{k=1}^{K} r_k(e)$, is defined by the convex hull of these rate vectors:

$$\prod = \{\vec{r} : \vec{r} = \sum_{I \in \mathcal{I}} c^I \vec{r}^I, c^I \geq 0, \sum_{I \in \mathcal{I}} c^I = 1\}. \tag{3.2}$$

For example, we have a network with three links e_1, e_2, and e_3, whose capacities are 1, 2, and 3, respectively. Suppose that $\{e_1\}$, $\{e_2\}$, $\{e_3\}$, and $\{e_1, e_3\}$ form all the independent sets I_1, I_2, I_3, and I_4 in C_G. Then, $\vec{r}^{I_1}$=(1, 0, 0), $\vec{r}^{I_2}$=(0, 2, 0), $\vec{r}^{I_3}$=(0, 0, 3), and $\vec{r}^{I_4}$=(1, 0, 3). The feasible region $\prod = \{\vec{r} : \vec{r} = c^{I_1}\vec{r}^{I_1} + c^{I_2}\vec{r}^{I_2} + c^{I_3}\vec{r}^{I_3} + c^{I_4}\vec{r}^{I_4}, c^{I_1} \geq 0, c^{I_2} \geq 0, c^{I_3} \geq 0, c^{I_4} \geq 0, c^{I_1} + c^{I_2} + c^{I_3} + c^{I_4} = 1\}$.

3.2.3 User Utility Model

If an information rate λ_k is allocated to user k, then user k can have a utility of $U_k(\lambda_k)$, where the user-defined $U_k(\cdot)$ is an increasing, concave, and continually

differentiable function, and $U_k(0) = 0$.

3.2.4 Load Constraint Model

We consider four constraints.

User QoS Rate Constraint We consider the following requirement: the user rate achieved by each QoS class cannot exceed a limit prescribed in the service level agreement. For example, in a network preferring to allocate resources to realtime services, a video service may have a higher rate limit while other non-realtime services, such as FTP, may have a lower rate limit. This requirement can be formulated as:

$$\lambda_k \leq \mathbb{Q}_k, \forall k \in [1, K], \tag{3.3}$$

where $\mathbb{Q}_k > 0$ is the rate limit set for user k.

Node Max Load Constraint The node max load constraint can be expressed as:

$$R(u) \leq \mathbb{M}(u), \forall u \in \mathcal{V}, \tag{3.4}$$

where $\mathbb{M}(u)>0$ is the max load that node u is willing to carry.

Node Load Balance Constraint A node may set a limit on at most how much more of the load it is willing to carry, compared with other nodes (such as H-hop neighbors). We use $\mathcal{A}(u)$ to denote the set of such nodes (called *load balance area*), which node u wants to keep balanced load with. Thus, this constraint can be expressed as:

$$R(u) - R(w) \leq \mathbb{B}(u, w), \forall w \in \mathcal{A}(u), \forall u \in \mathcal{V}, \tag{3.5}$$

where $\mathbb{B}(u,w)>0$ is set by node u, which denotes at most how much more of the load u is willing to carry, compared with that carried by $w \in \mathcal{A}(u)$. We assume that $w \in \mathcal{A}(u) \Leftrightarrow u \in \mathcal{A}(w)$ and $\mathbb{B}(u,w) = \mathbb{B}(w,u)$ to simplify the expressions in the following sections. However, there would be no big difference for the algorithm if this assumption does not hold.

Node-User Rate Constraint The rate requirement enforced by node u on user k can be expressed as:

$$R_k(u) \leq \mathbb{N}_k(u), \forall u \in \mathcal{V}_k, \forall k \in [1,K], \tag{3.6}$$

where $\mathbb{N}_k(u) > 0$ is the rate limit on the information flow of user k that goes through node u.

3.3 Problem Formulation

Based on the models above, we formulate our resource allocation problem as the following convex programming:

$$\max\ U(\vec{\lambda}) = \sum_{k=1}^{K} U_k(\lambda_k) \tag{3.7}$$

$$\text{s.t. } R_k(u) - \overline{R}_k(u) - h_k(u) \geq 0, \forall u \in \mathcal{V}_k, \forall k \in [1,K]; \tag{3.8}$$

$$\lambda_k \leq \mathbb{Q}_k, \forall k \in [1,K]; \tag{3.9}$$

$$R(u) \leq \mathbb{M}(u), \forall u \in \mathcal{V}; \tag{3.10}$$

$$R(u) - R(w) \leq \mathbb{B}(u,w), \forall w \in \mathcal{A}(u), \forall u \in \mathcal{V}; \tag{3.11}$$

$$R_k(u) \leq \mathbb{N}_k(u), \forall u \in \mathcal{V}_k, \forall k \in [1,K]; \tag{3.12}$$

$$\text{over } \vec{r} \in \prod. \tag{3.13}$$

Remark:

1. We do not require all the users and all the nodes to have all of these constraints. In that case, we remove the corresponding constraints from the system above.

2. Although this system can be solved using traditional convex programming techniques [7], a distributed algorithm is preferable for the purpose of practical implementations. We next present a distributed iterative algorithm and analyze its performance at each iteration.

3. If we want to consider the proportional fairness among users [115], we can replace the objective function $\sum_{k=1}^{K} U_k(\lambda_k)$ by $\sum_{k=1}^{K} \log U_k(\lambda_k)$. The algorithm solving this objective is similar to the distributed algorithm solving the above system.

3.4 α-Approximation Dual Subgradient Method

We first present a new subgradient method framework, which lays the foundation of our resource allocation algorithm.

3.4.1 Algorithm Description

The *primal problem* is the following:

$$\min \qquad f(\vec{x}), \qquad \text{s.t. } g(\vec{x}) \preceq 0, \qquad \text{over: } \vec{x} \in \vec{X}, \tag{3.14}$$

where $\vec{X} \in \mathbb{R}^N$ is a nonempty compact N-dimensional convex set, $f(\cdot):\mathbb{R}^N \mapsto \mathbb{R}$ is a convex function, $g(\cdot) = (g_1(\cdot), g_2(\cdot), ..., g_\rho(\cdot))^T$, and each $g_j(\cdot):\mathbb{R}^N \mapsto \mathbb{R}$ is a convex function. Note that $(\cdot)^T$ denotes the transpose of $(\cdot)$.

The *dual problem* is the following:

$$\max \quad q(\vec{\delta}) = \inf_{\vec{x}\in\vec{X}} \big(L(\vec{x},\vec{\delta})\big) \quad \text{s.t.} \quad \vec{\delta} \succeq 0, \qquad \text{over: } \vec{\delta} \in \mathbb{R}^{\rho},$$

where $L(\vec{x},\vec{\delta}) = f(\vec{x}) + \vec{\delta}\cdot g(\vec{x})$ is the Lagrangian of the primal problem, and $\vec{\delta}$ is the vector of the Lagrange multipliers.

Our α-ADSA is shown in Algorithm 1. It is an iterative algorithm. Given an approximation/optimal algorithm for solving the subproblem at each iteration, this framework leads to a result that can provide the following bounds at each iteration: (a) the bounds on the Lagrangian multipliers; (b) the bound on the amount of feasibility violation of the generated primal solutions; (c) the upper and lower bounds on the gap between the optimal solution and the generated primal solutions.

Algorithm 1 α-Approximation Dual Subgradient Algorithm

Output: a sequence of vectors $\hat{x}^{(i)}$, $i = 1, ...$

1: $i \leftarrow 0$
2: **while** the stopping criteria have not been reached **do**
3: $\quad i \leftarrow i+1$, $\hat{x}^{(i)} = \frac{1}{i}\sum_{m=0}^{i-1} \vec{x}^{(m)}$
4: $\quad$ update Lagrange multiplier: $\vec{\delta}^{(i)}=[\vec{\delta}^{(i-1)} + \eta g(\vec{x}^{(i-1)})]^{+}$
5: $\quad$ *try* to minimize $L(\vec{x},\vec{\delta}^{(i)})$ over $\vec{x}\in\vec{X}$: compute $\vec{x}^{(i)}$ such that $L(\vec{x}^{(i)},\vec{\delta}^{(i)}) \leq \alpha \cdot \inf_{\vec{x}\in\vec{X}} L(\vec{x},\vec{\delta}^{(i)}) = \alpha q(\vec{\delta}^{(i)})$
6: **end while**

In iteration 1, from a starting point $(\vec{x}^{(0)},\vec{\delta}^{(0)})$, we compute $\vec{\delta}^{(1)}$ (Line 4). $[\cdot]^{+}$ denotes the projection to $[0,+\infty]$ (*i.e.*, $[\vec{x}\,]^{+} = (\max(0,x_1), ..., \max(0,x_N))^{T}$, where $x_1, ..., x_N$ are elements of vector $\vec{x}$). η is a constant step size. Based on $\vec{\delta}^{(1)}$, we compute $\vec{x}^{(1)}$ (Line 5). In iteration 2, we compute $\vec{\delta}^{(2)}$ based on $\vec{\delta}^{(1)}$ and $\vec{x}^{(1)}$, and then $\vec{x}^{(2)}$ based on $\vec{\delta}^{(2)}$. This procedure continues until the stopping criteria are reached. The primal solution $\hat{x}^{(i)}$ in iteration i is generated by averaging (Line 3). The difference between Algorithm 1 and the subgradient method

proposed by Nedić and Ozdaglar [78] is Line 5. Their scheme tries to find $\vec{x}^{(i)}$ such that $L(\vec{x}^{(i)}, \vec{\delta}^{(i)})=\inf_{\vec{x}\in\vec{X}} L(\vec{x}, \vec{\delta}^{(i)})=q(\vec{\delta}^{(i)})$. However, sometimes it is impossible to achieve this efficiently. We allow an α-approximation to be integrated into subgradient methods, as shown in Line 5. Note that α could be either in $[1, +\infty)$ or $(0, 1)$, since $q(\vec{\delta}^{(i)})$ could be either nonnegative or negative.

3.4.2 Algorithm Analysis

Theorem 3 shows the properties of Algorithm 1.

Theorem 3. *If 1) the Euclidean norms of the subgradients are bounded by a constant $\mathcal{L}$ (i.e. $\mathcal{L} = \max_{\vec{x}\in\vec{X}} ||g(\vec{x})|| < \infty$) and 2) there exists a vector $\vec{x^s}$ that satisfies Slater's condition [7], then the following properties hold for all $i \geq 1$:*

1. *$||\vec{\delta}^{(i)}||$ is bounded by a constant $\varrho = \max\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q(\vec{\delta}^{(0)})}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L}\} + \frac{2}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)}))$, where $\varpi = \min_{j\in[1,\rho]} (-g_j(\vec{x^s}))$. Note that $\varpi > 0$, as $\vec{x^s}$ satisfies Slater's condition [7].*

2. *The amount of constraint violation $||g(\hat{x}^{(i)})^+||$ is bounded by $\frac{\varrho}{i\eta}$.*

3. *$f(\hat{x}^{(i)})$ is upper bounded by $\alpha f^* + \frac{||\vec{\delta}^{(0)}||^2}{2i\eta} + \frac{\eta\mathcal{L}^2}{2}$, where f^* is the optimal value of the primal objective function.*

4. *$f(\hat{x}^{(i)})$ is lower bounded by $f^* - \frac{\varrho^2}{i\eta}$.* □

Although α-ADSA is simple, the proof is quite involved.

Interpretation. 1) states that the Euclidean norm of the vector of the Lagrange multipliers is bounded during the entire iteration process. 2) states that the amount of feasibility violation of the primal solution so generated diminishes to zero at the rate $\frac{1}{i}$ as the number of iterations i increases. 3) and 4) imply that the value of the primal solution so generated will enter an area $[f^*, \alpha f^* + \frac{\eta\mathcal{L}^2}{2}]$ at the

rate $\frac{1}{i}$. Note that when $\alpha = 1$, our algorithm becomes the algorithm presented in Section 4 of [78], which converges to f^* within error level $\frac{\eta\mathcal{L}^2}{2}$ at the rate $\frac{1}{i}$. We can adjust η to obtain an arbitrarily small error at the cost of slowing down the convergence rate.

Before proving Theorem 3, we need to give the following two lemmas.

Lemma 2. *Consider that Slater's condition holds. Let $\vec{\delta}^S$ be a vector such that the set $\mathcal{S} = \{\vec{\delta} \geq 0 | q(\vec{\delta}) \geq q(\vec{\delta}^S)\}$ is not empty. Then we have $||\vec{\delta}|| \leq \frac{1}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^S)), \forall \vec{\delta} \in \mathcal{S}$, where $\varpi = \min_{j\in[1,\rho]}(-g_j(\vec{x^s}))$ and $\vec{x^s}$ is a Slater's vector.* □

This lemma is Lemma 1 in [78] but using our notations.

Lemma 3. *For each $i \geq 0$, $||\vec{\delta}^{(i+1)} - \vec{\delta}||^2 \leq ||\vec{\delta}^{(i)} - \vec{\delta}||^2 + \eta^2||g(\vec{x}^{(i)})||^2 + 2\eta(\alpha q(\vec{\delta}^{(i)}) - q(\vec{\delta}))$.* □

Proof. First, we have

$$\begin{aligned} ||\vec{\delta}^{(i+1)} - \vec{\delta}||^2 &= ||[\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)})]^+ - \vec{\delta}||^2 \leq ||\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)}) - \vec{\delta}||^2 \\ &= ||\vec{\delta}^{(i)} - \vec{\delta}||^2 + 2\eta g(\vec{x}^{(i)}) \cdot (\vec{\delta}^{(i)} - \vec{\delta}) + \eta^2||g(\vec{x}^{(i)})||^2. \end{aligned} \tag{3.15}$$

Second, by Line 5 in Algorithm 1, we have $f(\vec{x}^{(i)}) + \vec{\delta}^{(i)} \cdot g(\vec{x}^{(i)}) \leq \alpha q(\vec{\delta}^{(i)})$, and by the definition of $q(\vec{\delta})$, we know $f(\vec{x}^{(i)}) + \vec{\delta} \cdot g(\vec{x}^{(i)}) \geq q(\vec{\delta})$. These two inequalities imply that

$$\begin{aligned} \alpha q(\vec{\delta}^{(i)}) &\geq f(\vec{x}^{(i)}) + \vec{\delta}^{(i)} \cdot g(\vec{x}^{(i)}) + \vec{\delta} \cdot g(\vec{x}^{(i)}) - \vec{\delta} \cdot g(\vec{x}^{(i)}) \\ &\geq q(\vec{\delta}) + \vec{\delta}^{(i)} \cdot g(\vec{x}^{(i)}) - \vec{\delta} \cdot g(\vec{x}^{(i)}). \end{aligned}$$

Thus, we have $g(\vec{x}^{(i)}) \cdot (\vec{\delta}^{(i)} - \vec{\delta}) \leq \alpha q(\vec{\delta}^{(i)}) - q(\vec{\delta})$. Substituting this inequality into (3.15), we therefore prove this lemma. ■

Proof of Theorem 3.

1) Under Slater's condition the optimal dual set is $\mathcal{D}^*$ is not empty. Consider the set $\mathcal{S}$ defined by $\mathcal{S} = \{\vec{\delta} \geq 0 | q(\vec{\delta}) \geq (\frac{q^*}{\alpha} - \frac{\eta\mathcal{L}^2}{2\alpha})\}$ which is nonempty in view of $\mathcal{D}^* \subset \mathcal{S}$, where q^* is the optimal value of the dual problem. We fix an arbitrary $\vec{\delta}^* \in \mathcal{D}^*$ and we first prove that for all $i \geq 0$,

$$||\vec{\delta}^{(i)} - \vec{\delta}^*|| \leq \max\{\frac{1}{\varpi}(f(\vec{x^s}) - \frac{q^*}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + ||\vec{\delta}^*|| + \eta\mathcal{L}, ||\vec{\delta}^{(0)} - \vec{\delta}^*||\}. \quad (3.16)$$

Recall that $\varpi = \min_{j\in[1,\rho]}(-g_j(\vec{x^s}))$. We will prove (3.16) by induction on i. It obviously holds for $i = 0$. Assume that it holds for some $i \geq 0$. We will prove that it also holds for $i + 1$. We consider two cases.

Case 1: $q(\vec{\delta}^{(i)}) \geq \frac{q^*}{\alpha} - \frac{\eta\mathcal{L}^2}{2\alpha}$. We have

$$\begin{aligned} ||\vec{\delta}^{(i+1)} - \vec{\delta}^*|| &= ||[\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)})]^+ - \vec{\delta}^*|| \leq ||\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)}) - \vec{\delta}^*|| \\ &\leq ||\vec{\delta}^{(i)}|| + ||\eta g(\vec{x}^{(i)})|| + ||\vec{\delta}^*|| \leq ||\vec{\delta}^{(i)}|| + \eta\mathcal{L} + ||\vec{\delta}^*||. \end{aligned}$$

Since $q(\vec{\delta}^{(i)}) \geq \frac{q^*}{\alpha} - \frac{\eta\mathcal{L}^2}{2\alpha}$, it follows that $\vec{\delta}^{(i)} \in \mathcal{S}$. By Lemma 2, $||\vec{\delta}^{(i)}|| \leq \frac{1}{\varpi}(f(\vec{x^s}) - (\frac{q^*}{\alpha} - \frac{\eta\mathcal{L}^2}{2\alpha}))$. We hence have $||\vec{\delta}^{(i+1)} - \vec{\delta}^*|| \leq \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q^*}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L} + ||\vec{\delta}^*||$.

Case 2: $q(\vec{\delta}^{(i)}) < \frac{q^*}{\alpha} - \frac{\eta\mathcal{L}^2}{2\alpha}$. By using Lemma 3 with $\vec{\delta} = \vec{\delta}^* \in \mathcal{D}^*$ and $||g(\vec{x}^{(i)})||^2 \leq \mathcal{L}^2$, we have

$$\begin{aligned} ||\vec{\delta}^{(i+1)} - \vec{\delta}^*||^2 &\leq ||\vec{\delta}^{(i)} - \vec{\delta}^*||^2 + \eta^2||g(\vec{x}^{(i)})||^2 + 2\eta(\alpha q(\vec{\delta}^{(i)}) - q(\vec{\delta}^*)) \\ &\leq ||\vec{\delta}^{(i)} - \vec{\delta}^*||^2 + 2\eta(\alpha q(\vec{\delta}^{(i)}) + \frac{\eta\mathcal{L}^2}{2} - q^*) \leq ||\vec{\delta}^{(i)} - \vec{\delta}^*||^2. \end{aligned}$$

Combining these two cases above, we have proved that (3.16) holds for all $i \geq 0$. Thus, we can further have

$$\begin{aligned} &||\vec{\delta}^{(i)}|| \leq ||\vec{\delta}^{(i)} - \vec{\delta}^*|| + ||\vec{\delta}^*|| \\ &\leq \max\left\{||\vec{\delta}^{(0)} - \vec{\delta}^*||, \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q^*}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L} + ||\vec{\delta}^*||\right\} + ||\vec{\delta}^*|| \\ &\leq \max\left\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q^*}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L}\right\} + 2||\vec{\delta}^*||. \end{aligned}$$

Considering $\mathcal{D}^* = \{\vec{\delta} \geq 0 | q(\vec{\delta}) \geq q^*\}$, by Lemma 2, we know $||\vec{\delta}^*|| \leq \frac{1}{\varpi}(f(\vec{x^s}) - q^*)$. Therefore, for $i \geq 0$ we have

$$\begin{aligned} ||\vec{\delta}^{(i)}|| \leq & \max\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q^*}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L}\} + \frac{2}{\varpi}(f(\vec{x^s}) - q^*) \\ \leq & \max\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q(\vec{\delta}^{(0)})}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L}\} + \frac{2}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)})). \end{aligned} \quad (3.17)$$

We set $\varrho = \max\{||\vec{\delta}^{(0)}||, \frac{1}{\varpi}(f(\vec{x^s}) - \frac{q(\vec{\delta}^{(0)})}{\alpha} + \frac{\eta\mathcal{L}^2}{2\alpha}) + \eta\mathcal{L}\} + \frac{2}{\varpi}(f(\vec{x^s}) - q(\vec{\delta}^{(0)}))$.

2) By Line 4 in Algorithm 1, we have $\vec{\delta}^{(i+1)} = [\vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)})]^+ \geq \vec{\delta}^{(i)} + \eta g(\vec{x}^{(i)})$ for $i \geq 0$. We hence have $\eta g(\vec{x}^{(i)}) \leq \vec{\delta}^{(i+1)} - \vec{\delta}^{(i)}$ for $i \geq 0$. Summing them up over until $i-1$, we have $\eta \sum_{m=0}^{i-1} g(\vec{x}^{(m)}) \leq \vec{\delta}^{(i)} - \vec{\delta}^{(0)} \leq \vec{\delta}^{(i)}$ for $i \geq 1$. Since each of $g_j(\cdot)$ is convex, we have for $i \geq 1$, $g(\hat{x}^{(i)}) = g(\frac{1}{i}\sum_{m=0}^{i-1} \vec{x}^{(m)}) \leq \frac{1}{i}\sum_{m=0}^{i-1} g(\vec{x}^{(m)}) \leq \frac{\vec{\delta}^{(i)}}{\eta i}$. Considering $\vec{\delta}^{(i)} \geq 0$, we know $g(\hat{x}^{(i)})^+ \leq \frac{\vec{\delta}^{(i)}}{\eta i}$, for all $i \geq 1$. By (3.17), we hence have $||g(\hat{x}^{(i)})^+|| \leq \frac{||\vec{\delta}^{(i)}||}{\eta i} \leq \frac{\varrho}{i\eta}$, for $i \geq 1$.

3) Since $f(\cdot)$ is convex, we know

$$f(\hat{x}^{(i)}) \leq \frac{1}{i}\sum_{m=0}^{i-1} f(\vec{x}^{(m)}) = \frac{1}{i}\sum_{m=0}^{i-1}\Big(f(\vec{x}^{(m)}) + \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)})\Big) - \frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}). \quad (3.18)$$

Substituting $f(\vec{x}^{(m)}) + \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) \leq \alpha q(\vec{\delta}^{(m)})$ (by Line 5 in Algorithm 1) into (3.18), we have

$$\begin{aligned} f(\hat{x}^{(i)}) &\leq \frac{1}{i}\sum_{m=0}^{i-1} \alpha q(\vec{\delta}^{(m)}) - \frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}). \\ &\leq \alpha q^* - \frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}). \end{aligned} \quad (3.19)$$

Additionally, for $m \geq 0$ we know

$$\begin{aligned} ||\vec{\delta}^{(m+1)}||^2 = & ||[\vec{\delta}^{(m)} + \eta g(\vec{x}^{(m)})]^+||^2 \leq ||\vec{\delta}^{(m)} + \eta g(\vec{x}^{(m)})||^2 \\ = & ||\vec{\delta}^{(m)}||^2 + ||\eta g(\vec{x}^{(m)})||^2 + 2\eta\vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}). \end{aligned}$$

It implies $-\vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) \leq \frac{||\vec{\delta}^{(m)}||^2 + ||\eta g(\vec{x}^{(m)})||^2 - ||\vec{\delta}^{(m+1)}||^2}{2\eta}$. Summing this inequality over $m \in [0, i-1]$ for $i \geq 1$, we have $-\frac{1}{i}\sum_{m=0}^{i-1} \vec{\delta}^{(m)} \cdot g(\vec{x}^{(m)}) \leq \frac{||\vec{\delta}^{(0)}||^2 - ||\vec{\delta}^{(i)}||^2}{2i\eta} + \frac{\eta\sum_{m=0}^{i-1} ||g(\vec{x}^{(m)})||^2}{2i}$. Substituting this into (3.19), we have

$$f(\hat{x}^{(i)}) \leq \alpha q^* + \frac{||\vec{\delta}^{(0)}||^2 - ||\vec{\delta}^{(i)}||^2}{2i\eta} + \frac{\eta \sum_{m=0}^{i-1} ||g(\vec{x}^{(m)})||^2}{2i}$$
$$\leq \alpha q^* + \frac{||\vec{\delta}^{(0)}||^2}{2i\eta} + \frac{\eta \mathcal{L}^2}{2} = \alpha f^* + \frac{||\vec{\delta}^{(0)}||^2}{2i\eta} + \frac{\eta \mathcal{L}^2}{2}. \quad (3.20)$$

Note that (3.20) holds because of Slater's condition.

4) Given a dual optimal solution $\vec{\delta}^*$, we have

$$f(\hat{x}^{(i)}) = f(\hat{x}^{(i)}) + \vec{\delta}^* \cdot g(\hat{x}^{(i)}) - \vec{\delta}^* \cdot g(\hat{x}^{(i)}) \geq q(\vec{\delta}^*) - \vec{\delta}^* \cdot g(\hat{x}^{(i)}).$$

Additionally, since $\vec{\delta}^* \geq 0$ and $g(\hat{x}^{(i)})^+ \geq g(\hat{x}^{(i)})$, we have

$$-\vec{\delta}^* \cdot g(\hat{x}^{(i)}) \geq -\vec{\delta}^* \cdot g(\hat{x}^{(i)})^+ \geq -||\vec{\delta}^*|| ||g(\hat{x}^{(i)})^+||.$$

Combining the two inequalities above, we have $f(\hat{x}^{(i)}) \geq q(\vec{\delta}^*) - ||\vec{\delta}^*|| ||g(\hat{x}^{(i)})^+|| \geq q^* - \frac{\varrho^2}{i\eta} = f^* - \frac{\varrho^2}{i\eta}$. ■

3.5 Distributed Resource Allocation Algorithm

In this subsection, we present a distributed iterative algorithm for solving our resource allocation problem based on Algorithm 1.

3.5.1 Algorithm Description

We divide the timeline into a sequence of periods i=0, 1,$\cdots$. In each period, our algorithm iterates once. By Algorithm 1, in iteration period $i \geq 1$, we need to update Lagrange multipliers $\vec{\delta}^{(i)}$ (Line 4) and then compute $\vec{x}^{(i)}$ (Line 5). The details of these two operations are described in the following.

3.5.1.1 Lagrange Multiplier Update

By Line 4 of Algorithm 1, in iteration period i, we update the Lagrange multipliers $\vec{\delta}^{(i)}=((\vec{\alpha}^{(i)})^T,(\vec{\beta}^{(i)})^T,(\vec{\psi}^{(i)})^T,(\vec{\zeta}^{(i)})^T,(\vec{\theta}^{(i)})^T)^T$ as follows:

$$\alpha_k^{(i)}(u)=\Big[\alpha_k^{(i-1)}(u)+\eta\left(h_k^{(i-1)}(u)-R_k^{(i-1)}(u)+\overline{R}_k^{(i-1)}(u)\right)\Big]^+,$$
$$\beta_k^{(i)}=\left[\beta_k^{(i-1)}+\eta\left(\lambda_k^{(i-1)}-\mathbb{Q}_k\right)\right]^+,$$
$$\psi^{(i)}(u)=\left[\psi^{(i-1)}(u)+\eta\left(R^{(i-1)}(u)-\mathbb{M}(u)\right)\right]^+,$$
$$\zeta^{(i)}(u,w)=\left[\zeta^{(i-1)}(u,w)+\eta\big(R^{(i-1)}(u)-R^{(i-1)}(w)-\mathbb{B}(u,w)\big)\right]^+,$$
$$\theta_k^{(i)}(u)=\left[\theta_k^{(i-1)}(u)+\eta\left(R_k^{(i-1)}(u)-\mathbb{N}_k(u)\right)\right]^+.$$

We now analyze what information is needed to update the Lagrange multipliers. Each node u can individually update $\alpha_k^{(i)}(u)$, $\psi^{(i)}(u)$, and $\theta_k^{(i)}(u)$. Each user k can individually update $\beta_k^{(i)}$. In order to update $\zeta^{(i)}(u,w)$, each node u needs to know $R^{(i-1)}(w)$ of node w located in u's load balance area. In practice, a node can consider the set of its 1 or 2-hop neighbors as its load balance area. In summary, updating Lagrange multipliers can be done in a distributed manner.

3.5.1.2 User Rate Control and Node Rate Control

Let us consider the Lagrangian of our primal problem:

$$L(\vec{x},\vec{\delta}\,)=L\left((\vec{\lambda}^{\,T},\vec{r}^{\,T})^T,(\vec{\alpha}^{\,T},\vec{\beta}^{\,T},\vec{\psi}^{\,T},\vec{\zeta}^{\,T},\vec{\theta}^{\,T})^T\right)$$
$$=-\sum_{k=1}^{K}U_k\left(\lambda_k\right)+\sum_{k=1}^{K}\sum_{u\in\mathcal{V}_k}\alpha_k(u)\big(h_k(u)-R_k(u)+\overline{R}_k(u)\big)$$
$$+\sum_{k=1}^{K}\beta_k\left(\lambda_k-\mathbb{Q}_k\right)+\sum_{u\in\mathcal{V}}\psi(u)(R(u)-\mathbb{M}(u))$$
$$+\sum_{u\in\mathcal{V}}\sum_{w\in\mathcal{A}(u)}\zeta(u,w)\big(R(u)-R(w)-\mathbb{B}(u,w)\big)+\sum_{k=1}^{K}\sum_{u\in\mathcal{V}_k}\theta_k(u)\big(R_k(u)-\mathbb{N}_k(u)\big).$$

By Algorithm 1, in iteration period i, we need to find $\vec{x}^{(i)}$ to minimize the Lagrangian $L(\vec{x}, \vec{\delta}^{(i)})$. After some mathematical manipulations, minimizing the above Lagrangian can be decomposed to solving the following two problems.

User k Rate Control Problem:

$$\min \quad -U_k\left(\lambda_k^{(i)}\right) + (\alpha_k^{(i)}(s_k) + \beta_k^{(i)} - \alpha_k^{(i)}(d_k))\lambda_k^{(i)}, \tag{3.21}$$

and *Node Rate Control Problem*:

$$\begin{aligned}\min \quad & \sum_{k=1}^{K}\sum_{u\in\mathcal{V}_k}\sum_{(u,v)\in E_k} r_k^{(i)}(u,v)\Big(-\alpha_k^{(i)}(u) + \alpha_k^{(i)}(v) + \psi^{(i)}(u) \\ & + \sum_{w\in\mathcal{A}(u)}(\zeta^{(i)}(u,w) - \zeta^{(i)}(w,u)) + \theta_k^{(i)}(u)\Big) \text{ over } \vec{r}^{(i)} \in \prod. \end{aligned} \tag{3.22}$$

Problem (3.21) can be easily solved by each user k. Hence, we next focus on problem (3.22). Problem (3.22) is equivalent to

$$\begin{aligned}\min \quad & -\sum_{(u,v)\in E} r^{(i)}(u,v) \max_{k\in[1,K]} \Big(\alpha_k^{(i)}(u) - \alpha_k^{(i)}(v) - \psi^{(i)}(u) \\ & - \sum_{w\in\mathcal{A}(u)}(\zeta^{(i)}(u,w) - \zeta^{(i)}(w,u)) - \theta_k^{(i)}(u)\Big) \text{ over } \vec{r}^{(i)} \in \prod. \end{aligned} \tag{3.23}$$

Note that if $(u,v) \notin E_k$, we remove the corresponding variables in (3.23) or set them to 0. Let $w_{u,v}(i)$ denote $\max_{k\in[1,K]}(\alpha_k^{(i)}(u) - \alpha_k^{(i)}(v) - \psi^{(i)}(u) - \sum_{w\in\mathcal{A}(u)}(\zeta^{(i)}(u,w) - \zeta^{(i)}(w,u)) - \theta_k^{(i)}(u))$, and k^* denote the corresponding k to achieve this maximum value. Hence, (3.23) is rewritten as

$$\min S^{(i)}(\vec{r}^{(i)}) = -\sum_{(u,v)\in E} r^{(i)}(u,v) w_{u,v}(i) \text{ over } \vec{r}^{(i)} \in \prod. \tag{3.24}$$

Recall that the feasible region $\prod$ is the convex hull of the rate vectors constructed based on the independent sets in the conflict graph C_G. This implies that solving problem (3.24) is equivalent to finding a *maximum weighted independent set* (MWIS) over C_G, where the vertex corresponding to the link (u,v) in $G(\mathcal{V}, E)$ has a weight $r^{(i)}(u,v)w_{u,v}(i)$. However, finding an MWIS for a general graph is

NP-hard [4]. In the following, we will describe how to solve (3.24) for PIM and GIM in a distributed manner, respectively.

PIM: Recall that in PIM there is a conflict between links which share a common node. An MWIS in C_G must correspond to a maximum weighted matching (MWM) in $G(\mathcal{V}, E)$, since a matching is a set of links without common nodes. Hence, solving (3.24) is equivalent to finding an MWM in $G(\mathcal{V}, E)$, where the weight of link (u, v) is $r^{(i)}(u, v)w_{u,v}(i)$. There exist polynomial-time *distributed* approximation algorithms for this problem. We can use the *distributed scheduling algorithm* in [13], which is based on [40], to solve this MWM problem. It finds a matching in $O(E)$ time, whose weight is at least $1/2$ of that of the MWM. *Hence, in iteration period i, node u transmits information for user k^* on link (u, v) at the rate of $\mathbb{C}(u, v)$ if link (u, v) is in the resulting matching, and 0 otherwise.* This hence ensures that $S^{(i)}(\vec{r}^{(i)}) \leq 0.5S^{*(i)} \leq 0$, where $S^{*(i)}$ is the optimal value of (3.24) in PIM.

GIM: Different from PIM, for GIM we have to find an MWIS over C_G. This is an NP-hard problem for which even approximating in polynomial time is NP-hard [4]. Moreover, finding an MWIS is very difficult to efficiently decentralize in a general graph [60]. Thus, for the practical purpose we adopt the *maximal scheduling* rather than *maximum scheduling*. More specifically, A *maximal independent set* is an independent set that is not properly contained in any other independent set. We want to find a maximal independent set by choosing the nodes of the set so to maximize the total weight (maximal weighted independent set problem [4]). Note that the MWIS is the independent set with the biggest weight among all the independent sets. Basagni [4] proposed a distributed algorithm for the efficient determination of a maximal weighted independent set. This algorithm is executed at each vertex in C_G (i.e. each link in $G(\mathcal{V}, E)$) with the

local topology knowledge required. Its time complexity is proven to be bounded by the stability number of C_G. Our extensive experiments show that it typically achieves a weight which is at least 1/2 of that of the MWIS. *Hence in iteration period i, node u transmits information for user k^* on link (u, v) at the rate of $\mathbb{C}(u, v)$ if link (u, v) is in the resulting independent set, and 0 otherwise.* This ensures that $S^{(i)}(\vec{r}^{(i)}) \leq 0.5S^{*(i)} \leq 0$ in practice, where $S^{*(i)}$ is the optimal value of (3.24) in GIM.

Thus far, we have $\vec{x}^{(i)}=((\vec{\lambda}^{(i)})^T, (\vec{r}^{(i)})^T)^T$. By Line 3 of Algorithm 1, we need to generate the primal solution $\hat{x}^{(i)}$ by computing $\frac{1}{i}\sum_{m=0}^{i-1}\vec{x}^{(m)}$. However, we do not need to explicitly generate $\hat{x}^{(i)}$, because our algorithm has *automatically* generated $\hat{x}^{(i)}$ in the following sense: For the first i iteration periods, following $\vec{x}^{(0)}$, ..., $\vec{x}^{(i-1)}$ is equivalent to following $\hat{x}^{(i)}$ in every iteration period, as $i\hat{x}^{(i)} = \sum_{m=0}^{i-1}\vec{x}^{(m)}$.

3.5.2 Algorithm Analysis

3.5.2.1 Optimality Analysis

We first show that a bound on the norm of subgradients exists and Slater's condition holds. Since our programming system is defined on a compact set, by [78] we know that there exists a constant $\mathcal{L}<\infty$ such that $\mathcal{L}=\max_{\vec{x}\in\vec{X}}||g(\vec{x})||$. Slater's condition also holds, since we can easily construct a strictly feasible solution by assigning each user a small amount of information flow, which goes through all the nodes involved in its session, so that constraints (3.9-3.13) hold. Additionally, we need to ensure that at each involved node the rate of outgoing flow is a slightly greater than its incoming rate so that (3.8) is strictly feasible.

We rewrite our objective programming system (i.e. (3.7)-(3.13)) to the standard form (3.14) for brevity. Since the norm of subgradients is bounded and

Slater's condition holds, *the properties shown in Theorem 3 hold for our resource allocation algorithm.* Note that for PIM, α in Theorem 3 is 0.5. This is because we decompose the minimization of Lagrangian into two problems (3.21), which we solve optimally, and (3.24), for which we have $S^{(i)}(\vec{r}^{(i)}) \leq 0.5 S^{*(i)} \leq 0$. Hence, we have $L(\vec{x}^{(i)}, \vec{\delta}^{(i)}) \leq 0.5 \inf_{\vec{x} \in \vec{X}} L(\vec{x}, \vec{\delta}^{(i)})$. For a similar reason, the typical value of α for GIM is 0.5 in practice.

3.5.2.2 Dual Space Information Analysis

In this subsection, we analyze the dual space information and its implication on the algorithm performance.

Network: Queue Length and System Stability Suppose that the length of each iteration period is τ. Let $\mathcal{Q}_k^{(i)}(u)$ denote the length of the queue for user k at node u at the beginning of iteration period i. $\mathcal{Q}_k^{(i)}(u)$ is computed by:

$$\mathcal{Q}_k^{(i)}(u) = \left[\mathcal{Q}_k^{(i-1)}(u) + \tau \left(h_k^{(i-1)}(u) - R_k^{(i-1)}(u) + \overline{R}_k^{(i-1)}(u) \right) \right]^+,$$

where the initial queue length is $\mathcal{Q}_k^{(0)}(u)$. The following theorem shows a bound on the queue length.

Theorem 4. *For any* $i \geq 0$

$$\mathcal{Q}_k^{(i)}(u) \leq \frac{\tau}{\eta} \alpha_k^{(i)}(u) + \left[\mathcal{Q}_k^{(0)}(u) - \frac{\tau}{\eta} \alpha_k^{(0)}(u) \right]^+ . \square \tag{3.25}$$

Proof. We prove (3.25) by mathematical induction. Inequality (3.25) is obviously true for $i = 0$. Suppose that it holds for some $i \geq 0$. Now we prove that this also holds for $i + 1$.

$$\begin{aligned} &\mathcal{Q}_k^{(i+1)}(u) = \left[\mathcal{Q}_k^{(i)}(u) + \tau \left(h_k^{(i)}(u) - R_k^{(i)}(u) + \overline{R}_k^{(i)}(u) \right) \right]^+ \\ &\leq \frac{\tau}{\eta} \left[\alpha_k^{(i)}(u) + \eta \left(h_k^{(i)}(u) - R_k^{(i)}(u) + \overline{R}_k^{(i)}(u) \right) \right]^+ \\ &+ \left[\mathcal{Q}_k^{(0)}(u) - \frac{\tau}{\eta} \alpha_k^{(0)}(u) \right]^+ \leq \frac{\tau}{\eta} \alpha_k^{(i+1)}(u) + \left[\mathcal{Q}_k^{(0)}(u) - \frac{\tau}{\eta} \alpha_k^{(0)}(u) \right]^+ . \blacksquare \end{aligned}$$

Interpretation of Theorem 4. Theorem 4 shows a bound on the queue length in each iteration period. By Theorem 3.1, we know that $||\vec{\delta}^{(i)}|| \leq \varrho$, which further implies $\alpha_k^{(i)}(u) \leq \varrho$. In addition, we know $\left[\mathcal{Q}_k^{(0)}(u) - \frac{\tau}{\eta}\alpha_k^{(0)}(u)\right]^+ \leq \mathcal{Q}_k^{(0)}(u)$. Therefore, *as long as the buffer at node u for user k is greater than* $\frac{\tau}{\eta}\varrho + \mathcal{Q}_k^{(0)}(u)$*, no overflow will happen (note that both* $\frac{\tau}{\eta}\varrho$ *and* $\mathcal{Q}_k^{(0)}(u)$ *are constants).* This lemma further implies that the system is stable, since the queue length is bounded and does not blow up to infinity at any time.

User: User Utility Deficit Recall Theorem 3.2, which quantifies the feasibility violation. This property implies that $\alpha-$ADSA, the underlying framework of our resource allocation algorithm, cannot always ensure the constraint feasibility. Hence, we next analyze, by using dual space information obtained from our resource allocation algorithm, at most how much more utility a user k could obtain (called *user utility deficit*), compared with the maximum allowed utility $U_k(\mathbb{Q}_k)$.

We use $\Delta_k^{(i)} = \sum_{m=0}^{i-1} \tau(\lambda_k^{(m)} - \mathbb{Q}_k)$ to represent the gap between the total amount of transmitted information and the maximum allowed amount from iteration 0 to iteration period $i-1$. Hence, the user utility deficit $\mathcal{D}_k^{(i)}$ from iteration period 0 to iteration period $i-1$ is defined by $U_k\left(\frac{i\tau\mathbb{Q}_k + [\Delta_k^{(i)}]^+}{i\tau}\right) - U_k(\mathbb{Q}_k)$. The following theorem shows a bound on $\mathcal{D}_k^{(i)}$.

Theorem 5. *For* $i \geq 1$, $\mathcal{D}_k^{(i)} \leq U_k(\frac{\beta_k^{(i)}}{i\eta} + \frac{[\lambda_k^{(0)} - \mathbb{Q}_k]^+}{i})$. □

Proof. Let $A_1 = \tau(\lambda_k^{(0)} - \mathbb{Q}_k)$. We claim that for $i \geq 1$, $\Delta_k^{(i)} \leq \frac{\tau}{\eta}\beta_k^{(i)} + [A_1]^+$, and prove this claim by induction. This claim is obviously true for $i = 1$ since $\beta_k^{(1)} \geq 0$. Suppose that it holds for some $i \geq 1$. Now we prove that the claim also holds for $i+1$. By the definition of $\Delta_k^{(i)}$, we have

$$\Delta_k^{(i+1)} = \Delta_k^{(i)} + \tau(\lambda_k^{(i)} - \mathbb{Q}_k) \leq \frac{\tau}{\eta}\beta_k^{(i)} + [A_1]^+ + \tau(\lambda_k^{(i)} - \mathbb{Q}_k)$$
$$\leq \frac{\tau}{\eta}[\beta_k^{(i)} + \eta(\lambda_k^{(i)} - \mathbb{Q}_k)]^+ + [A_1]^+ = \frac{\tau}{\eta}\beta_k^{(i+1)} + [A_1]^+.$$

Thus, the claim is proved. We now can bound the user utility deficit using this claim as follows.

$$\mathcal{D}_k^{(i)} = U_k\left(\mathbb{Q}_k + \left[\frac{\Delta_k^{(i)}}{i\tau}\right]^+\right) - U_k\left(\mathbb{Q}_k\right) \leq U_k\left(\left[\frac{\Delta_k^{(i)}}{i\tau}\right]^+\right) \tag{3.26}$$
$$= U_k\left(\frac{\beta_k^{(i)}}{i\eta} + \frac{[A_1]^+}{i\tau}\right).$$

Inequality (3.26) holds since $U_k(\cdot)$ is an increasing and concave function, and $U_k(0) = 0$. ■

Interpretation of Theorem 5. Theorem 5 shows a bound on the user utility deficit in each iteration period using dual variable $\beta_k^{(i)}$. Since $\beta_k^{(i)}$ is bounded by a constant ϱ according to Theorem 3.1 and $[\lambda_k^{(0)} - \mathbb{Q}_k]^+$ is a constant, *Theorem 5 implies that the user utility deficit is decreasing to 0 at the rate* $\frac{1}{i}$ *as our algorithm iterates.* When i is sufficiently large, our algorithm can guarantee nearly zero user utility deficit.

Node: Load Violation Like the user utility deficit, we next analyze the node load constraint violation from the following three aspects: node-user load violation, node max load violation, and node load balance violation.

For node u and user k, we define the *node-user load violation ratio* by $\tilde{\mathcal{N}}_k^{(i)}(u) = \left[\frac{\sum_{m=0}^{i-1} R_k^{(m)}(u)}{i\mathbb{N}_k(u)} - 1\right]^+$. Note that $\frac{\sum_{m=0}^{i-1} R_k^{(m)}(u)}{i\mathbb{N}_k(u)}$ represents the ratio of the total load carried by node u for user k until the beginning of iteration period i over the maximum allowed load. We then subtract 1 from this ratio and take $[\cdot]^+$ operation, because only when $\sum_{m=0}^{i-1} R_k^{(m)}(u) > i\mathbb{N}_k(u)$, the node-user load constraint is violated. Similarly, we can define the *node max load violation ratio*

by $\tilde{\mathcal{M}}^{(i)}(u) = \left[\frac{\sum_{m=0}^{i-1} R^{(m)}(u)}{i\mathbb{M}(u)} - 1\right]^+$, and *the node load balance violation ratio* by $\tilde{\mathcal{B}}^{(i)}(u,w) = \left[\frac{\sum_{m=0}^{i-1}\left(R^{(m)}(u) - R^{(m)}(w)\right)}{i\mathbb{B}(u,w)} - 1\right]^+$.

We have the following theorem to bound these three ratios using dual space information.

Theorem 6. *For any* $i \geq 1$,

$$\tilde{\mathcal{N}}_k^{(i)}(u) \leq \frac{\theta_k^{(i)}(u) + \eta\left[R_k^{(0)}(u) - \mathbb{N}_k(u)\right]^+}{i\eta\mathbb{N}_k(u)};$$
$$\tilde{\mathcal{M}}^{(i)}(u) \leq \frac{\psi^{(i)}(u) + \eta\left[R^{(0)}(u) - \mathbb{M}(u)\right]^+}{i\eta\mathbb{M}(u)};$$
$$\tilde{\mathcal{B}}^{(i)}(u,w) \leq \frac{\zeta^{(i)}(u,w) + \eta\left[R^{(0)}(u) - R^{(0)}(w) - \mathbb{B}(u,w)\right]^+}{i\eta\mathbb{B}(u,w)}.\square$$

Proof. The proofs for these three inequalities are similar. We hence only prove the first inequality.

Let $\mathcal{N}_k'^{(i)}(u) = \sum_{m=0}^{i-1}\tau(R_k^{(m)}(u) - \mathbb{N}_k(u))$, and $A_2 = \eta[R_k^{(0)}(u) - \mathbb{N}_k(u)]^+$. We claim that for $i \geq 1$, $\mathcal{N}_k'^{(i)}(u) \leq \frac{\tau}{\eta}\theta_k^{(i)}(u) + \frac{\tau}{\eta}A_2$. This is obviously true for $i = 1$. Suppose that this claim holds for some $i \geq 1$. We now prove that this claim also holds for $i+1$.

$$\begin{aligned}
&\mathcal{N}_k'^{(i+1)}(u) = \mathcal{N}_k'^{(i)}(u) + \tau\left(R_k^{(i)}(u) - \mathbb{N}_k(u)\right)\\
&\leq \frac{\tau}{\eta}\theta_k^{(i)}(u) + \frac{\tau}{\eta}A_2 + \tau\left(R_k^{(i)}(u) - \mathbb{N}_k(u)\right)\\
&\leq \frac{\tau}{\eta}\left[\theta_k^{(i)}(u) + \eta\left(R_k^{(i)}(u) - \mathbb{N}_k(u)\right)\right]^+ + \frac{\tau}{\eta}A_2 = \frac{\tau}{\eta}\theta_k^{(i+1)}(u) + \frac{\tau}{\eta}A_2.
\end{aligned}$$

The claim is proved, which implies $\frac{\sum_{m=0}^{i-1}R_k^{(m)}(u)}{i\mathbb{N}_k(u)} - 1 \leq \frac{\theta_k^{(i)}(u)+A_2}{i\eta\mathbb{N}_k(u)}$. Considering $\frac{\theta_k^{(i)}(u)+A_2}{i\eta\mathbb{N}_k(u)} \geq 0$, we know $\tilde{\mathcal{N}}_k^{(i)}(u) \leq \frac{\theta_k^{(i)}(u)+A_2}{i\eta\mathbb{N}_k(u)}$. ■

Interpretation of Theorem 6. Theorem 6 shows the bounds on the node-user load violation ratios, node max load violation ratios, and node load balance violation ratios in each iteration period. Let us first consider the node-user load

violation ratios. Since $\theta_k^{(i)}(u)$ is bounded by ϱ according to Theorem 3.1 and $\frac{\left[R_k^{(0)}(u)-\mathbb{N}_k(u)\right]^+}{\mathbb{N}_k(u)}$ is a constant, this theorem indicates that the node-user load violation ratios are approaching zero at the rate $\frac{1}{i}$ as our algorithm iterates. Similarly, this theorem indicates both node max load violation ratios and node load balance violation ratios approaching zero at the rate $\frac{1}{i}$ as our algorithm iterates.

3.6 Performance Evaluation

We simulated a wireless network with 30 nodes and 10 users by randomly distributing nodes in a $1000\times1000m$ region, and by randomly choosing two different nodes as the source and the destination for each user. The node transmission range was set to be 300m. In the simulation, we used a normalized rate. The capacity of each link followed a normal distribution with a mean of 1 and a variance of 0.5. The iterative step size η was set to 0.01. We evaluated the performance of our algorithm in both PIM and GIM. For GIM, we only evaluated RCM because the obtained utility in RCM can be regarded as a lower bound (not tight) on that in PM, since RCM takes into account more interference than PM does.

Fig.3.1a shows the average user utility obtained in each iteration period. We evaluate two constraint settings. In Fig.3.1a, "NC" represents the setting that no load constraint is being used, while "WC" represents the setting that all the $\mathbb{Q}_k$'s, $\mathbb{N}_k(u)$'s, $\mathbb{M}(u)$'s and $\mathbb{B}(u,w)$'s follow a normal distribution with a mean of 0.1 and a variance of 0.05. We used the transmission rate λ_k of the source of user k to compute the utility (specifically, $U_k(\lambda_k) = \ln(1+\lambda_k)$) rather than the rate of the flow received by the destination, in order to observe how our algorithm adjusts the transmission rate of the source. Note that the transmission rate of the source may be far larger than the rate of the information received by the destination before the network stabilizes. At the beginning, the source node

sends out information as much as possible. Hence, we observe that at this moment the obtained utility is large. However, the corresponding rate is so large that the available resources cannot satisfy such rate. Continually using this transmission rate will result in network congestion and instability. As a result, we observe that the obtained utility (and the corresponding outgoing flow rate) is reduced by our algorithm. The utility drops quickly before iteration period 50 and then decreases slowly, approaching a stable point. Hence, our algorithm converges very fast. In addition, we observe that if the load constraints are considered, the utilities in PIM and GIM (i.e. "WC, PIM", "WC, GIM") are almost the same although GIM considers more interference. This is because our load constraints make links unsaturated.

Fig. 3.1b shows the average length of node queues. We normalized the length of iteration periods to 1. We observe that the queue length is increasing but approaching a stable status. This length never exceeds 300 even after a very long time (200,000 iteration periods).

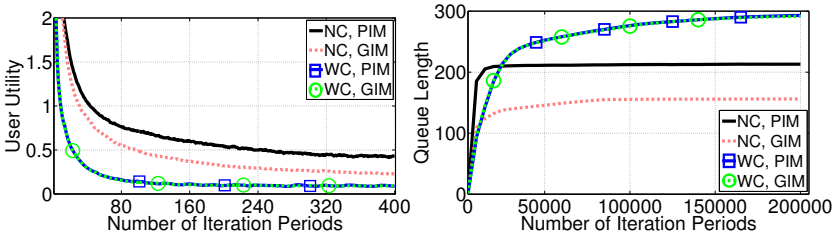
Fig.3.1c-3.1g show user utility deficits, node-user load violation ratios, node max load violation ratios, node load balance violation ratios, and load fairness indices (defined later), respectively. In the simulation for Fig.3.1c, the $\mathbb{Q}_k$'s followed a normal distribution with a mean of 0.1 and a variance of 0.05, while node-user load constraints, node max load constraints, and node load balance constraints were not taken into account. Similarly, in the simulation for Fig.3.1d (Fig.3.1e, and Fig.3.1f, respectively), the $\mathbb{N}_k(u)$'s ($\mathbb{M}(u)$'s, and $\mathbb{B}(u,w)$'s, respectively) followed a normal distribution with a mean of 0.1 and a variance of 0.05, while the other three constraints were not considered. In addition, the load balance area of each node in the simulation for Fig.3.1f-3.1g was the set of its one-hop neighbors. We compared our algorithm with the cross-layer congestion control,

routing, and scheduling design presented by Chen *et al.* [13] (denoted by "CO" in Fig.3.1c-3.1g). This scheme optimizes the total user utility without considering load constraints. We observe that the metrics for CO in Fig.3.1c-3.1f approach stable values far larger than 0, while in our algorithm these metrics drop quickly to a small value. Thus, our algorithm achieves small utility deficits, node-user load violation ratios, node max load violation ratios, and node load balance violation ratios within a fairly short time.

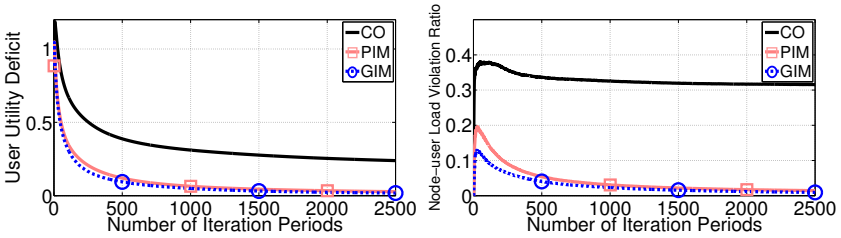
Recall that we introduce the node load balance constraints to shape the load relationship among different nodes. However, such constraint is considered from the perspective of each node. We hence use the following *node load fairness index* to quantify the load balance level from the perspective of the entire network. This index is defined by $\frac{(\sum_{u\in V} R(u))^2}{|V|\sum_{u\in V} R(u)^2}$ [28, 115]. The parameter "B" in Fig.3.1g means that in that test case, the $\mathbb{B}(u, w)$'s followed a normal distribution with a mean of B and a variance of $\frac{B}{2}$. The number of nodes was chosen to be 10 and 30. Fig.3.1g shows that compared with CO, our algorithm increases the fairness index by 13%-41%, and hence achieves a higher level of node load fairness.

3.7 Conclusion

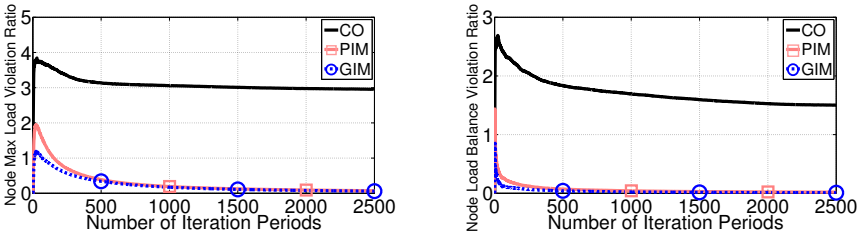
In this work, we have studied the problem of allocating network resources to maximize the total user utility in a load-constrained wireless network. We have formulated this problem as a convex programming system. We have presented an α-approximation dual subgradient algorithmic framework, and have proved the upper bound on the Lagrange multipliers, the upper bound on the amount of feasibility violation, and the upper bound and lower bound on our solution at each iteration. Based on this framework, we have presented a distributed iterative algorithm to solve the resource allocation problem, which provides user rate and



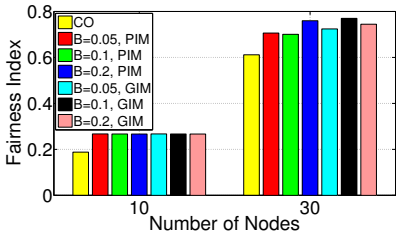
(a) Utility (b) Queue length



(c) User utility deficit (d) Node-user load violation ratio



(e) Node max load violation ratio (f) Node load balance violation ratio



(g) Node load fairness index

Figure 3.1: Numerical results

node rate control strategies in each iteration period. We have proved bounds on the amount of feasibility violation and the gap between our solution and the optimal solution at each iteration. We have also proved bounds on node queue lengths, user utility deficits, and node load violation ratios in each iteration period using dual space information.

Chapter 4

MAP: Multi-Constrained Anypath Routing in Wireless Mesh Networks

4.1 Introduction

4.1.1 Motivation

Traditional routing algorithms and protocols for wireless networks often follow the design methodology for wired networks by abstracting the wireless links as wired links and looking for the shortest delay, least cost, or widest bandwidth path(s) between a pair of source and destination nodes [113]. However, for unreliable wireless networks, due to the broadcast nature of the wireless medium, it is usually less costly to transmit a packet to one of the nodes in a set of neighbors than to one specific neighbor. This observation motivated the emergence of a new technology, known as *opportunistic routing*, which takes advantage of the intermediate nodes overhearing the transmissions. It has been shown that opportunistic routing can help improve the performance of wireless networks [6, 9]. Dubois-Ferrière [25] generalized opportunistic routing and introduced the concept of *anypath routing*, which was subsequently studied in [24, 58, 59, 94, 116]. In anypath routing, each packet is broadcast to a forwarding set composed of several neighbors (called forwarders), and the packet is retransmitted only if none of the forwarders in this set receives it. As long as one of the forwarders receives this packet, it will be forwarded.

One of the key research issues in anypath routing is finding an anypath which optimizes one or more QoS metrics, such as *delay* or *cost*. The anypath computation often reduces to the selection of forwarders for each node. However, there is a tradeoff in selecting forwarders. On one hand, a node with more forwarders can have less forwarding delay or cost to reach one of the these for-

warders. On the other hand, a neighbor not on the optimal single path does not make as much progress as the next hop on the optimal single path. Therefore, having too many forwarders may increase the likelihood of a packet veering away from the optimal single path, and ultimately even result in loops in the routing topology [24].

Previous works on anypath routing are focused on computing a shortest delay or least cost anypath, with only one QoS metric taken into account for route selection [24, 25, 58, 59]. However, many applications are associated with multiple QoS constraints. For instance, usually the energy consumption affects the network lifetime or the cost of the pair of source-destination nodes charged by the intermediate nodes providing forwarding service. Obviously, both delay and energy consumption should be taken into account when we are computing a route between a pair of source-destination nodes.

There have been extensive studies on multi-constrained single path routing. Since the problem is NP-hard [105], many heuristics and approximation algorithms have been proposed [14,42,66,105,107,109]. However, to the best of our knowledge, multi-constrained anypath routing has not been studied.

4.1.2 Contribution

In this work, we formulate and study the problem of anypath routing subject to multiple (K) constraints. We show that the problem is NP-hard when the number of constraints is larger than one. We then present two polynomial time K-approximation algorithms for this problem. One is a centralized algorithm while the other is a distributed algorithm.

4.1.3 Related Work

We now review the works on opportunistic routing and anypath routing theory. Biswas and Morris [6] designed and implemented ExOR, an opportunistic routing protocol for wireless mesh networks. Chachulski *et al.* [9] introduced MORE by combining opportunistic routing and network coding. Dubois-Ferrière [25] introduced the concept of anypath routing and studied the shortest anypath problem. Dubois-Ferrière *et al.* [24] addressed the least-cost anypath routing problem. The authors of [58] and [59] presented an optimal algorithm for computing a shortest anypath when different nodes are allowed to use different transmission rates. Schaefer *et al.* [94] proposed a coordinated anypath routing for energy-constrained wireless sensor networks. Zorzi and Rao [117] combined opportunistic and geographic routing for wireless sensor networks.

For multi-constrained QoS routing algorithms, there are two lines of research. One line is the design of provably good algorithms. The authors of [66, 79, 103, 107, 109] presented fully polynomial time approximation schemes (FPTASs) for the delay constrained least cost single path problem. For the decision version of MCP, Xue *et al.* [109] improved the algorithm proposed in [14]. Xue *et al.* [107] introduced a metric to compare multiple feasible solutions to the decision version of MCP. Based on this metric they studied an optimization version of MCP, and proposed K-approximation algorithms for computing a provably good single path. Tsaggouris and Zaroliagis [103] presented an FPTAS for a class of multiobjective optimization problems. The other line of research is the development of effective heuristic algorithms [14, 53, 112]. Two excellent surveys can be found in [15, 56]. A comparison of approximation and heuristic algorithms can be found in [55].

Our work differs from the aforementioned works since we study anypath routing subject to multiple constraints.

4.2 Model

In anypath routing, each node broadcasts a packet to one or more next hop *neighbors*. As long as one of these neighbors receives this packet, the packet can be forwarded on. We call this set of next hop neighbors a *forwarding set*, which is similar to "next hop" for each node in classic routing. We also call node v a *forwarder* of node u if v is in the forwarding set of u. Since more than one node in a forwarding set may receive the same packet, unnecessary redundant forwarding should be avoided. In order to suppress redundant forwarding, the nodes in a forwarding set are each given a priority in relaying the received packets. Higher priorities are assigned to the nodes with shorter distances (or less costs) to the destination. A node forwards a received packet only when all higher priority nodes in the same forwarding set fail to receive it. As a result, this node will forward the packet towards the destination while other lower priority nodes suppress their transmissions, even if they also receive the packet. For each packet, the source keeps rebroadcasting it until someone in its forwarding set receives it or a threshold is reached. Once a neighbor receives this packet, it will repeat the same procedure until the packet is delivered to the destination.

As defined in [25], an *anypath* from a source to a destination is a directed acyclic graph where every node (but the source) is a successor of the source, and every node (but the destination) is a predecessor of the destination. Since an anypath is acyclic, all of its potential paths are simple, hence no packet will traverse a forwarder more than once.

We model a wireless mesh network by a directed graph $G = (V, E)$, where

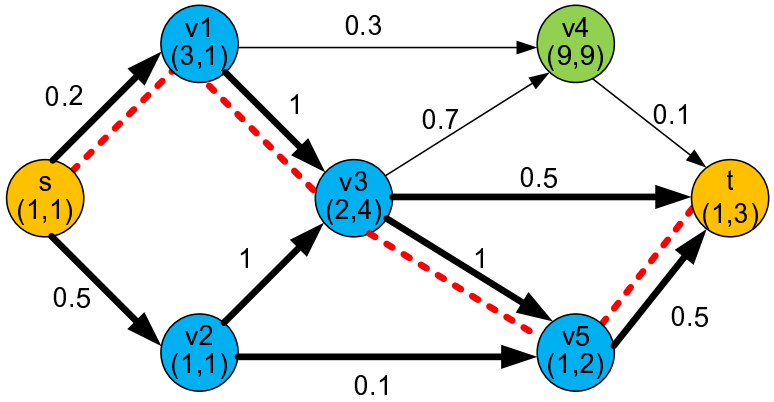


Figure 4.1: Illustration of anypaths and subanypaths. The link labels show link delivery probabilities; the vertex labels show (w_1, w_2) pairs. For example, w_1 and w_2 represent the average transmission time and the average energy consumption of the node for each transmission, respectively.

V is the set of n vertices, and E is the set of m edges. We use the following terms interchangeably: edge and link, vertex and node. Each edge $(v, u) \in E$ is associated with a *packet delivery probability* $p(v, u)$. Let J be a forwarding set of node v. The *hyperlink delivery probability* $p(v, J)$ is defined as the probability that a packet transmitted by node v is received by at least one of the nodes in set J. As demonstrated in [58], the loss of a packet at different receivers occurs independently in practice, such as in light load regimes. Therefore, the hyperlink delivery probability can be computed as $p(v, J) = 1 - \prod_{j \in J}(1 - p(v, j))$. For example, consider the forwarding set $J = \{v_1, v_2\}$ of node s in the network in Fig. 4.1. We have $p(s, J) = 1-(1-p(s, v_1))(1-p(s, v_2)) = 1-(1-0.2)(1-0.5) = 0.6$.

Each node $v \in V$ is also associated with K *vertex weights* $w_k(v)$, $1 \leq k \leq K$, where K is a given integer. We use the following examples to explain some possible physical meaning of vertex weights. For example, when $K = 1$, $w_1(v)$ may represent average time required for node v to finish one transmission on average (we call it *transmission time*). When $K = 2$, $w_1(v)$ may represent the

transmission time and $w_2(v)$ may represent the energy consumed by node v to finish one transmission on average (we call it *transmission energy consumption*).

Based on vertex weights and link delivery probabilities, we define a new metric, called *expected weight of anypath transmissions* (EWATX), as $\frac{w_k(v)}{p(v,J)}, 1 \leq k \leq K$. Consider the example of $K = 2$ mentioned before, where $w_1(v)$ represents the average transmission time for each transmission and $w_2(v)$ represents the average energy consumption for each transmission. The first (second, respectively) EWATX represents the expected transmission time (the expected energy consumption, respectively) for a packet sent by node v to be successfully received by at least one node in J.

Recall that the hyperlink delivery probability from a node v to its forwarding set (on the anypath P_s from source s to destination t) $J(v, P_s) = <j_1, j_2, \ldots>$ is

$$p(v, J(v, P_s)) \;=\; 1 - \prod_{j_\beta \in J(v,P_s)} (1 - p(v, j_\beta)) = \sum_{j_\beta \in J(v,P_s)} p(v, j_\beta) \prod_{q=1}^{\beta-1} (1 - p(v, j_q)).$$

We define the coefficient $\alpha(j_\beta, P_s)$ as follows:

$$\alpha(j_\beta, P_s) = \frac{p(v, j_\beta) \prod_{q=1}^{\beta-1} (1 - p(v, j_q))}{p(v, J(v, P_s))}, \forall j_\beta \in J(v, P_s). \tag{4.1}$$

The numerator of the right hand side of Equation (4.1) is the probability that a node in the forwarding set with priority β successfully forwards a packet. Recall that it forwards a packet only when it receives this packet and none of the higher priority nodes receives it. The denominator in Equation (4.1) is the normalizing constant so that $\sum_{j_\beta \in J(v,P_s)} \alpha(j_\beta, P_s) = 1$.

We define *the k^{th} anypath weight of the forwarding set* $J(v, P_s) = <j_1, j_2, ...>$ along anypath P_s as

$$\mathcal{W}_k(J(v, P_s), P_s) = \sum_{j_\beta \in J(v,P_s)} \alpha(j_\beta, P_s) \mathcal{W}_k(j_\beta, P_s). \tag{4.2}$$

It is a weighed average of *the anypath weights from the nodes* (defined below) in the forwarding set $J(v, P_s)$ to the destination along P_s.

Based on Equations (4.1) and (4.2), we define the k^{th} *anypath weight from* v *to* t along an anypath P_s as

$$\mathcal{W}_k(v, P_s) = \frac{w_k(v)}{p(v, J(v, P_s))} + \mathcal{W}_k(J(v, P_s), P_s), k = 1, 2, ..., K. \tag{4.3}$$

Clearly, from Equations (4.1)-(4.3) we know that

$$\mathcal{W}_k(v, P_s) = \frac{w_k(v) + \sum_{j_\beta \in J(v,P)} \mathcal{W}_k(j_\beta, P_s) p(v, j_\beta) \prod_{q=1}^{\beta-1} (1 - p(v, j_q))}{p(v, J(v, P_s))}, \\ k = 1, 2, ..., K. \tag{4.4}$$

Note that we define the k^{th} anypath weight in a recursive manner. The boundary condition is $\mathcal{W}_k(t, P_s) = 0 (1 \leq k \leq K)$ for destination node t.

Consider the example of $K = 2$ mentioned before, where $w_1(v)$ represents the average transmission time for each transmission and $w_2(v)$ represents the average energy consumption for each transmission. $\mathcal{W}_1(v, P_s)$ ($\mathcal{W}_2(v, P_s)$, respectively) represents the expected total transmission time (the expected total energy consumption, respectively) necessary for a packet sent by v to be successfully received by the destination node t along anypath P_s. Since an anypath is a directed acyclic graph, once the anypath from s to t is given, we can compute the anypath weights of all the nodes on this anypath in a reversed topological order.

As an example, let us consider the network depicted in Fig. 4.2 which is slightly different from Fig. 4.1 and focus on the transmission time ($k = 1$).

An anypath P_s is shown by bold red lines. The topological order for this anypath is $s \rightarrow v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow t$. The boundary condition is $\mathcal{W}_1(t, P_s) = 0$. We can first compute the first anypath weight of v_5 (i.e. the expected transmission

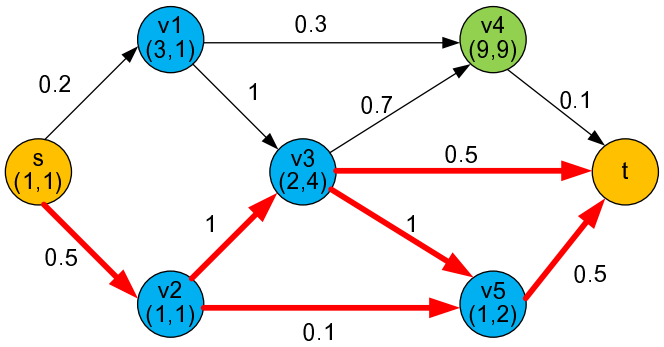


Figure 4.2: An example to illustrate the computation of anypath weight

time of a packet sent from v_5 to t via the forwarding set $J = <t>$) as follows:

$$\mathcal{W}_1(v_5, P_s) = \frac{w_1(v_5)}{p(v_5, <t>)} + \mathcal{W}_1(<t>, P_s)$$
$$= \frac{w_1(v_5) + p(v_5, t)\mathcal{W}_1(t, P_s)}{p(v_5, <t>)} = \frac{1 + 0.5 * 0}{0.5} = 2.$$

Based on the values at t and v_5, we can then compute the first anypath weight of v_3 (i.e. the expected transmission time of a packet sent from v_3 to t via the forwarding set $J = <t, v_5>$) as follows:

$$\begin{aligned}
&\mathcal{W}_1(v_3, P_s) \\
=&\frac{w_1(v_3)}{p(v_3, <t, v_5>)} + \mathcal{W}_1(<t, v_5>, P_s) \\
=&\frac{w_1(v_3)+p(v_3,t)\mathcal{W}_1(t,P_s)+(1-p(v_3,t))p(v_3,v_5)\mathcal{W}_1(v_5,P_s)}{1-(1-p(v_3,t))(1-p(v_3,v_5))} \\
=&\frac{2 + 0.5 \times 0 + (1 - 0.5) \times 1 \times 2}{1 - (1 - 0.5)(1 - 1)} = 3.
\end{aligned}$$

Continuing this process, we can compute $\mathcal{W}_1(v_2, P_s) = 3.9$, and $\mathcal{W}_1(s, P_s) = 5.9$ (assuming that the forwarding set of v_2 is $<v_5, v_3>$).

4.3 Problem Formulation

A *decision version* of the multi-constrained anypath problem, denoted by DM-CAP, is defined as follows:

Definition 1. *DMCAP*$(G, s, t, \overrightarrow{\mathbb{W}}, \overrightarrow{w}, \overrightarrow{p}, K)$*:* ***Instance****: A directed graph* $G = (V, E, \overrightarrow{w}, \overrightarrow{p})$*, with a packet delivery probability* $p(e) \in (0, 1]$ *associated with each edge* $e \in E$ *and* K *positive vertex weights* $w_k(v)(k \in [1, K])$ *associated with each vertex* $v \in V$*; a constraint vector* $\overrightarrow{\mathbb{W}} = (\mathbb{W}_1, \mathbb{W}_2, ..., \mathbb{W}_K)$ *where each element is a positive constant; and a source-destination node pair* (s, t)*.* ***Problem****: Find an anypath* P_s *from source* s *to destination* t *such that* $\mathcal{W}_k(s, P_s) \leq \mathbb{W}_k, \forall k \in [1, K]$.□

The inequality $\mathcal{W}_k(s, P_s) \leq \mathbb{W}_k$ is called *the* k^{th} *QoS constraint*. An anypath P_s from s to t satisfying all K constraints is called a *feasible* solution to the given instance of DMCAP.

The DMCAP problem is NP-hard, which we will show later. Therefore, we study OMCAP, an optimization version of DMCAP, which approximates all K constraints simultaneously. We introduce a metric, called *anypath length*, which is similar to the concept of *path length* used in [107, 108]. The anypath length from a node v to the destination node t along an anypath P_s is defined as

$$l(v, P_s) = \max_{1 \leq k \leq K} \frac{\mathcal{W}_k(v, P_s)}{\mathbb{W}_k}. \tag{4.5}$$

Now we formally define OMCAP as follows.

Definition 2. *OMCAP*$(G, s, t, \overrightarrow{\mathbb{W}}, \overrightarrow{w}, \overrightarrow{p}, K)$*:* ***Instance****: A directed graph* $G = (V, E, \overrightarrow{w}, \overrightarrow{p})$*, with a packet delivery probability* $p(e) \in (0, 1]$ *associated with each edge* $e \in E$ *and* K *positive vertex weights* $w_k(v)(k \in [1, K])$ *associated with each vertex* $v \in V$*; a constraint vector* $\overrightarrow{\mathbb{W}} = (\mathbb{W}_1, \mathbb{W}_2, ..., \mathbb{W}_K)$ *where each element is a positive constant; and a source-destination node pair* (s, t)*.* ***Problem****: Find an* s*-*t *anypath* P_s *with minimum* $l(s, P_s)$*.* □

We briefly explain the relationship between DMCAP and OMCAP. If an instance of OMCAP has a solution P_s such that $l(s, P_s) \leq 1$, then the correspond-

ing instance of DMCAP has a feasible solution. Otherwise, the corresponding instance of DMCAP does not have a feasible solution.

4.4 Analysis of Computational Complexity

Theorem 7. *For any given integer $K \geq 2$, DMCAP is NP-hard. The problem remains NP-hard when no link in the network has packet delivery probability equal to* 1. □

In the following, we will prove the (weak) NP-hardness of DMCAP for a special case $K = 2$ by a reduction from the Partition problem [21], which implies that DMCAP is (weakly) NP-hard for any given $K \geq 2$.

An instance of Partition is given by a finite set A, where each element $a \in A$ is associated with a positive integer $s(a)$, called as the *size* of a. It asks for the existence of a subset A' of A such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$. This problem is known to be (weakly) NP-hard [21].

In the following, we show how to construct an instance $\mathcal{I}_2$ of DMCAP for $K = 2$ from an instance $\mathcal{I}_1$ of Partition, given by $A = \{a_1, a_2, \ldots, a_n\}$ and size function $s(\cdot)$. An illustrative example is shown in Fig. 4.3.

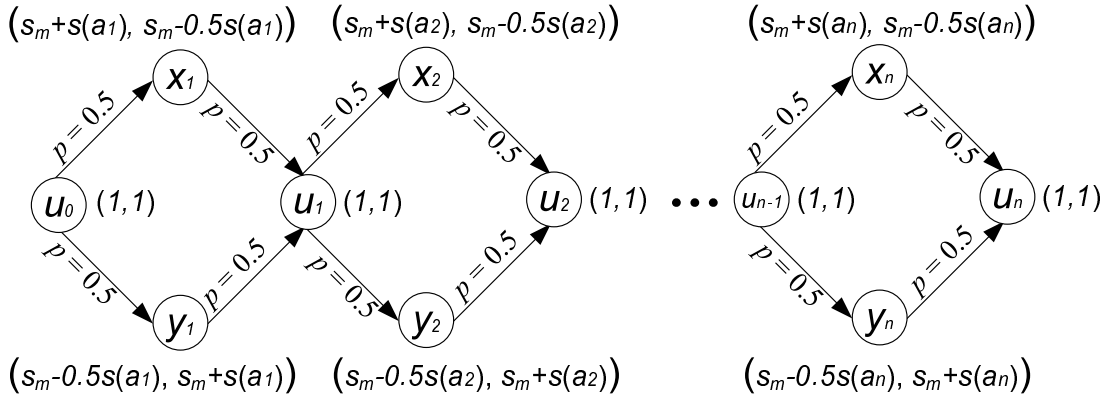


Figure 4.3: Reduction from Partition to DMCAP for $K = 2$.

The set of nodes of graph $G(V, E)$ is given by $V = \{u_0, u_1, \ldots, u_n; x_1, x_2, \ldots, x_n; y_1, y_2, \ldots, y_n\}$. The vertex weights of nodes in the sets $\{u_0, u_1, \ldots, u_i, \ldots, u_n\}$ are all set to $(1, 1)$. The vertex weights of node $x_i \in \{x_1, x_2, \ldots, x_n\}$ are set to $(s_m + s(a_i), s_m - 0.5s(a_i))$, where s_m is the largest size of the elements in A (i.e. $s_m = \max_{a \in A} s(a)$). The vertex weights of node $y_i \in \{y_1, y_2, \ldots, y_n\}$ are set to $(s_m - 0.5s(a_i), s_m + s(a_i))$. Note that the sizes of all the elements in A are positive. Therefore all the vertex weights so constructed are positive. The set of directed links is given by $\{(u_{i-1}, x_i), (x_i, u_i), (u_{i-1}, y_i), (y_i, u_i) | i \in [1, n]\}$. The packet delivery probabilities of all the links are set to 0.5. We set $\mathbb{W}_1 = \mathbb{W}_2 = \frac{\sum_{1 \le i \le n} s(a_i)}{2} + 2ns_m + \frac{4n}{3}$. In $\mathcal{I}_2$, we are looking for an anypath from u_0 to u_n which satisfies the above two constraints. The construction of $\mathcal{I}_2$ is complete and takes polynomial time.

Clearly, for each node u_i $(i = 0, \cdots, n-1)$, we have the following four forwarding set candidates: $<x_{i+1}, y_{i+1}>$, $<y_{i+1}, x_{i+1}>$, $<x_{i+1}>$, and $<y_{i+1}>$. One of them is selected and used as the forwarding set for u_i $(i = 0, ..., n-1)$. Next, let us compute the corresponding anypath weights of node u_i corresponding to these four forwarding sets.

We use $\mathcal{W}_1(u_i)$ and $\mathcal{W}_2(u_i)$ to denote the first and the second anypath weights from u_i to u_n along an anypath, respectively. Clearly, $\mathcal{W}_1(u_n) = \mathcal{W}_2(u_n) = 0$. Given $\mathcal{W}_1(u_{i+1})$ and $\mathcal{W}_2(u_{i+1})$ $(i = 0, ..., n-1)$, we can compute $\mathcal{W}_k^{<J>}(u_i)$ for $k = 1, 2$, which denotes the kth anypath weight from u_i to u_n if we use J as the

forwarding set of node u_i. Based on (4.3), we derive the following equations:

$$\mathcal{W}_1^{<x_{i+1},y_{i+1}>}(u_i) = \frac{4}{3} + 2s_m + s(a_{i+1}) + \mathcal{W}_1(u_{i+1}); \tag{4.6}$$

$$\mathcal{W}_1^{<y_{i+1},x_{i+1}>}(u_i) = \frac{4}{3} + 2s_m + \mathcal{W}_1(u_{i+1}); \tag{4.7}$$

$$\mathcal{W}_1^{<x_{i+1}>}(u_i) = 2 + 2s_m + 2s(a_{i+1}) + \mathcal{W}_1(u_{i+1}); \tag{4.8}$$

$$\mathcal{W}_1^{<y_{i+1}>}(u_i) = 2 + 2s_m - s(a_{i+1}) + \mathcal{W}_1(u_{i+1}); \tag{4.9}$$

$$\mathcal{W}_2^{<x_{i+1},y_{i+1}>}(u_i) = \frac{4}{3} + 2s_m + \mathcal{W}_2(u_{i+1}); \tag{4.10}$$

$$\mathcal{W}_2^{<y_{i+1},x_{i+1}>}(u_i) = \frac{4}{3} + 2s_m + s(a_{i+1}) + \mathcal{W}_2(u_{i+1}); \tag{4.11}$$

$$\mathcal{W}_2^{<x_{i+1}>}(u_i) = 2 + 2s_m - s(a_{i+1}) + \mathcal{W}_2(u_{i+1}); \tag{4.12}$$

$$\mathcal{W}_2^{<y_{i+1}>}(u_i) = 2 + 2s_m + 2s(a_{i+1}) + \mathcal{W}_2(u_{i+1}). \tag{4.13}$$

We use binary variables $Z_y^x(u_i)$, $Z_x^y(u_i)$, $Z^x(u_i)$, and $Z^y(u_i)$ to denote the following four forwarding set selections of node $u_i (i = 0, ..., n-1)$, respectively.

- If forwarding set $<x_{i+1}, y_{i+1}>$ is used as the forwarding set of u_i, then $Z_y^x(u_i) = 1$ and $Z_x^y(u_i) = Z^x(u_i) = Z^y(u_i) = 0$.
- If forwarding set $<y_{i+1}, x_{i+1}>$ is used as the forwarding set of u_i, then $Z_x^y(u_i) = 1$ and $Z_y^x(u_i) = Z^x(u_i) = Z^y(u_i) = 0$.
- If forwarding set $<x_{i+1}>$ is used as the forwarding set of u_i, then $Z^x(u_i) = 1$ and $Z_y^x(u_i) = Z_x^y(u_i) = Z^y(u_i) = 0$.
- If forwarding set $<y_{i+1}>$ is used as the forwarding set of u_i, then $Z^y(u_i) = 1$ and $Z_y^x(u_i) = Z_x^y(u_i) = Z^x(u_i) = 0$.

According to (4.6)-(4.13), we can start from node u_n and compute the anypath weights from $u_l (l = 0, ..., n-1)$ to u_n using notation $Z_y^x(u_l)$, $Z_x^y(u_l)$,

$Z^x(u_l)$, and $Z^y(u_l)$, as follows:

$$\mathcal{W}_1(u_l) = \sum_{i=l}^{n-1} \Bigg(Z_y^x(u_i)(\frac{4}{3} + 2s_m + s(a_{i+1})) + Z_x^y(u_i)(\frac{4}{3} + 2s_m) \\ + Z^x(u_i)(2 + 2s_m + 2s(a_{i+1})) + Z^y(u_i)(2 + 2s_m - s(a_{i+1})) \Bigg), \tag{4.14}$$

and

$$\mathcal{W}_2(u_l) = \sum_{i=l}^{n-1} \Bigg(Z_y^x(u_i)(\frac{4}{3} + 2s_m) + Z_x^y(u_i)(\frac{4}{3} + 2s_m + s(a_{i+1})) \\ + Z^x(u_i)(2 + 2s_m - s(a_{i+1})) + Z^y(u_i)(2 + 2s_m + 2s(a_{i+1}))) \Bigg). \tag{4.15}$$

We then prove the following two lemmas by leveraging (4.14) and (4.15).

Lemma 4. *If instance $\mathcal{I}_2$ has a solution, then instance $\mathcal{I}_1$ also has a solution.* □

Proof. If instance $\mathcal{I}_2$ has a solution, we know

$$\mathcal{W}_1(u_0) \leq \mathbb{W}_1 = \frac{\sum_{1 \leq i \leq n} s(a_i)}{2} + 2ns_m + \frac{4n}{3} \tag{4.16}$$

and

$$\mathcal{W}_2(u_0) \leq \mathbb{W}_2 = \frac{\sum_{1 \leq i \leq n} s(a_i)}{2} + 2ns_m + \frac{4n}{3}. \tag{4.17}$$

Therefore, we have

$$\mathcal{W}_1(u_0) + \mathcal{W}_2(u_0) \leq \sum_{1 \leq i \leq n} s(a_i) + 4ns_m + \frac{8n}{3}. \tag{4.18}$$

According to (4.14), (4.15), and (4.18), we know that $Z^x(u_i)$ and $Z^y(u_i)$ *have to* be zero for $i = 0, 1, ..., n-1$. Otherwise, (4.18) can never be satisfied. In other words, for node u_i only $<x_{i+1}, y_{i+1}>$ and $<y_{i+1}, x_{i+1}>$ are possible forwarding sets in the solution to instance $\mathcal{I}_2$. Therefore, according to (4.14) and (4.15), we have

$$
\begin{aligned}
&\mathcal{W}_1(u_0) \\
&= \sum_{i=0}^{n-1} \left(Z_y^x(u_i)(\frac{4}{3} + 2s_m + s(a_{i+1})) + (1 - Z_y^x(u_i))(\frac{4}{3} + 2s_m) \right) \\
&= \frac{4n}{3} + 2ns_m + \sum_{i=0}^{n-1} Z_y^x(u_i)s(a_{i+1}), \qquad (4.19)
\end{aligned}
$$

and

$$
\begin{aligned}
&\mathcal{W}_2(u_0) \\
&= \sum_{i=0}^{n-1} \left(Z_y^x(u_i)(\frac{4}{3} + 2s_m) + (1 - Z_y^x(u_i))(\frac{4}{3} + 2s_m + s(a_{i+1})) \right) \\
&= \frac{4n}{3} + 2ns_m + \sum_{i=0}^{n-1} (1 - Z_y^x(u_i))s(a_{i+1}). \qquad (4.20)
\end{aligned}
$$

According to (4.19) and (4.20), we know that

$$
\mathcal{W}_1(u_0) + \mathcal{W}_2(u_0) = \sum_{1 \le i \le n} s(a_i) + 4ns_m + \frac{8n}{3}. \qquad (4.21)
$$

Considering (4.16), (4.17), and (4.21) we know that

$$
\mathcal{W}_1(u_0) = \mathcal{W}_2(u_0) = \frac{4n}{3} + 2ns_m + \frac{\sum_{1 \le i \le n} s(a_i)}{2}. \qquad (4.22)
$$

According to (4.19), (4.20), and (4.22), we thus can construct a solution to instance $\mathcal{I}_1$ according to the solution to instance $\mathcal{I}_2$ as follows. If $<x_{i+1}, y_{i+1}>$ (i.e. $Z_y^x(u_i) = 1$) is the forwarding set of u_i in the solution to instance $\mathcal{I}_2$, $s(a_{i+1})$ is added to A'. If $<y_{i+1}, x_{i+1}>$ (i.e. $Z_y^x(u_i) = 0$) is the forwarding set of u_i, $s(a_{i+1})$ is added to $A \setminus A'$. Due to (4.19), (4.20), and (4.22), this assignment can ensure that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a) = \frac{\sum_{1 \le i \le n} s(a_i)}{2}$. ■

Lemma 5. *If instance $\mathcal{I}_1$ has a solution, then instance $\mathcal{I}_2$ also has a solution.* □

Proof. We can construct a solution to instance $\mathcal{I}_2$ according to the solution to $\mathcal{I}_1$ as follows. If a_i is in A', $<x_i, y_i>$ is chosen as the forwarding set for node u_{i-1} (i.e.

$Z_y^x(u_{i-1}) = 1$, $Z_x^y(u_{i-1}) = 0$, $Z^x(u_{i-1}) = 0$, and $Z^y(u_{i-1}) = 0$); otherwise $<y_i, x_i>$ is chosen as the forwarding set for node u_{i-1} (i.e. $Z_y^x(u_{i-1}) = 0$, $Z_x^y(u_{i-1}) = 1$, $Z^x(u_{i-1}) = 0$, and $Z^y(u_{i-1}) = 0$). Therefore, according to (4.14) and (4.15), we have

$$\begin{aligned}
&\mathcal{W}_1(u_0) \\
&= \sum_{i=0}^{n-1} \left(Z_y^x(u_i)(\frac{4}{3} + 2s_m + s(a_{i+1}) + (1 - Z_y^x(u_i))(\frac{4}{3} + 2s_m) \right) \\
&= \frac{4n}{3} + 2ns_m + \sum_{i=0}^{n-1} Z_y^x(u_i)s(a_{i+1}) \\
&= \frac{4n}{3} + 2ns_m + \sum_{a \in A'} s(a) = \mathbb{W}_1
\end{aligned} \tag{4.23}$$

and

$$\begin{aligned}
&\mathcal{W}_2(u_0) \\
&= \sum_{i=0}^{n-1} \left(Z_y^x(u_i)(\frac{4}{3} + 2s_m) + (1 - Z_y^x(u_i))(\frac{4}{3} + 2s_m + s(a_{i+1})) \right) \\
&= \frac{4n}{3} + 2ns_m + \sum_{i=0}^{n-1} (1 - Z_y^x(u_i))s(a_{i+1}) \\
&= \frac{4n}{3} + 2ns_m + \sum_{a \in A \setminus A'} s(a) = \mathbb{W}_2.
\end{aligned} \tag{4.24}$$

Note that (4.23) and (4.24) hold since we have a solution to instance $\mathcal{I}_1$ that ensures $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a) = \frac{\sum_{1 \le i \le n} s(a_i)}{2}$. ■

Combining our reduction, the two lemmas above, and the fact that the Partition problem is (weakly) NP-hard, we know that DMCAP is (weakly) NP-hard when $K = 2$, which implies that DMCAP is (weakly) NP-hard for any given $K \ge 2$. Hence, the proof for Theorem 7 is complete.

4.5 An Efficient Centralized K-Approximation Algorithm

In this subsection we present an algorithm named MAP, which computes an s-t anypath in polynomial time. Furthermore, we prove that the anypath so computed

is a K-approximation to the OMCAP problem. In other words, the length of the anypath returned by our algorithm is within a factor K of the length of an optimal solution.

In order to simplify the notations, we assume that $\mathbb{W}_1{=}\mathbb{W}_2{=}\cdots{=}\mathbb{W}_K{=}1$ in this subsection. This is equivalent to normalize all of the vertex weights (from $w_k(v)$ to $w_k(v)/\mathbb{W}_k$ for all $v \in V$ and $k = 1, 2, \ldots, K$). Therefore the algorithm and all the proofs can be straightforwardly extended to the case without this assumption.

4.5.1 Algorithm Description

Before presenting our algorithm, we define an *auxiliary vertex weight* $\omega_m(v)$ (the subscript m means "max") for each node $v \in V$ by $\omega_m(v) = \max_{1\le k\le K} w_k(v)$. We define the *auxiliary anypath weight* (AAW) from node v to destination t along an anypath P_s as:

$$\mathcal{W}_m(v, P_s) = \frac{\omega_m(v)}{p(v, J(v, P_s))} + \mathcal{W}_m(J(v, P_s), P_s), \tag{4.25}$$

where

$$\mathcal{W}_m(J(v, P_s), P_s) = \sum_{j_\beta \in J(v,P_s)} \alpha(j_\beta, P_s)\mathcal{W}_m(j_\beta, P_s). \tag{4.26}$$

Here $\mathcal{W}_m(J(v, P_s), P_s)$ is the auxiliary anypath weight of the forwarding set $J(v, P_s)$ along the anypath P_s. The boundary condition is $\mathcal{W}_m(t, P_s) = 0$.

We introduce AAW because it can be used to bridge $l(s, \pi_s^{opt})$ and $l(s, \pi_s^m)$, where π_s^{opt} denotes an optimal anypath from s to t and π_s^m denotes the anypath computed by Algorithm 2. More specifically, in Theorem 8 we will prove that π_s^m is a shortest s–t anypath with respect to AAW; in Theorem 9, we will prove that $l(v, \pi_v^m) \le K \cdot l(v, \pi_v^{opt})$, which implies that Algorithm 2 is a K-approximation algorithm for OMCAP.

Table 4.1 lists frequently used notations.

Table 4.1: Frequently Used Notation

$w_k(v)$	k^{th} vertex weight of node v
$\mathcal{W}_k(v,P)$	k^{th} anypath weight from node v to destination t along anypath P
$\omega_m(v)$	auxiliary vertex weight of node v
$\mathcal{W}_m(v,P)$ or $\mathcal{W}_m(J,P)$	auxiliary anypath weight from node v or a set of nodes J (aggregated vertex) to destination t along anypath P
$\mathcal{W}_m^{(J)}(v)$	auxiliary anypath weight from v to destination t via the forwarding set J
$\mathbb{W}_k$	k^{th} QoS constraint
$l(v,P)$	length from v to destination t along anypath P
$J(v,P)$	forwarding set of node v along anypath P
$p(v,u)$ or $p(v,J)$	packet delivery probability of link (v,u) or hyperlink (v,J)

We now present our MAP algorithm for the OMCAP problem. For every node $v \in V$, we keep a variable $\hat{W}_m(v)$, a set $\hat{J}(v)$, and K variables $\hat{W}_k(v), k = 1, \cdots, K$. $\hat{W}_m(v)$ is the AAW of the currently computed anypath from v to t, $\hat{J}(v)$ is the forwarding set of v on this currently computed anypath, and $\hat{W}_k(v)$ is the currently computed kth anypath weight from v to t. We keep two data structures: L and Q. L is a list, which is used to store the nodes for which we have already found the shortest AAW anypaths. Q is a priority queue, in which we store all the other nodes keyed by their current $\hat{W}_m$ values. In addition, we also use $\hat{W}_m(J)$ and $\hat{W}_k(J)$, respectively, to denote the currently computed AAW and the currently computed kth anypath weight of the set J.

The algorithm MAP is presented in Algorithm 2. MAP consists of two major phases. In the first phase, it calculates the AAW for each node and initializes the data structures (Lines 1-4). In the second phase, it computes the *shortest AAW anypaths* from all nodes to the given destination t (Lines 5-17). Our algorithm for computing the shortest AAW anypaths is similar to the Short-

Algorithm 2 MAP

Input: graph G, destination node t, vertex weight vector $\overrightarrow{w}$, delivery probability vector $\overrightarrow{p}$, and the number of constraints K
Output: an anypath from each node to destination node t

1: **for** each node v in V **do**
2: $\omega_m(v) \leftarrow \max_{1\le k\le K} w_k(v)$, $\hat{W}_m(v) \leftarrow \infty$, $\hat{J}(v) \leftarrow \emptyset$
3: **end for**
4: $\hat{W}_m(t) \leftarrow 0$, $L \leftarrow \emptyset$, $Q \leftarrow V$
5: **while** $Q \neq \emptyset$ **do**
6: $j \leftarrow$ Extract-Min(Q), $L \leftarrow L \cup \{j\}$
7: $l(j, P) \leftarrow \max_{1\le k\le K} \hat{W}_k(j)$
8: **for** each incoming link (v, j) in E **do**
9: $F \leftarrow \hat{J}(v) \cup \{j\}$, $\hat{W}'_m(v) \leftarrow \frac{\omega_m(v)}{p(v,F)} + \hat{W}_m(F)$
10: **if** $\hat{W}'_m(v) < \hat{W}_m(v)$ **then**
11: $\hat{W}_m(v) \leftarrow \hat{W}'_m(v)$, update Q (i.e. decrease key for v), $\hat{J}(v) \leftarrow F$
12: **for** $k = 1, 2, \ldots, K$ **do**
13: $\hat{W}_k(v) \leftarrow \frac{w_k(v)}{p(v,\hat{J}(v))} + \hat{W}_k(\hat{J}(v))$
14: **end for**
15: **end if**
16: **end for**
17: **end while**

est Anypath First (SAF) algorithm proposed by [58, 59]. We will prove that the shortest AAW anypath actually is a K-approximation to OMCAP.

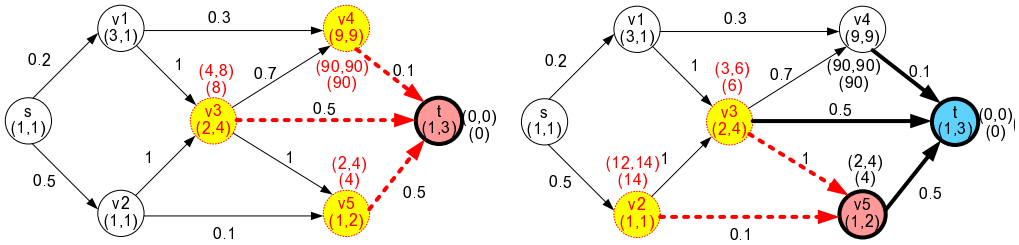
Now we explain the second phase (the while-loop) in detail. Each time Line 6 is executed, from the set of nodes for which we have not found the shortest AAW anypaths, we extract the one (denoted by j) that has the smallest $\hat{W}_m$ value and insert this node into L. We then perform relaxation for each of its incoming neighbors (denoted by v) as shown in Lines 8-16. In the relaxation operation for node v, we check whether adding node j into v's forwarding set can reduce v's $\hat{W}_m$ value (Lines 9-6). If so, we update v's forwarding set, the priority queue Q and the corresponding data structures as shown in Lines 7-12. Algorithm 2 repeats this procedure until the priority queue Q becomes empty. In this way we can find the shortest AAW anypath from each node to the given destination.

4.5.2 Algorithm Illustration

We illustrate the steps of the algorithm using the example shown in Fig. 4.4, where $K = 2$. The original graph G is shown in Fig. 4.1.

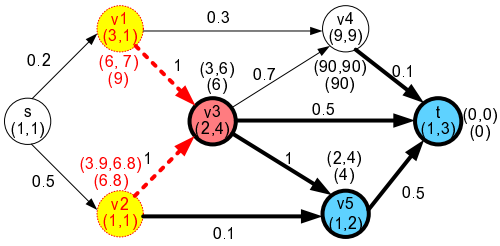
In the initialization phase (Lines 1-4), we calculate the auxiliary vertex weight for each node and initialize the necessary data structures. $\hat{W}_m(t)$ is set to 0 and $\hat{W}_m(v)$ is set to ∞ for all $v \in V \setminus \{t\}$. Since now t has the smallest $\hat{W}_m$ value, in the first iteration, Line 6 extracts t from Q and inserts it into L, as shown in Fig. 4.4a. Then Lines 8-16 perform relaxations for its incoming links and update the values of $\hat{W}_m$ of its incoming neighbors. For each incoming neighbor, Line 6 checks whether adding t into the forwarding set can decrease its $\mathcal{W}_m$ value. If so, Lines 7-12 update its forwarding set, AAW and anypath weights. Therefore, at the end of the current iteration, nodes v_3, v_4 and v_5 get updated labels: $\left(\hat{W}_1(v_3), \hat{W}_2(v_3)\right) = (4, 8)$, $\hat{W}_m(v_3) = 8$, $\left(\hat{W}_1(v_4), \hat{W}_2(v_4)\right) = (90, 90)$, $\hat{W}_m(v_4) = 90$, $\left(\hat{W}_1(v_5), \hat{W}_2(v_5)\right) = (2, 4)$, and $\hat{W}_m(v_5) = 4$. Since now v_5 has the smallest $\hat{W}_m$ value, Line 6 will extract v_5 from Q and insert it into L in the second iteration and perform relaxations for its incoming links. Nodes v_2 and v_3 get updated labels due to the relaxations. This is shown in Fig. 4.4b. As shown in Fig. 4.4c-4.4g, Algorithm 2 repeats this procedure until all the nodes have been inserted into L. Note that the algorithm calculates anypaths from all the nodes to t. If we just look for an s-t anypath, our algorithm can terminate once s is inserted into L.

Note that the s–t anypath computed by our algorithm has a length of $\max\{\frac{5.9}{1}, \frac{8.8}{1}\} = 8.8$, while as shown in Fig. 4.4h the optimal anypath has a length of $\max\{\frac{5.91}{1}, \frac{8.5}{1}\} = 8.5$. Although the anypath computed by Algorithm 2 is not optimal, its length is within a factor of 2 of that of the optimal anypath.

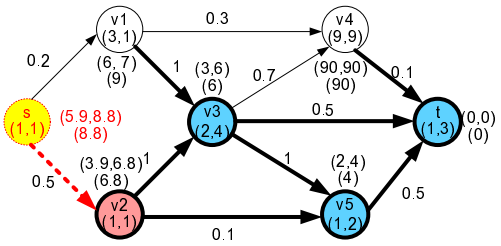


(a) t is inserted into L

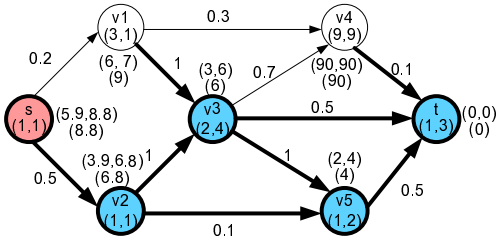
(b) v_5 is inserted into L



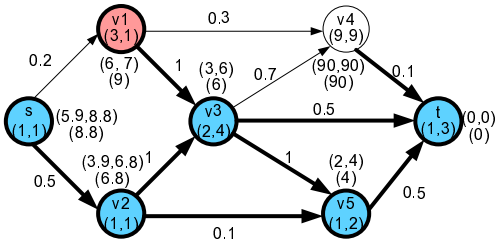
(c) v_3 is inserted into L



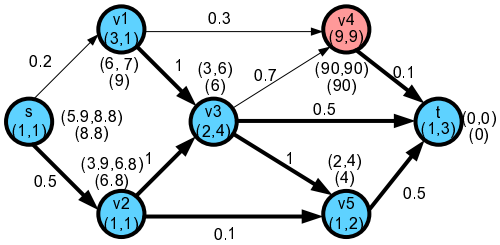
(d) v_2 is inserted into L



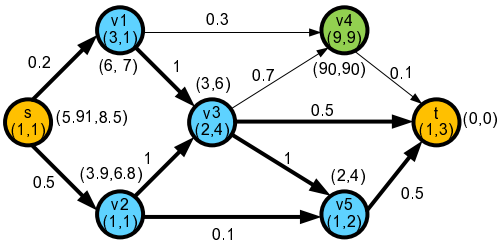
(e) s is inserted into L



(f) v_1 is inserted into L



(g) v_4 is inserted into L



(h) Optimal $s-t$ anypath

Figure 4.4: Execution of Algorithm 2 from every node to t. The link labels show link delivery probabilities; the vertex labels show (w_1, w_2) pairs of the vertices; the labels next to each vertex show the currently computed anypath weights $(\hat{W}_1, \hat{W}_2)$ (the first row) and the AAW $(\hat{W}_m)$ (the second row). (a)-(g) show the situations after each successive iteration of the algorithm. (h) shows the optimal $s-t$ anypath.

4.5.3 Algorithm Analysis

In this subsection, we analyze the performance of Algorithm 2. We use π_v^{opt} to denote an optimal anypath from v to t, and use π_v^m to denote the anypath computed by Algorithm 2. Note that when the algorithm terminates, $\mathcal{W}_m(v, \pi_v^m) = \hat{W}_m(v)$, and $\mathcal{W}_k(v, \pi_v^m) = \hat{W}_k(v), \forall k \in [1, K]$.

Theorem 8. *Let v be any node that can reach the destination t. The anypath π_v^m computed by Algorithm 2 is a shortest AAW anypath. This computation is done in $O(|V| \log |V| + K|E|)$ time.* □

Although the rigorous proof of Theorem 8 is very intricate, this complexity is an aspect of the analysis, not of the algorithm itself. As we have seen before, our algorithm is as simple as Dijkstra's algorithm for computing a shortest path.

In the following proofs, we omit the notation P of the anypath for simplicity. Instead, we use $\mathcal{W}_m^{(J)}(v)$ to denote the AAW of v via the forwarding set J if all the nodes in J are using their shortest AAW anypaths. In addition, we use $\theta(v)$ to denote the AAW of v along π_v^m(i.e. the optimal AAW of v). Before proving Theorem 8, we need the following four lemmas.

Lemma 6. *Consider a forwarding set $J = <j_1, j_2, \ldots, j_y>$ of a node v with $\mathcal{W}_m(j_1) \leq \mathcal{W}_m(j_2) \leq \cdots \leq \mathcal{W}_m(j_y)$ and a positive integer $\Delta \leq y$. We can arbitrarily divide J into Δ subsets of nodes with contiguous priorities: $<J_1, J_2, \ldots, J_\Delta> = <<j_{\tau_1}, j_{\tau_1+1}, \ldots, j_{\tau_2-1}>, <j_{\tau_2}, j_{\tau_2+1}, \ldots, j_{\tau_3-1}>, \ldots, <j_{\tau_\Delta}, j_{\tau_\Delta+1}, \ldots, j_{\tau_{\Delta+1}-1}>>$, where $j_{\tau_1} = j_1,\ j_{\tau_1+1} = j_2, \ldots, j_{\tau_{\Delta+1}-1} = j_y$. For instance, let us consider $\Delta = 2$. $<j_1, j_2, j_3, j_4, j_5>$ can be divided to $<<j_1>, <j_2, j_3, j_4, j_5>>, <<j_1, j_2>, <j_3, j_4, j_5>>,$*

$<<j_1, j_2, j_3>, <j_4, j_5>>$, *or* $<<j_1, j_2, j_3, j_4>, <j_5>>$. *We have*

$$\mathcal{W}_m^{(J)}(v) = \frac{\omega_m(v)}{1 - \prod_{\delta=1}^{\Delta}(1 - p(v, J_\delta))} + \sum_{J_\delta \in J} \frac{p(v, J_\delta) \prod_{q=1}^{\delta-1}(1 - p(v, J_q))}{1 - \prod_{\delta=1}^{\Delta}(1 - p(v, J_\delta))} \mathcal{W}_m(J_\delta).$$

We call these $J_1, J_2, \ldots, J_\Delta$ contiguous priority forwarding subsets. □

Proof. By (4.1), (4.25) and (4.26),

$$\begin{aligned}
&\mathcal{W}_m^{(J)}(v) \\
&= \frac{\omega_m(v)}{p(v, J)} + \sum_{j_\beta \in J} \frac{p(v, j_\beta) \prod_{q=1}^{\beta-1}(1 - p(v, j_q)) \mathcal{W}_m(j_q)}{p(v, J)} \\
&= \frac{\omega_m(v)}{1 - \prod_{\delta=1}^{\Delta}(1 - p(v, J_\delta))} + \sum_{\delta=1}^{\Delta} \frac{p(v, J_\delta) \mathcal{W}_m(J_\delta) \prod_{q=1}^{\tau_\delta - 1}(1 - p(v, j_q))}{1 - \prod_{\delta=1}^{\Delta}(1 - p(v, J_\delta))} \\
&= \frac{\omega_m(v)}{1 - \prod_{\delta=1}^{\Delta}(1 - p(v, J_\delta))} + \sum_{\delta=1}^{\Delta} \frac{p(v, J_\delta) \prod_{q=1}^{\delta-1}(1 - p(v, J_q))}{1 - \prod_{\delta=1}^{\Delta}(1 - p(v, J_\delta))} \mathcal{W}_m(J_\delta).
\end{aligned}$$

■

Intuitively, this lemma implies that we can group together the nodes with contiguous priorities as an "*aggregated vertex*". In other words, each contiguous priority forwarding subset is an aggregated vertex. For instance, when $\Delta = 3$, $<j_1, j_2, j_3, j_4, j_5>$ can form three aggregated vertices:$<j_1, j_2>$, $<j_3>$, $<j_4, j_5>$. Note that an aggregated vertex can consist of a single node or a set of nodes with contiguous priorities. Thus if $J_1, J_2, \ldots, J_\beta, \ldots$ are contiguous priority forwarding subsets, we can rewrite (4.25) and (4.26) as

$$\mathcal{W}_m(v, P_s) = \frac{\omega_m(v)}{p(v, J(v, P_s))} + \mathcal{W}_m(J(v, P_s), P_s), \tag{4.27}$$

where

$$\mathcal{W}_m(J(v, P_s), P_s) = \sum_{J_\beta \in J(v, P_s)} \alpha(J_\beta, P_s) \mathcal{W}_m(J_\beta, P_s), \tag{4.28}$$

and

$$\alpha(J_\beta, P_s) = \frac{p(v, J_\beta) \prod_{q=1}^{\beta-1}(1 - p(v, J_q))}{p(v, J(v, P_s))}. \tag{4.29}$$

Note that β can be considered the forwarding priority of this aggregated vertex, and that the boundary condition is $\mathcal{W}_m(t, P_s) = 0$.

Lemma 7. *For all the neighbors $j_1, j_2, \ldots, j_z$ of a node v, where z is the number of neighbors, if $\theta(j_1) \leq \theta(j_2) \leq \cdots \leq \theta(j_z)$, then there must exist an AOFS of the form $<j_1, j_2, \ldots, j_b>$ for some $b \in \{1, 2, \ldots, z\}$, called a* full AAW optimal forwarding set (FAOFS). □

Proof. We index all the neighbors of v in increasing order of their optimal AAWs. In other words, a node j with a larger $\theta(j)$ should have a larger *priority index*. Now each neighbor has a unique index which will be used in this proof. For a single metric, Dubois-Ferrière [25] indicates that a lower priority should be assigned to a forwarder with a larger anypath weight. Thus, a forwarder with a larger priority index (i.e. a larger AAW) should have a lower priority. Since each node has a finite number of neighbors, there must exist at least one AOFS for each node. Among all the AOFSs of v, we compare the priority indices of their lowest priority nodes, and select the AOFS whose lowest priority node has the smallest priority index. We use j_b to denote this node. If there are multiple such sets, we arbitrarily select one (denoted by ψ). For instance, there are three AOFSs:$<j_1, j_2, j_3>$, $<j_1, j_2, j_4>$ and $<j_1, j_3, j_5>$. The lowest priority node in $<j_1, j_2, j_3>$ has the smallest priority index. Thus $b = 3$. We call the nodes with priority indices smaller than b *potential AAW optimal forwarders* (PAOFs). In the example above, j_1 and j_2 are PAOFs.

Based on ψ, we will construct an FAOFS which is composed of all the PAOFs and j_b. The outline of our proof is as follows. We will prove that inserting an arbitrary PAOF into the forwarding set, whose lowest priority node is j_b, will not increase the AAW of v. This implies that if a PAOF is not in ψ, we can add it into ψ without increasing the AAW of v. Repeat this procedure

until the forwarding set contains all the PAOFs and j_b. Take $b = 5$ as an example. Assume the AOFS ψ is of the form $<j_1, j_3, j_5>$. We will prove that the AAW of v via $\psi \cup \{j_2\}$ or $\psi \cup \{j_4\}$ is not larger than that via ψ. After j_2(or j_4) is added into the forwarding set, we will prove that the AAW of v via $\psi \cup \{j_2\} \cup \{j_4\}$ is not larger than that via $\psi \cup \{j_2\}$(or $\psi \cup \{j_4\}$). Clearly, the forwarding set $<j_1, j_2, j_3, j_4, j_5>$ is not worse than ψ, thus it is also an AOFS. The term *hole* is used to represent the PAOFs which are not in the forwarding set ψ. For instance, j_2 and j_4 in the example above are holes. *From now on, we deal with the case that ψ has at least one hole, since for the case ψ has no hole, ψ is naturally an FAOFS.* Among all the PAOFs, we arbitrarily select one subset of PAOFs, which has one hole (denoted by j_h). We use S_h to denote the union of this selected set and j_b. Then taking this hole j_h as a *pivot*, we divide this set into two subsets S_l and S_r such that the optimal AAW of nodes in S_l (S_r) are not greater (less) than $\theta(j_h)$. Considering the example above, we select a set of PAOFs $<j_1, j_2, j_3>$ and j_2 is a hole. Then S_h is $<j_1, j_2, j_3, j_5>$. Taking j_2 as a pivot, we can obtain two subsets $S_l = <j_1>$ and $S_r = <j_3, j_5>$. Note that S_l could be empty since the hole could be the highest priority node in S_h, while S_r is always nonempty since it contains at least j_b. Now we prove some properties of S_l, S_r and PAOFs.

First, for a nonempty set S_r, we know $\theta(j_h) \leq \theta(j_{r_i}), \forall j_{r_i} \in S_r, i \in [1, |S_r|]$. Thus, if each node in S_r uses its shortest AAW anypath, by (4.26) the AAW of S_r can be calculated as follows:

$$\begin{aligned}\mathcal{W}_m(S_r) &= \sum_{j_{r_i} \in S_r} \frac{p(v, j_{r_i}) \prod_{q=1}^{i-1}(1 - p(v, j_{r_q}))}{p(v, S_r)} \theta(j_{r_i}) \\ &\geq \theta(j_h) \sum_{j_{r_i} \in S_r} \frac{p(v, j_{r_i}) \prod_{q=1}^{i-1}(1 - p(v, j_{r_q}))}{1 - \prod_{i=1}^{|S_r|}(1 - p(v, j_{r_i}))} = \theta(j_h). \end{aligned} \tag{4.30}$$

Second, for any nonempty set S composed of PAOFs, we claim that if all

the nodes in S use their shortest AAW anypaths, the AAW of v via this forwarding set S satisfies

$$\mathcal{W}_m^{(S)}(v) = \frac{\omega_m(v)}{p(v,S)} + \mathcal{W}_m(S) > \theta(j_b). \tag{4.31}$$

We will prove this claim by contradiction. Assume that there exists a set S'_a such that $\frac{\omega_m(v)}{p(v,S'_a)} + \mathcal{W}_m(S'_a) \leq \theta(j_b)$. We use ψ' to denote the set $\psi \setminus \{j_b\}$. Since S'_a is composed of PAOFs, it does not contain j_b. Let S_a denote the set $S'_a \cup \{j_b\}$ (i.e. $S'_a = S_a \setminus \{j_b\}$). Since ψ is optimal with respect to AAW, we have

$$\begin{aligned}&\frac{\omega_m(v)+p(v,S'_a)\mathcal{W}_m(S'_a)+(1-p(v,S'_a))p(v,j_b)\theta(j_b)}{p(v,S_a)} \geq \\ &\frac{\omega_m(v)+p(v,\psi')\mathcal{W}_m(\psi')+(1-p(v,\psi'))p(v,j_b)\theta(j_b)}{p(v,\psi)}.\end{aligned} \tag{4.32}$$

By (4.27)(4.28)(4.29), the left side of (4.32) is the AAW of v via the forwarding set S_a, and the right side is that of v via the AOFS ψ. We can transform (4.32) to

$$\begin{aligned}&(\omega_m(v)+p(v,\psi')\mathcal{W}_m(\psi'))(1-(1-p(v,S'_a))(1-p(v,j_b))) \\ -&(\omega_m(v)+p(v,S'_a)\mathcal{W}_m(S'_a))(1-(1-p(v,\psi'))(1-p(v,j_b))) \\ +&p(v,j_b)\theta(j_b)(p(v,S'_a)-p(v,\psi')) \leq 0.\end{aligned} \tag{4.33}$$

According to the assumption $\frac{\omega_m(v)}{p(v,S'_a)} + \mathcal{W}_m(S'_a) \leq \theta(j_b)$ and $p(v,j_b) > 0$ (by the definition of OMCAP), (4.33) implies

$$\begin{aligned}&(\omega_m(v)+p(v,\psi')\mathcal{W}_m(\psi'))(1-(1-p(v,S'_a))(1-p(v,j_b))) \\ &\leq \theta(j_b)p(v,\psi')(1-(1-p(v,S'_a))(1-p(v,j_b))).\end{aligned} \tag{4.34}$$

Since $p(v,j_b) > 0$, we know $1-(1-p(v,S'_a))(1-p(v,j_b)) > 0$. Thus, we can divide both sides of (4.34) by $1-(1-p(v,S'_a))(1-p(v,j_b))$ and derive the AAW of v via ψ'

$$\mathcal{W}_m^{(\psi')}(v) = \frac{\omega_m(v)}{p(v,\psi')} + \mathcal{W}_m(\psi') \leq \theta(j_b). \tag{4.35}$$

We can now prove (4.31) by contradiction. When v uses ψ as its forwarding set, by (4.27-4.29), we know

$$\mathcal{W}_m^{(\psi)}(v) = \frac{\omega_m(v)+p(v,\psi')\mathcal{W}_m(\psi')+p(v,j_b)(1-p(v,\psi'))\theta(j_b)}{1-(1-p(v,j_b))(1-p(v,\psi'))}. \tag{4.36}$$

When v uses ψ' as its forwarding set, its AAW is

$$\mathcal{W}_m^{(\psi')}(v) = \frac{\omega_m(v)+p(v,\psi')\mathcal{W}_m(\psi')}{p(v,\psi')}. \tag{4.37}$$

Combining (4.36) and (4.37), we have

$$\mathcal{W}_m^{(\psi)}(v) = \frac{\mathcal{W}_m^{(\psi')}(v)p(v,\psi')+p(v,j_b)(1-p(v,\psi'))\theta(j_b)}{1-(1-p(v,j_b))(1-p(v,\psi'))}. \tag{4.38}$$

According to (4.35) and (4.38), we can derive $\mathcal{W}_m^{(\psi)}(v) \geq \mathcal{W}_m^{(\psi')}(v)$. This is a contradiction considering the following two cases. In the first case: $\mathcal{W}_m^{(\psi)}(v) > \mathcal{W}_m^{(\psi')}(v)$. This implies ψ' is better and thus ψ is not an AOFS, which is a contradiction. In the second case: $\mathcal{W}_m^{(\psi)}(v) = \mathcal{W}_m^{(\psi')}(v)$. We can therefore safely remove j_b from ψ to obtain another AOFS ψ', whose lowest priority node has a smaller priority index than that of j_b. Recall that ψ is the AOFS whose lowest priority node j_b has the smallest priority index. This is also a contradiction. Therefore, the inequality (4.31) is proved.

Third, since S is an arbitrary set composed of PAOFs, according to (4.31), we know that when S_l is not empty (recall that S_l is composed of PAOFs), $\mathcal{W}_m^{(S_l)}(v) = \frac{\omega_m(v)}{p(v,S_l)} + \mathcal{W}_m(S_l) > \theta(j_b)$. We know that $\theta(j_b)$ is not smaller than the optimal AAW of any PAOF and that j_h is a PAOF. Thus

$$\mathcal{W}_m^{(S_l)}(v) = \frac{\omega_m(v)}{p(v,S_l)} + \mathcal{W}_m(S_l) > \theta(j_h). \tag{4.39}$$

Based on the three properties above, we will show that we can safely fill the hole j_h. We now compare the AAW of v via S_h with that via $S_h' = S_h \setminus \{j_h\}$. By

(4.27)(4.28)(4.29), we have

$$\mathcal{W}_m^{(S'_h)}(v) = \frac{\omega_m(v)+p(v,S_l)\mathcal{W}_m(S_l)+(1-p(v,S_l))p(v,S_r)\mathcal{W}_m(S_r)}{1-(1-p(v,S_l))(1-p(v,S_r))},$$

and

$$\begin{aligned}
&\mathcal{W}_m^{(S_h)}(v)\\
&= \frac{\omega_m(v)+p(v,S_l)\mathcal{W}_m(S_l)+(1-p(v,S_l))p(v,j_h)\theta(j_h)}{1-(1-p(v,S_l))(1-p(v,S_r))(1-p(v,j_h))}\\
&+\frac{(1-p(v,S_l))(1-p(v,j_h))p(v,S_r)\mathcal{W}_m(S_r)}{1-(1-p(v,S_l))(1-p(v,S_r))(1-p(v,j_h))}.
\end{aligned}$$

Since $p(v,S'_h) = 1-(1-p(v,S_l))(1-p(v,S_r))$ and $p(v,S_h) = 1-(1-p(v,S_l))(1-p(v,S_r))(1-p(v,j_h))$, we can have

$$\begin{aligned}
&\mathcal{W}_m^{(S_h)}(v) - \mathcal{W}_m^{(S'_h)}(v) =\\
&= \frac{-(1-p(v,S_l))p(v,j_h)(\omega_m(v)+p(v,S_l)\mathcal{W}_m(S_l))(1-p(v,S_r))}{p(v,S_h)p(v,S'_h)}\\
&-\frac{(1-p(v,S_l))p(v,j_h)p(v,S_r)\mathcal{W}_m(S_r)}{p(v,S_h)p(v,S'_h)}\\
&+\frac{(1-p(v,S_l))p(v,j_h)\theta(j_h)(p(v,S_l)+p(v,S_r)-p(v,S_l)p(v,S_r))}{p(v,S_h)p(v,S'_h)}. \qquad (4.40)
\end{aligned}$$

We now consider two cases. The first case is that S_l is empty. Obviously, we can set $p(v,S_l) = p(v,S_l)\mathcal{W}_m(S_l) = 0$. Recall that S_r is always nonempty. Thus, we can substitute (4.30) and $p(v,S_l) = 0$ into (4.40). We must have

$$\mathcal{W}_m^{(S_h)}(v) - \mathcal{W}_m^{(S'_h)}(v) \le \frac{-p(v,j_h)\omega_m(v)(1-p(v,S_r))}{p(v,S_h)p(v,S'_h)} \le 0. \qquad (4.41)$$

The second case is that both S_l and S_r are nonempty. Substituting (4.30) and (4.39) into (4.40), after some mathematical manipulation, we have

$$\mathcal{W}_m^{(S_h)}(v) - \mathcal{W}_m^{(S'_h)}(v) \le \frac{-(1-p(v,S_l))p(v,j_h)\omega_m(v)(1-p(v,S_r))}{p(v,S_h)p(v,S'_h)} \le 0. \quad (4.42)$$

Recall that S_h is the union of j_b and an *arbitrary* set composed of PAOFs. Inequalities (4.41) and (4.42) imply that if ψ has at least one hole, we can first

safely fill a hole in ψ without increasing the AAW of v. Then we can safely continue to fill another hole in the forwarding set obtained in the step above without increasing the AAW of v. The procedure is repeated until all the holes are filled. Now, we have a forwarding set, which is composed of j_b and all the PAOFs. Furthermore, the AAW of v via this forwarding set is not larger than that via ψ. Thus, we obtain an FAOFS. ■

Lemma 7 implies that if we want to find an AOFS, we only have to check forwarding sets $<j_1>$, $<j_1, j_2>$, ... Thus the complexity of the algorithm can be reduced to polynomial time from exponential time. We call the full AAW optimal forwarding set so constructed a *minimum full AAW optimal forwarding set* (MFAOFS). Recall that the AAW of the lowest priority node in the MFAOFS is not greater than that of the lower priority node in any other AOFS. Therefore, if we check the forwarding sets in the order $<j_1>$, $<j_1, j_2>$, ..., the MFAOFS would be found earliest among all the AOFSs. If all the nodes on a shortest AAW anypath from v to t are using their MFAOFSs, we call it a *full shortest AAW anypath*.

Lemma 8. *The MFAOFS of v is $<j_1, j_2, \ldots, j_b>$ with optimal AAWs $\theta(j_1) \leq \theta(j_2) \leq \ldots \leq \theta(j_b)$. We use S_μ to denote the forwarding set $<j_1, j_2, \ldots, j_\mu>, 1 \leq \mu \leq b$. Then we have $\mathcal{W}_m^{(S_1)}(v) > \mathcal{W}_m^{(S_2)}(v) > \ldots > \mathcal{W}_m^{(S_b)}(v) = \theta(v)$.* □

Proof. Consider an arbitrary integer μ in the range of $[1, b-1]$. By the definition of AAW, we have

$$\mathcal{W}_m^{(S_\mu)}(v) = \frac{\omega_m(v) + p(v, S_\mu)\mathcal{W}_m(S_\mu)}{p(v, S_\mu)}.$$

By (4.27-4.29), we have

$$\mathcal{W}_m^{(S_{\mu+1})}(v) = \frac{\omega_m(v) + p(v, S_\mu)\mathcal{W}_m(S_\mu) + p(v, j_{\mu+1})(1 - p(v, S_\mu))\theta(j_{\mu+1})}{1 - (1 - p(v, S_\mu))(1 - p(v, j_{\mu+1}))}.$$

Combining these two equalities, we have

$$\mathcal{W}_m^{(S_{\mu+1})}(v) = \frac{\mathcal{W}_m^{(S_\mu)}(v)p(v,S_\mu)+p(v,j_{\mu+1})(1-p(v,S_\mu))\theta(j_{\mu+1})}{1-(1-p(v,S_\mu))(1-p(v,j_{\mu+1}))}. \tag{4.43}$$

Since the nonempty forwarding set S_μ is composed of PAOFs, by (4.31) we have $\mathcal{W}_m^{(S_\mu)}(v) > \theta(j_b)$. According to $\theta(j_b) \geq \theta(j_{\mu+1})$, we have $\mathcal{W}_m^{(S_\mu)}(v) > \theta(j_{\mu+1})$. In addition, we claim $p(v,S_\mu) < 1$. Otherwise, according to the definition of AAW, if $p(v,S_\mu) = 1$, j_b has no chance to forward a received packet. Thus j_b can be safely removed. This contradicts the fact that $<j_1, j_2, \ldots, j_b>$ is an MFAOFS. Substituting $p(v,S_\mu) < 1$, $p(v,j_{\mu+1}) > 0$ (by the definition of OMCAP) and $\mathcal{W}_m^{(S_\mu)}(v) > \theta(j_{\mu+1})$ into (4.43), we have

$$\mathcal{W}_m^{(S_{\mu+1})}(v) < \frac{p(v,S_\mu)+p(v,j_{\mu+1})(1-p(v,S_\mu))}{1-(1-p(v,S_\mu))(1-p(v,j_{\mu+1}))}\mathcal{W}_m^{(S_\mu)}(v) = \mathcal{W}_m^{(S_\mu)}(v). \tag{4.44}$$

Since μ is an arbitrary integer in the range of $[1, b-1]$, we have $\mathcal{W}_m^{(S_1)}(v) > \mathcal{W}_m^{(S_2)}(v) > \cdots > \mathcal{W}_m^{(S_b)}(v) = \theta(v)$. ■

Lemma 9. *The optimal AAW $\theta(v)$ of a node v is always larger than the optimal AAW $\theta(j)$ of any node j (other than v) on the full shortest AAW anypath of v.* □

Proof. First, we consider an arbitrary node j in v's MFAOFS ψ. We use ψ' to denote the set $\psi \setminus \{j_b\}$ and thus ψ' is composed of PAOFs. According to (4.31), we have $\mathcal{W}_m^{(\psi')}(v) > \theta(j_b)$. By (4.38) and $p(v,\psi') > 0$, we have

$$\begin{aligned} & \mathcal{W}_m^{(\psi)}(v) \\ = & \frac{\mathcal{W}_m^{(\psi')}(v)p(v,\psi') + p(v,j_b)(1-p(v,\psi'))\theta(j_b)}{1-(1-p(v,j_b))(1-p(v,\psi'))} \\ > & \frac{p(v,\psi') + p(v,j_b)(1-p(v,\psi')}{1-(1-p(v,j_b))(1-p(v,\psi'))}\theta(j_b) = \theta(j_b). \end{aligned}$$

Since j_b is the node with the largest AAW in ψ, we must have $\theta(v) = \mathcal{W}_m^{(\psi)}(v) > \theta(j)$ for every node j in ψ.

We can inductively apply the same proof to all the nodes in ψ. Repeating this procedure, eventually we will have the conclusion that the optimal AAW $\theta(v)$ of a node v is always larger than the optimal AAW $\theta(j)$ of any node j (other than v) on the full shortest AAW anypath of v. ■

Proof of Theorem 8. We show that for each node $v \in V$, when it is inserted into L, we have $\mathcal{W}_m(v) = \theta(v)$.

For the purpose of contradiction, let u be the first node added to L for which $\mathcal{W}_m(u) > \theta(u)$. We claim that *the MFAOFS of u contains at least one node, which has not been added into L.* Otherwise if all the nodes in its MFAOFS have been added into L, Algorithm 2 must have found this MFAOFS for u for the following reason. By Lemma 7, we know that there exists an MFAOFS of the form $<j_1, j_2, \ldots, j_b>$ for u. Moreover, u is the first node added to L for which $\mathcal{W}_m(u) > \theta(u)$. This implies that all the nodes already in its MFAOFS were inserted into L in the increasing order of their optimal AAWs, since they satisfied $\mathcal{W}_m(u) = \theta(u)$ when they were added into L, and Algorithm 2 always chooses the node with the smallest $\mathcal{W}_m(u)$ and adds it into L. Thus $<j_1>$, $<j_1, j_2>\ldots$ have been checked by Algorithm 2. By Lemma 8, if j_1 and j_2 are in the MFAOFS of u, using $<j_1, j_2>$ always provides a smaller AAW than just using $<j_1>$. Thus the condition in Line 6 of Algorithm 2 will be true and the forwarding set is updated to $<j_1, j_2>$. The same procedure is repeated until the MFAOFS is found. Note that by Lemma 7 the MFAOFS will be found earliest. This implies that when u is added into L, it is using its MFAOFS. Furthermore, the AAWs of the nodes in the MFAOFS of u are equal to their optimal values since u is the first one for which $\mathcal{W}_m(u) > \theta(u)$. We therefore know $\mathcal{W}_m(u) = \theta(u)$. However, this contradicts $\mathcal{W}_m(u) > \theta(u)$. Hence, the MFAOFS of u contains at least one node, which has not been added into L.

We arbitrarily select one of these nodes, denoted by u_1. By Lemma 9, we have $\theta(u_1) < \theta(u)$. Since we assume $\mathcal{W}_m(u) > \theta(u)$, we must have $\theta(u_1) < \mathcal{W}_m(u)$. Let us consider a full shortest AAW anypath $\pi_{u_1}^m$ from u_1 to t. Without loss of generality, assume that node u_2 has the smallest optimal AAW to t among all nodes on $\pi_{u_1}^m$ which have not been inserted into L. Thus $\theta(u_1) \geq \theta(u_2)$. **Claim** (1): *all the nodes in u_2's MFAOFS must be in L.* To prove this claim, let us assume that u_3, which is in u_2's MFAOFS, is not in L. By Lemma 9, we know that in this case we must have $\theta(u_2) > \theta(u_3)$. However, since we assume that u_2 has the smallest optimal AAW (note that u_3 is also on $\pi_{u_1}^m$), then $\theta(u_2) \leq \theta(u_3)$, which contradicts $\theta(u_2) > \theta(u_3)$. Thus, all the nodes in u_2's MFAOFS are in L. Since u is also in L, we next consider whether u is u_2's MFAOFS.

Claim (2): *u is not in u_2's MFAOFS.* We also prove this claim by contradiction. Assume that u is in u_2's MFAOFS. By Lemma 9, we therefore know $\theta(u_2) > \theta(u)$. On the other hand, since we have deduced that $\theta(u_2) \leq \theta(u_1)$ and $\theta(u_1) < \theta(u)$, we know $\theta(u_2) < \theta(u)$, which contradicts $\theta(u_2) > \theta(u)$. Thus u in not in u_2's MFAOFS.

Claims (1) and (2) imply that at the time just before u is inserted into L, all the nodes in the MFAOFS of u_2 have been inserted into L. Recall how we proved that if all the nodes in the MFAOFS of u have been added into L, this set must be found by Algorithm 2. We can use the same proof to prove that the MFAOFS of u_2 must have been found by Algorithm 2. Thus, at that time $\mathcal{W}_m(u_2) = \theta(u_2)$.

We now derive the contradiction based on our first assumption that u is the first node added to L for which $\mathcal{W}_m(u) > \theta(u)$. Since we have deduced $\theta(u_2) \leq \theta(u_1)$ and $\theta(u_1) < \mathcal{W}_m(u)$, we have $\theta(u_2) < \mathcal{W}_m(u)$. Additionally, we deduced $\mathcal{W}_m(u_2) = \theta(u_2)$. We therefore know $\mathcal{W}_m(u_2) < \mathcal{W}_m(u)$. However this is

a contradiction, since this inequality implies that u_2, which is a node outside of L at the time u is inserted into L, should be inserted into L before u.

We therefore conclude that for each node v in L, we have $\mathcal{W}_m(v) = \theta(v)$. This implies that when a node is inserted into L, its shortest AAW anypath has been found and will be returned by Algorithm 2.

We now analyze the running time of Algorithm 2. Lines 1-4 take $O(K|V|)$ to finish initializations. If we use a Fibonacci heap [21], each of the Extract-Min operations takes $O(\log|V|)$ in Line 6, with a total of $O(|V|\log|V|)$. Each of the max-operations in Line 7 takes $O(K)$, with a total of $O(K|V|)$. Note that the for-loop of Lines 8-16 is executed $O(|E|)$ times in total, since there are a total number of $|E|$ incoming links. If we use the technique proposed by [59], Lines 9 and 11 take constant time respectively. Thus each execution of the for-loop of Lines 10-12 takes $O(K)$, with a total of $O(K|E|)$. Evaluating the condition in Line 6 and the operations in Line 7 take $O(|E|)$ in total, respectively. Updating Q requires $O(|E|)$ in total if a Fibonacci heap is used. Thus the total running time is $O(|V|\log|V| + K|E|)$. ■

Theorem 8 states that the anypath so computed is a shortest AAW anypath. In other words, the forwarding set of each node computed by Algorithm 2 is actually an optimal forwarding set with respect to AAW, namely *AAW optimal forwarding set* (**AOFS**).

The next theorem is the main result of this work. It characterizes how good Algorithm 2 is, compared to the optimal solution to OMCAP$(G, s, t, \overrightarrow{\mathbb{W}}, \overrightarrow{w}, \overrightarrow{p}, K)$.

Theorem 9. *Let π_s^m be the anypath from s to t computed by Algorithm 2, and π_s^{opt} be any optimal solution to OMCAP$(G, s, t, \overrightarrow{\mathbb{W}}, \overrightarrow{w}, \overrightarrow{p}, K)$. We have*

$$l(s, \pi_s^m) \leq K \cdot l(s, \pi_s^{opt}). \tag{4.45}$$

In other words, Algorithm 2 is a K-approximation algorithm for OMCAP. □

Proof. For an arbitrary node v in the network, we use π_v^m to denote the anypath from v to t computed by Algorithm 2, and use π_v^{opt} to denote an optimal solution to OMCAP$(G, v, t, \overrightarrow{\mathbb{W}}, \overrightarrow{w}, \overrightarrow{p}, K)$. To prove the truth of the theorem, it suffices to prove that the following three inequalities hold for every node v in the network.

$$l(v, \pi_v^m) \leq \mathcal{W}_m(v, \pi_v^m), \tag{4.46}$$

$$\mathcal{W}_m(v, \pi_v^m) \leq \mathcal{W}_m(v, \pi_v^{opt}), \tag{4.47}$$

$$\mathcal{W}_m(v, \pi_v^{opt}) \leq K \cdot l(v, \pi_v^{opt}). \tag{4.48}$$

It follows from Theorem 8 that π_v^m is a shortest AAW anypath from v to t. **Therefore we have Inequality (4.47)**.

Next we will prove Inequality (4.46). According to Algorithm 2, nodes are inserted into L one by one. We index all the nodes on π_v^m in this order. Therefore the index of node t is 0, and v has the largest index (at the time v is inserted into L). We use v_i to denote the vertex with index i, and **claim** that for each v_i, we have

$$l(v_i, \pi_v^m) \leq \mathcal{W}_m(v_i, \pi_v^m). \tag{4.49}$$

We will use mathematical induction to prove Inequality (4.49). For node v_1, only the destination t is in its forwarding set. Therefore we have

$$\mathcal{W}_m(v_1, \pi_v^m) = \frac{\omega_m(v_1)}{p(v_1, t)} + \mathcal{W}_m(<t>, \pi_v^m) = \frac{\omega_m(v_1)}{p(v_1, t)} = \max_{1 \leq k \leq K} \frac{w_k(v_1)}{p(v_1, t)}.$$

On the other hand, by the definition of anypath length, we have

$$l(v_1, \pi_v^m) = \max_{1 \leq k \leq K} \mathcal{W}_k(v_1, \pi_v^m) = \max_{1 \leq k \leq K} \frac{w_k(v_1)}{p(v_1, t)} + \mathcal{W}_k(<t>, \pi_v^m) = \max_{1 \leq k \leq K} \frac{w_k(v_1)}{p(v_1, t)}.$$

Therefore we have $l(v_1, \pi_v^m) \leq \mathcal{W}_m(v_1, \pi_v^m)$.

Assume that the claim (4.49) is true for all the nodes with indices in the range $[1, \eta]$. We will prove that the claim is still true for $v_{\eta+1}$. By the definition of $\mathcal{W}_m$, we have

$$\mathcal{W}_m(v_{\eta+1}, \pi_v^m) = \frac{\omega_m(v_{\eta+1})}{p(v_{\eta+1}, J(v_{\eta+1}, \pi_v^m))} + \sum_{j_\beta \in J(v_{\eta+1}, \pi_v^m)} \alpha(j_\beta, \pi_v^m) \mathcal{W}_m(j_\beta, \pi_v^m). \quad (4.50)$$

According to Algorithm 2, before $v_{\eta+1}$ is inserted into L, all its forwarders along π_v^m have been inserted into L. Thus, the assumption that the claim (4.49) is true for all the nodes with indices in the range $[1, \eta]$ can be applied to all its forwarders. Therefore we have

$$\mathcal{W}_m(j_\beta, \pi_v^m) \geq l(j_\beta, \pi_v^m) = \max_{1 \leq k \leq K} \mathcal{W}_k(j_\beta, \pi_v^m), \forall j_\beta \in J(v_{\eta+1}, \pi_v^m).$$

Substituting the above inequality and the definition of $\omega_m(v_{\eta+1})$ into (4.50), we have

$$\begin{aligned}
& \mathcal{W}_m(v_{\eta+1}, \pi_v^m) \\
\geq & \max_{1 \leq k \leq K} \frac{w_k(v_{\eta+1})}{p(v_{\eta+1}, J(v_{\eta+1}, \pi_v^m))} \sum_{j_\beta \in J(v_{\eta+1}, \pi_v^m)} \alpha(j_\beta, \pi_v^m) \max_{1 \leq k \leq K} \mathcal{W}_k(j_\beta, \pi_v^m) \\
\geq & \max_{1 \leq k \leq K} \left(\frac{w_k(v_{\eta+1})}{p(v_{\eta+1}, J(v_{\eta+1}, \pi_v^m))} + \sum_{j_\beta \in J(v_{\eta+1}, \pi_v^m)} \alpha(j_\beta, \pi_v^m) \mathcal{W}_k(j_\beta, \pi_v^m) \right) \\
= & \max_{1 \leq k \leq K} \mathcal{W}_k(v_{\eta+1}, \pi_v^m) = l(v_{\eta+1}, \pi_v^m).
\end{aligned}$$

Therefore, we have proved claim (4.49), which implies Inequality (4.46).

Now we prove Inequality (4.48). Recall that the anypaths considered are directed acyclic graphs. We can index all the nodes on the optimal anypath π_v^{opt} of node v in a reversed topological order [21]. In other words, the index of t is 0, v has the largest index, and the index of every node is always greater than the indices of its forwarders. We use v_i to denote the vertex with index i, and **claim** that for any node v_i on π_v^{opt}, we have

$$\sum_{1 \leq k \leq K} \mathcal{W}_k(v_i, \pi_v^{opt}) \geq \mathcal{W}_m(v_i, \pi_v^{opt}). \quad (4.51)$$

We prove this claim using mathematical induction on i.

For v_1, since only t is in its forwarding set, we have

$$\begin{aligned}
&\sum_{1\le k\le K}\mathcal{W}_k(v_1,\pi_v^{opt})\\
&=\sum_{1\le k\le K}\frac{w_k(v_1)}{p(v_1,t)}+\mathcal{W}_k(<t>,\pi_v^{opt})=\sum_{1\le k\le K}\frac{w_k(v_1)}{p(v_1,t)}\\
&\ge\frac{1}{p(v_1,t)}\max_{1\le k\le K}w_k(v_1)=\frac{\omega_m(v_1)}{p(v_1,t)}=\mathcal{W}_m(v_1,\pi_v^{opt}).
\end{aligned}$$

Assume that the claim (4.51) is true for all the nodes with indices in the range $[1,\eta]$. We will prove that the claim is still true for $v_{\eta+1}$ as follows. For $v_{\eta+1}$, we have

$$\begin{aligned}
&\sum_{1\le k\le K}\mathcal{W}_k(v_{\eta+1},\pi_v^{opt})\\
&=\sum_{k=1}^{K}\left(\frac{w_k(v_{\eta+1})}{p(v_{\eta+1},J(v_{\eta+1},\pi_v^{opt}))}+\sum_{j_\beta\in J(v_{\eta+1},\pi_v^{opt})}\alpha(j_\beta,\pi_v^{opt})\mathcal{W}_k(j_\beta,\pi_v^{opt})\right)\\
&\ge\max_{1\le k\le K}\frac{w_k(v_{\eta+1})}{p(v_{\eta+1},J(v_{\eta+1},\pi_v^{opt}))}+\sum_{j_\beta\in J(v_{\eta+1},\pi_v^{opt})}\alpha(j_\beta,\pi_v^{opt})\sum_{1\le k\le K}\mathcal{W}_k(j_\beta,\pi_v^{opt}).
\end{aligned}$$

Since the indices of the forwarders of $v_{\eta+1}$ are smaller than that of $v_{\eta+1}$, the assumption that the claim (4.51) is true for all the nodes with indices in the range $[1,\eta]$ can be applied to all its forwarders, which means $\sum_{1\le k\le K}\mathcal{W}_k(j_\beta,\pi_v^{opt})\ge\mathcal{W}_m(j_\beta,\pi_v^{opt}),\forall j_\beta\in J(v_{\eta+1},\pi_v^{opt})$. Hence, we have

$$\begin{aligned}
&\sum_{1\le k\le K}\mathcal{W}_k(v_{\eta+1},\pi_v^{opt})\\
&\ge\max_{1\le k\le K}\frac{w_k(v_{\eta+1})}{p(v_{\eta+1},J(v_{\eta+1},\pi_v^{opt}))}+\sum_{j_\beta\in J(v_{\eta+1},\pi_v^{opt})}\alpha(j_\beta,\pi_v^{opt})\mathcal{W}_m(j_\beta,\pi_v^{opt})\\
&=\frac{\omega_m(v_{\eta+1})}{p(v_{\eta+1},J(v_{\eta+1},\pi_v^{opt}))}+\sum_{j_\beta\in J(v_{\eta+1},\pi_v^{opt})}\alpha(j_\beta,\pi_v^{opt})\mathcal{W}_m(j_\beta,\pi_v^{opt})\\
&=\mathcal{W}_m(v_{\eta+1},\pi_v^{opt}).
\end{aligned}$$

Therefore we have proved Claim (4.51), which implies the following.

$$\mathcal{W}_m(v, \pi_v^{opt}) \leq \sum_{1 \leq k \leq K} \mathcal{W}_k(v, \pi_v^{opt}). \tag{4.52}$$

According to (4.5), we have $\mathcal{W}_k(v, \pi_v^{opt}) \leq l(v, \pi_v^{opt})$, $\forall k \in [1, K]$. Therefore Inequality (4.52) implies the following.

$$\mathcal{W}_m(v, \pi_v^{opt}) \leq \sum_{1 \leq k \leq K} \mathcal{W}_k(v, \pi_v^{opt}) \leq K \cdot l(v, \pi_v^{opt}), \tag{4.53}$$

which implies Inequality (4.48).

Substituting v with s in Inequalities (4.46)-(4.48), we obtain Inequality (4.45). This completes the proof of the theorem. ■

Remark. For the case $K = 1$, our algorithm computes an *optimal solution* to the corresponding problem in polynomial time. When $K \geq 2$, the problem becomes NP-hard, and our algorithm computes a K-approximation solution in polynomial time.

4.6 An Efficient Distributed K-Approximation Algorithm

In this subsection we first present a distributed algorithm DMART to compute anypaths from all the nodes to a destination t and then analyze its performance. As before, in order to simplify the notations, we assume that $\mathbb{W}_1{=}\mathbb{W}_2{=}\cdots{=}\mathbb{W}_K{=}1$ in this subsection. This is equivalent to normalize all of the vertex weights (from $w_k(v)$ to $w_k(v)/\mathbb{W}_k$ for all $v \in V$ and $k = 1, 2, \ldots, K$). Therefore the algorithm and all the proofs can be straightforwardly extended to the case without this assumption.

4.6.1 Algorithm Description

DMART requires each node to first initialize the data structures and then execute a sequence of iterations. For every node $v \in V$ we use a variable $\mathcal{W}_m^{(i)}(v, P)$ to denote the AAW from v to t along anypath P computed in the i^{th} iteration. In addition, each node v keeps a data structure $Q(v)$ to store all its outgoing neighbors. At the beginning of the i^{th} iteration, each node v sorts the nodes in $Q(v)$ in the increasing order of their AAWs which are obtained in the $(i-1)^{th}$ iteration.

DMART's initialization phase for node v:

1: $\omega_m(v) \leftarrow \max_{1 \le k \le K} w_k(v)$, $Q(v) \leftarrow \emptyset$, $J(v,P) \leftarrow \emptyset$
2: **if** $v = t$ **then**
3: $\quad \mathcal{W}_m^{(0)}(v,P) \leftarrow 0$
4: $\quad$ **for** each k **do**
5: $\quad\quad \mathcal{W}_k(v,P) \leftarrow 0$
6: $\quad$ **end for**
7: **else**
8: $\quad \mathcal{W}_m^{(0)}(v,P) \leftarrow \infty$
9: $\quad$ **for** each k **do**
10: $\quad\quad \mathcal{W}_k(v,P) \leftarrow \infty$
11: $\quad$ **end for**
12: **end if**

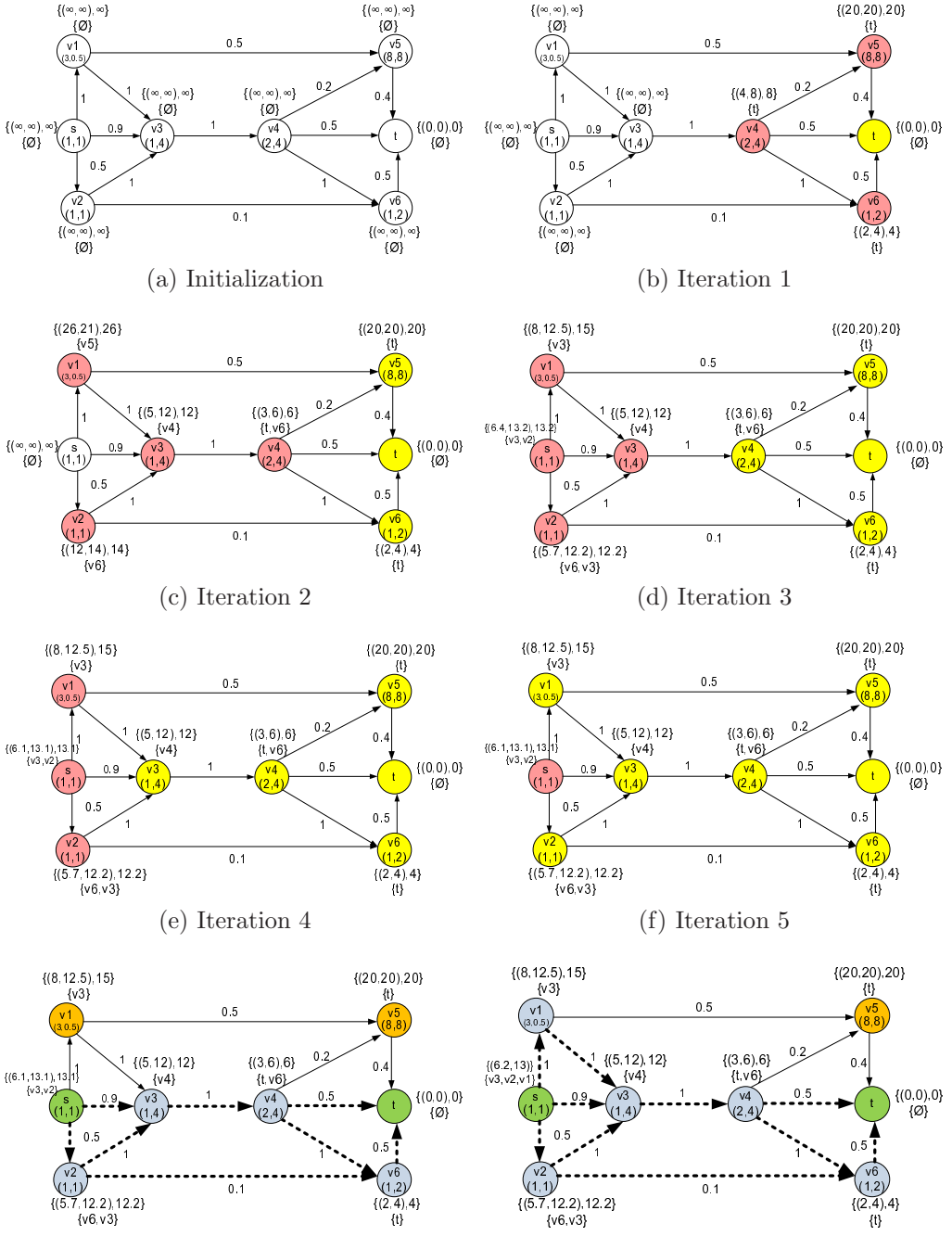
DMART's i^{th} iteration for node v:

1: Update $Q(v)$ such that all the elements in it (denoted by j) are sorted in the increasing order of $\mathcal{W}_m^{(i-1)}(j,P)$.
2: $F \leftarrow \emptyset$, $\mathcal{W}_m^{(i)}(v,P) \leftarrow \mathcal{W}_m^{(i-1)}(v,P)$
3: **while** $Q(v) \neq \emptyset$ **do**
4: $\quad j \leftarrow$ Extract-Min$(Q(v))$, $F \leftarrow F \cup \{j\}$
5: $\quad \mathcal{W}'_m(v,P) \leftarrow \frac{\omega_m(v)}{p(v,F)} + \sum_{j_\beta \in F} \alpha(j_\beta, P) \mathcal{W}_m^{(i-1)}(j_\beta, P)$
6: $\quad$ **if** $\mathcal{W}_m^{(i)}(v,P) > \mathcal{W}'_m(v,P)$ **then**
7: $\quad\quad J(v,P) \leftarrow F, \mathcal{W}_m^{(i)}(v,P) \leftarrow \mathcal{W}'_m(v,P)$
8: $\quad$ **end if**
9: **end while**
10: **for** each k **do**
11: $\quad \mathcal{W}_k(v,P) \leftarrow \frac{w_k(v)}{p(v,J(v,P))} + \mathcal{W}_k(J(v,P),P)$
12: **end for**

4.6.2 Algorithm Illustration

We now illustrate how DMART works using a simple example, shown in Fig. 4.5. In this example, $K = 2$ and $\mathbb{W}_1 = \mathbb{W}_2 = 1$.

In the initialization phase, each node calculates its auxiliary vertex weight and initializes the necessary data structures. $\mathcal{W}_m^{(0)}(t,P)$ is set to 0, and $\mathcal{W}_m^{(0)}(v,P) = \infty$ for all $v \in V \setminus \{t\}$ since so far no anypath has been found for them.



(a) Initialization (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

(e) Iteration 4 (f) Iteration 5

(g) Iteration 6 (s-t anypath) (h) Optimal $s-t$ anypath

Figure 4.5: Execution of DMART from every node to t. The link labels show link delivery probabilities; the vertex labels show (w_1, w_2) pairs of the vertices; the labels next to each vertex show the currently computed anypath weights $(\mathcal{W}_1, \mathcal{W}_2)$ (the first parenthesis in the first bracket), the AAW $(\mathcal{W}_m)$ (the second element in the first bracket), and the forwarding set (the second bracket). The original network graph is shown in Fig.4.5a. Fig. 4.5b-4.5g show the situations after each iteration. Fig. 4.5h shows an optimal s-t anypath.

At the beginning of the first iteration, v_4, v_5 and v_6 know that $\mathcal{W}_m^{(0)}(t,P) = 0$. Now we take v_4 as an example. $Q(v_4)$ has three elements t, v_5 and v_6. Note that $\mathcal{W}_m^{(0)}(v_5,P) = \mathcal{W}_m^{(0)}(v_6,P) = \infty$. Thus in the first loop (Lines 3-9), v_4 extracts t (Line 4), and computes $\mathcal{W}'_m(v,P) = 8$ (Line 5). Since $\mathcal{W}_m^{(1)}(v_4,P) = \infty > \mathcal{W}'_m(v,P) = 8$, Line 7 updates the AAW for v_4. In the second loop, F is updated to $\{t, v_5\}$. Obviously, $\mathcal{W}'_m(v,P) = \infty$, and thus Lines 6-8 will be skipped. In the third loop, F is updated to $\{t, v_5, v_6\}$. Likewise $\mathcal{W}'_m(v,P) = \infty$, and Lines 6-8 will be skipped. The result of the first iteration is shown in Fig.4.5b.

At the beginning of the second iteration, for example, v_1 knows $\mathcal{W}_m^{(1)}(v_3,P) = \infty$ and $\mathcal{W}_m^{(1)}(v_5,P) = 20$. Similar to the procedure above, after two loops (Lines 3-9) v_1 will select v_5 as its forwarder. The same procedure repeats until no updates happen any more.

We note that the s–t anypath obtained by DMART has a length of $13.1 = \max\{\frac{6.1}{1}, \frac{13.1}{1}\}$, while as shown in Fig. 4.5h the optimal one has a length of $13 = \max\{\frac{6.2}{1}, \frac{13}{1}\}$. Although the anypath so computed is not optimal, its length is within a factor of 2 of that of the optimal one.

4.6.3 Algorithm Analysis

Theorem 10. *DMART computes a K-approximation to the OMCAP problem after at most $\mathcal{D} \leq |V|$ iterations and each iteration can be done in $O(\Delta(\log\Delta + K))$ time by each node, where Δ is the max out-degree of graph G.* □

We now prove that the anypath computed by DMART is a K-approximation to the OMCAP problem. Let $\theta(v)$ denote v's optimal AAW. We use $\mathcal{W}_m^{(J)}(v)$ to denote v's AAW via the forwarding set J if all the nodes in J use their shortest AAW anypaths. Here we recall two lemmas which have been proved before.

Lemma 10. *For all of node v's neighbors $j_1, j_2, \ldots, j_z$, where z is the number of v's outdegree, if $\theta(j_1) \leq \theta(j_2) \leq \cdots \leq \theta(j_z)$, then there must exist an AAW optimal forwarding set (AOFS) of the form $\{j_1, j_2, \ldots, j_b\}$ for some $b \in \{1, 2, \ldots, z\}$, called a* full AAW optimal forwarding set (FAOFS). □

Lemma 10 implies that if we want to find an AOFS, we only need to check forwarding sets $\{j_1\}$, $\{j_1, j_2\}$, ... The complexity of the algorithm can therefore be reduced to polynomial time from exponential time. Recall that the proof of this lemma constructs a full AAW optimal forwarding set from a particular forwarding set. This constructed forwarding set is called a *minimum full AAW optimal forwarding set* (MFAOFS). When all the nodes on a shortest AAW anypath from v to t are using their MFAOFSs, we call it a *full shortest AAW anypath.*

Lemma 11. *The MFAOFS of v is $\{j_1, j_2, ..., j_b\}$ with optimal AAWs $\theta(j_1) \leq \theta(j_2) \leq \cdots \leq \theta(j_b)$. We use S_μ to denote the forwarding set $\{j_1, j_2, ..., j_\mu\}, 1 \leq \mu \leq b$. Then we have $\mathcal{W}_m^{(S_1)}(v) > \mathcal{W}_m^{(S_2)}(v) \cdots > \mathcal{W}_m^{(S_b)}(v) = \theta(v)$.* □

Now we define a *distance index* for each node v. Since the full shortest AAW anypath from v to the destination t is an acyclic directed graph (recall that we only consider the case in which anypaths are acyclic), we can use the topological order [21] to index all the nodes on this anypath. In other words, t has a distance index of 0, and v has the largest index. Since there may exist more than one shortest AAW for node v, node v could have different topological orders in different shortest anypaths. We select the smallest one as v's distance index (denoted by $D(v)$). Let $\mathcal{D}$ denote $\max_{v \in V} D(v)$. Obviously, $\mathcal{D} \leq |V|$.

Lemma 12. *In iteration i, each node with distance index $D(v) = i$ computes its shortest AAW anypath to the destination t.* □

Proof. We prove this lemma by using mathematical induction. Obviously, only the destination node t has a distance index of 0. In the zeroth iteration (i.e. the initialization phase), t can find its own shortest AAW anypath (although it is trivial).

Now we assume that at iteration $i \geq 0$ each node with the distance index $D(v) \in [0, i]$ finds or has found its shortest AAW anypath to the destination t. We will prove the **claim** that *in iteration* $(i+1)$ *each node* v *with the distance index* $D(v) = i+1$ *computes its shortest AAW anypath to the destination* t. Since all the nodes in v's MFAOFS have distance indices less than $i+1$ (recall the definition of the distance index), we know that all the nodes in v's MFAOFS have found their shortest AAW anypaths before the $(i+1)^{th}$ iteration. We are now ready to prove that v can find its MFAOFS in the $(i+1)^{th}$ iteration.

By Lemma 10, we know there exists an MFAOFS of the form $\{j_1, j_2, ..., j_b\}$ for v. The while loop (Lines 3-9) will check $\{j_1\}, \{j_1, j_2\}$... By Lemma 11, if j_1 and j_2 are in v's MFAOFS, using $\{j_1, j_2\}$ always provides a smaller AAW than just using $\{j_1\}$. Thus the condition in Line 6 will be true and the forwarding set is updated to $\{j_1, j_2\}$. The same procedure is repeated until the MFAOFS is found. The claim is therefore proved. The proof is complete. ■

Proof of Theorem 10. Lemma 12 indicates that a shortest AAW anypath for each node to the destination t can be found by DMART. According to Theorem 9, we know that a shortest AAW anypath is actually a K-approximation to the OMCAP$(G, s, t, \overrightarrow{\mathbb{W}}, \overrightarrow{w}, \overrightarrow{p}, K)$ problem. Therefore DMART can find a K-approximation anypath for each node after at most $\mathcal{D} \leq |V|$ iterations.

We now analyze the running time of each iteration. For each node, Line 1 takes $O(\Delta \log \Delta)$ time. If we store some status variables as in [58], Line 5 takes

$O(\Delta)$ in total. Obviously, Lines 4, 6 and 7 take $O(1)$ time, with $O(\Delta)$ in total. Lines 10-12 take $O(K\Delta)$ time. Thus each node needs $O(\Delta(\log \Delta + K))$ time to finish one iteration. ■

Remark. When K=1, our algorithm is the optimal algorithm for the corresponding problem. The shortest anypath algorithm proposed in [25] is a special case of DMART when $K = 1$ and all the vertex weights are equal to 1. When $K \geq 2$, our result is the first distributed $O(1)$-approximation algorithm for the NP-hard OMCAP problem.

4.6.4 Distributed Implementation

Inspired by the Distributed Bellman-Ford protocol proposed in [21], we present the following synchronous proactive protocol based on our DMART algorithm. Each node maintains a routing table entry for each destination <*destination, K anypath weights, auxiliary anypath weight, forwarding set*>. The timeline is divided into a sequence of time intervals of a constant length, each of which is used for one iteration in DMART. In each time interval, each node runs DMART to update its anypath to each destination. If the entries in the routing table change, this node sends path vector tuples <*destination, K anypath weights, auxiliary anypath weight*> to all its immediate neighbors. This is called *path vector updating*. In the next iteration, its immediate neighbors can use these new path vectors to update their routing tables.

Since our synchronous proactive protocol requires a rough time synchronization, we also propose an asynchronous proactive table-driven protocol. Every node periodically sends path vector tuples <*destination, K anypath weights, auxiliary anypath weight*> to all its immediate neighbors. The updating operation frequency depends on the size and the dynamic of the network. Whenever the

entries in the routing table change, this node also triggers the path vector updating. Once a node receives the path vector updates, it uses DMART to update the anypath to the destination. If this computation leads to a routing table change, it triggers a path vector updating.

Now we discuss the stopping criteria for the iterations in our synchronous proactive protocol. Obviously, for a dynamic network this protocol should periodically update the path vector. However, for a static network, we may ask when we can terminate the algorithm iteration. Although we know we can terminate the algorithm after the upper bound of the number of iterations, the tight upper bound $\mathcal{D}$ cannot be obtained easily and the loose bound $|V|$ sometimes is too large. We propose the following "guessing" strategy. An obvious stopping criterion is $\mathcal{W}_m^{(i-1)}(v,P) = \mathcal{W}_m^{(i)}(v,P), \forall v \in V$. We need the following simple *global status checking* as a building block: every node v sends one bit to a leader (elected using well-known leader-election algorithms) to notify whether $\mathcal{W}_m^{(i-1)}(v,P) = \mathcal{W}_m^{(i)}(v,P)$. If this leader finds $\mathcal{W}_m^{(i-1)}(v,P) = \mathcal{W}_m^{(i)}(v,P), \forall v \in V$, it sends back one bit iteration termination signaling. Although this global status checking involves global information exchanges, obviously the overhead is very small especially if the data fusion technique is used. In order to reduce the number of global status checking times, the "guessing" strategy works as follows: after $b^0, b^1, ...b^h, ...$ iterations (b is an integer chosen by network administrators), we do global status checking. Obviously, although we do not know the value of $\mathcal{D}$, after at most $\lceil \log_b \mathcal{D} \rceil$ global status checking operations, the algorithm will terminate.

4.7 Performance Evaluation

4.7.1 Performance of MAP

In this subsection, we present numerical results to demonstrate the performance

of our approximation algorithm MAP.

Since there is no previous algorithm for solving the OMCAP problem and it takes exponential time to obtain an optimal solution, we compared MAP with some variants of the SAF algorithm proposed in [58, 59]. Note that SAF is designed based on EATX, rather than EWATX. However, we can extend the SAF algorithm to support EWATX by computing EWATX instead of EATX in the SAF algorithm. Since SAF is designed to optimize a single metric, we use SAF-k ($k = 1, \ldots, K$) to represent the implementation of SAF only taking into account the kth weight, using EWATX instead of EATX. Using ideas similar to those in the proof of Theorem 8, we can prove that SAF-k computes an optimal anypath with respect to the kth EWATX.

All tests were performed on a 1.8GHz Linux PC with 2G bytes of memory. In the simulation, we uniformly distributed nodes in a $1000m \times 1000m$ square region and evaluated different network sizes that will be described later. For each network size, we randomly generated 1000 test cases by randomly choosing the coordinates for all the nodes and the source-destination pair in each test. As in [113], it is assumed the link delivery probability is inversely proportional to the distance with a random Gaussian deviation of 0.1.

For the generation of vertex weights, we used two settings: a *practical setting* and a *random setting*. For the practical setting, we used two vertex weights which represent average transmission time and power consumption for each transmission, respectively. All the nodes were equipped with 802.11 wireless LAN client adapters. Since anypath routing needs a MAC that supports anycast with relay priority enforced, we use the anycast MAC proposed by Jain and Das [46], which is an enhancement to IEEE802.11. This scheme uses a variant of the IEEE 802.11 handshaking scheme to realize a reliable anycast MAC, and *reduces to 802.11 when*

there is only one next hop forwarder. We briefly describe this scheme in the following. Please refer to [46] for details. This scheme follows the 802.11 Distributed Coordination Function. The transmitter broadcasts to its forwarders an RTS (called MRTS) with all the forwarder addresses (ordered according to their priorities). Intended forwarders use the following rules to reply CTSs if they receive this MRTS. These CTS transmissions must be staggered in time in order of their priorities. The highest priority forwarder replies the CTS after an SIFS, the second one replies after (CTS+3×SIFS) time, the third one after (2×CTS+5×SIFS) time and so on. Once the transmitter receives a CTS, it sends the DATA to the sender of this CTS after an SIFS interval. Other lower priority forwarders hearing this DATA suppress their CTS replies. If DATA is successfully received, this forwarder replies an ACK. Any other node that overhears the MRTS or a CTS will set their corresponding network allocation vectors to avoid sending packets which may result in collision or interference. Each node in our simulation can choose one of the usable transmit power levels: 63, 50, 30, 20 and $10mW$ [20]. In addition, every node can use 18, 11, 6 or $1Mbps$ as its transmission rate. According to [20], the corresponding maximum transmission ranges R_{max} are 122, 149, 198 and $213m$ when the adapter is being used at the maximum transmit power. Since the wireless card may operate on different power levels, the actual transmission range of each transmission rate configuration satisfies $R_{max} \times (\frac{P_t}{P_{max}})^{\frac{1}{\gamma}}$ [32], where the path loss exponent is $\gamma = 2$, the maximum transmit power is $P_{max} = 63mW$, and its actual transmit power P_t is 63, 50, 30, 20 or $10mV$. The two constraints were defined by $\mathbb{W}_1 = 30(ms)$ and $\mathbb{W}_2 = 375(mW)$. Since the first weight and the second weight represent delay and cost, respectively, we use SAF-D to represent SAF-1 and SAF-C to represent SAF-2 in this setting.

For the random setting, we considered $K = 2$ and $K = 3$. All the vertex

weights of each node were uniformly chosen between [1, 10] and all the QoS constraints were defined by $\mathbb{W}_k = 30$ $(k = 1, \ldots, K)$. The node transmission range was set to $200m$.

We report simulation results about the algorithm running time, the lengths of the anypaths obtained by MAP and SAF-k, and some properties of the anypaths obtained by MAP.

4.7.1.1 Evaluation of Running Time

The running times of MAP and SAF are almost identical. Fig. 4.6 shows the running time of MAP. Note that the running times of MAP for the random setting and the practical setting are similar. We observe that the average running time is no more than $3.5ms$ in all cases studied.

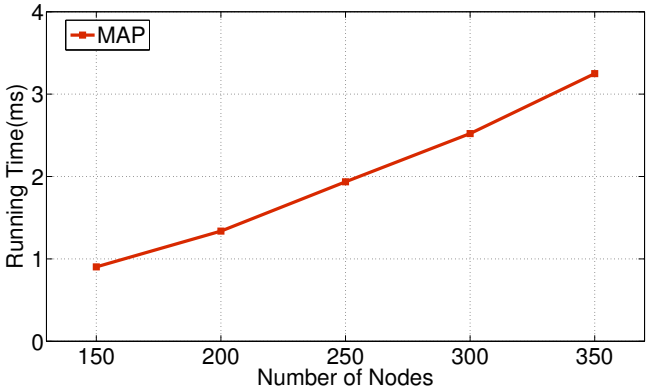


Figure 4.6: Running time

4.7.1.2 Evaluation of Anypath Length

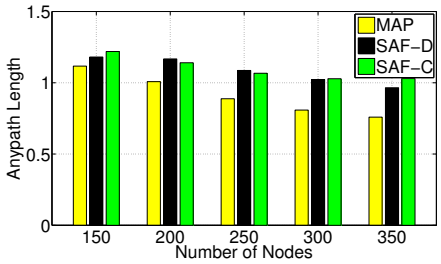
We evaluated the lengths of the anypaths obtained by MAP and SAF-k for both the practical setting and the random setting. The number of nodes was chosen to be 150, 200,..., and 350.

For the practical setting, Fig. 4.7a shows the anypath lengths computed by MAP, SAF-D, and SAF-C. We observe that compared with SAF-D and SAF-C, MAP reduces the average anypath length by 5%-30%. This is as expected, because SAF-D (SAF-C) uses only one metric in decision-making, while MAP uses both metrics in decision-making. Essentially, MAP seeks for some tradeoff between the two metrics.

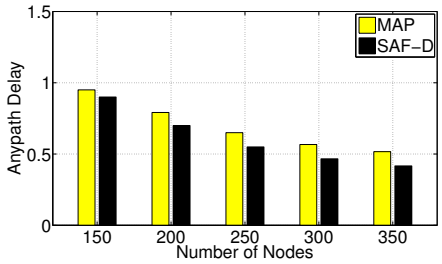
In order to have a closer look at this tradeoff, we compared the delays of the anypaths computed by MAP and SAF-D and also compared the costs of the anypaths computed. Fig. 4.7b shows that the anypath computed by MAP has a higher delay than the anypath computed by SAF-D. This is as expected, since SAF-D aims to find the anypath with the shortest delay. However, the anypath computed by SAF-D has a much larger cost than the anypath computed by MAP, as shown in Fig. 4.7c. When multiple constraints are taken into account, MAP can find a tradeoff (although the delay of the anypath it computes is larger than that computed by SAF-D), so that the anypath length computed has a smaller anypath length than the anypath computed by SAF-D.

Fig. 4.7a also shows that on average the anypath found by MAP is more likely to be a feasible solution to DMCAP than the anypath found by either SAF-D or SAF-C. Recall that an anypath is a feasible solution to DMCAP if and only if its anypath length is less than or equal to 1.

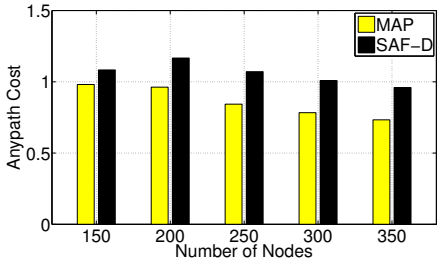
Fig.4.7d and Fig. 4.7e show the comparison of the anypath lengths for the random setting. We observe that compared with SAF-k ($k = 1, \ldots, K$), MAP reduces the anypath length by 29%-49%. As in the case of practical setting, we observe that MAP is more likely to find a feasible solution to DMCAP than SAF-k ($k = 1, \ldots, K$).



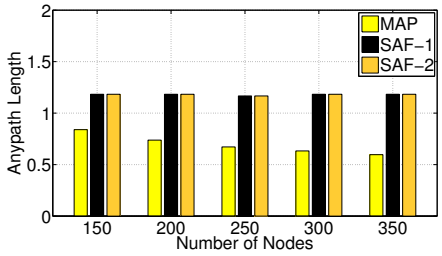
(a) Anypath length (practical setting)



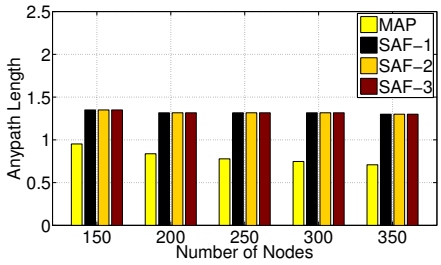
(b) Anypath delay (practical setting)



(c) Anypath cost (practical setting)



(d) Anypath length (random setting: $K = 2$)



(e) Anypath length (random setting: $K = 3$)

Figure 4.7: Anypath lengths

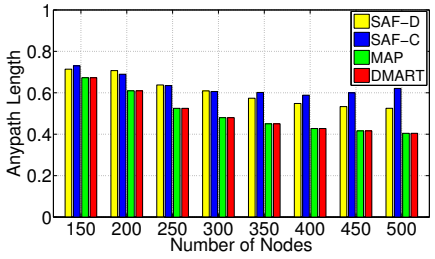
4.7.2 Performance of DMART

In this subsection, we present some numerical results to evaluate the performance of DMART. We implemented DMART for the case of $K = 2$ (i.e. average transmission time and power consumption) in a practical setting using our synchronous proactive protocol on a discrete event simulator. The two vertex weights represent average transmission time and power consumption for each transmission, respectively. We compared our implementation with the SAF algorithm in [58] and our

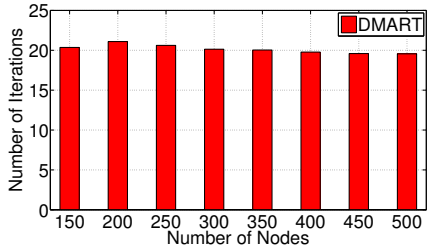
Algorithm 2. As before, we extended the SAF algorithm to support EWATX. Since SAF can only deal with one metric each time, we used SAF to compute the anypath lengths, with respect to the metrics transmission delay and power consumption, respectively (denoted by SAF-D and SAF-C).

We assume that each node was equipped with an 802.11b/g wireless LAN client adapter. Each node can choose 10, 20, 30, 50, or $63mW$ as its transmit power level. In addition, each node randomly selected 1, 6, 11, or $18Mbps$ as its transmission rate [20]. According to [20], the corresponding maximum transmission ranges R_{max} are 213, 198, 149, and $122m$ when the adapter is being used at the maximum transmit power. Since wireless cards may operate on different power levels, the actual transmission range of each transmission rate configuration must satisfy $R_{max} \times (\frac{P_t}{P_{max}})^{\frac{1}{\gamma}}$ [32], where the path loss exponent $\gamma = 2$, $P_{max} = 63mW$ and P_t is the actual transmit power (10, 20, 30, 50, and $63mV$) of a wireless card. The link delivery probability was inversely proportional to the distance with a random gaussian deviation of 0.1. The two constraints were set to $50ms$ and $625mW$. We uniformly distributed nodes in a $1000m \times 1000m$ square region, with the number of nodes chosen to be 150, 200, ... 500. For each network size, we randomly generated 1000 test cases by randomly choosing the transmission ranges and coordinates for all the nodes. We performed all tests on a 1.8GHz Linux PC with 2G bytes of memory.

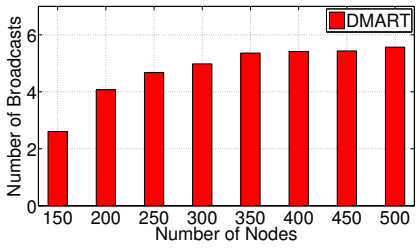
Fig. 4.8a shows the lengths of the anypaths computed by DMART, MAP, SAF-D and SAF-C for a random source-destination pair. We observe that DMART and MAP always outperform SAF-D and SAF-C. This is as expected, because DMART and MAP combine two metrics and thus use more information in decision making, while with only one metric taken into account each time SAF-D and SAF-C obtain biased results. In addition, the average anypath lengths obtained

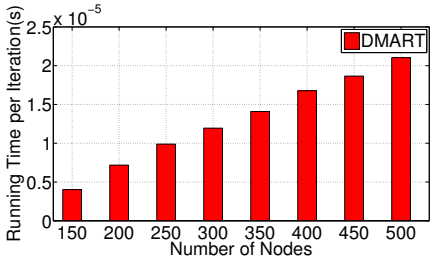


(a) Comparison of average anypath lengths



(b) Average number of iterations





(c) Average number of path vector updating operations for each node

(d) Average running time for each iteration

Figure 4.8: Numerical results

by DMART and MAP are almost the same. This is because both of DMART and MAP approximate OMCAP by looking for the shortest AAW anypath. According to our experiment data, in 97% test cases the anypaths computed by DMART and MAP are the same. However, we must emphasize that DMART is a distributed algorithm which just needs local information from one-hop neighbor, while MAP is a centralized algorithm and needs the global information of the entire network.

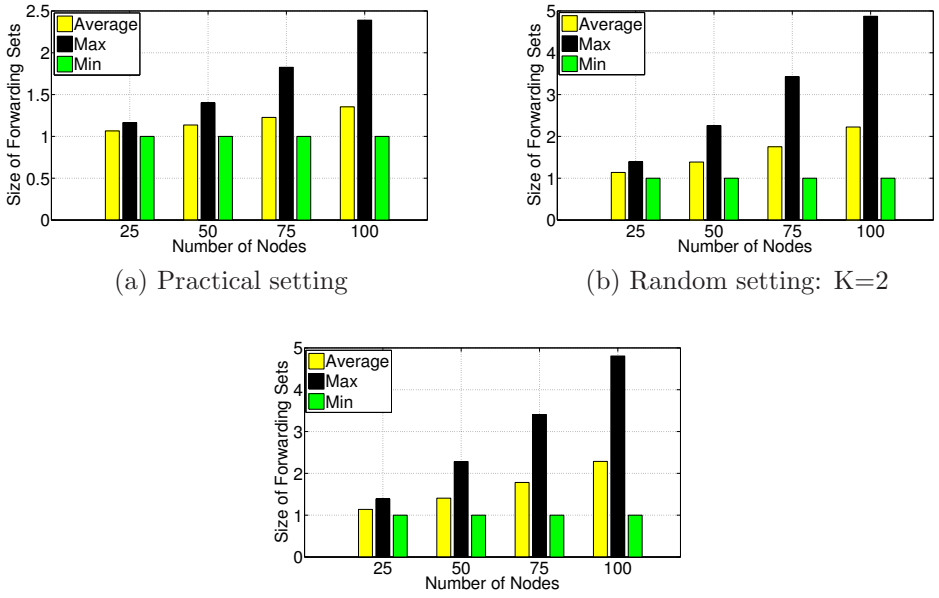
Fig. 4.8b shows the average number of iterations required by DMART to compute anypaths from all nodes to a destination. We observe that the average number of iterations is around 20. Fig. 4.8c shows how many path vector updating operations a node needs. In all cases studied, the number of updating operations is no more than 6. Fig. 4.8d shows that the average running time which each node needs to finish one iteration is also very small. Thus DMART is very efficient.

4.7.3 Performance of Anypaths

In the following, we report the simulation results which reveal some important properties of the anypath computed by our algorithms. Note that since in 97% test cases the anypaths found by MAP and DMART are the same, in the following experiments, we only consider the anypaths computed by MAP. First, we analyze the size of the forwarding sets of the anypaths so computed. Second, we analyze the properties of the single paths within an anypath that a packet may traverse. Recall that different packets may take different single paths within a given anypath. Since the number of single paths within a given anypath may be exponential, we only studied four small network sizes, i.e., the number of nodes was chosen to be 25, 50, 75, and 100.

Figs. 4.9a-4.9c report the size of the forwarding sets computed by MAP. In these figures, "Average", "Max", and "Min" represent the average size, the maximum size, and the minimum size of all the forwarding sets of a computed anypath, respectively. Note that these parameters are computed once for each test case and the results reported here are the averages over 1000 test cases. We observe that the maximum size is no more than 5 in all the cases studied and the average size is between 1 and 2.2. This indicates that in practice a node does not use too many forwarders. We also observe that as the number of nodes increases, the average size of forwarding sets increases as well. This is as expected, because a higher node density implies that a node may have more forwarder candidates to choose from.

The experimental results on the size of the forwarding sets reveal another property of anypaths–the overhead of anypath routing compared to that of traditional routing. Recall that we can use the anycast scheme proposed in [46] as a



(a) Practical setting

(b) Random setting: K=2

(c) Random setting: K=3

Figure 4.9: Size of forwarding sets

MAC protocol to support anypath routing. Compared with the overhead of traditional routing based on 802.11, we will have the following two types of overheads due to the use of anypath: 1) the handshaking packet MRTS should have the addresses of all the forwarders and 2) a forwarder receiving an MRTS has to wait for a longer time before replying a CTS, in order to make sure that higher priority forwarders also receiving the MRTS will reply CTS first. Both types of overheads increase as the size of the forwarding sets increases. Fortunately, according to our simulation, in practice the average size is between 1 and 2.2, which implies that the overhead introduced is fairly small. In other words, compared with the traditional routing protocols that abstract wireless links as wired links to compute link states, anypath routing leverages the broadcast nature of wireless medium to increase the transmission and reception opportunity, with fairly small overhead introduced.

Recall that an anypath is composed of the union of many different paths.

For example, there exist five single paths in the anypath shown in Fig. 4.1. Figs. 4.10a, 4.11a, and 4.12a show the hop counts of the single paths within the computed anypath. In these figures, "Average", "Max", and "Min" represent the average hop count, the maximum hop count, and the minimum hop count of all the single paths existing in an anypath, respectively. Like before, these parameters are computed once for each test case and the results reported here are the averages over 1000 test cases. We observe that the maximum hop count is no more than 7 in the practical setting and no more than 12 in the random setting. Furthermore, as the number of nodes increases, the average hop count increases as well. This is because in a higher node density setting, a node may prefer shorter links that may have higher packet delivery probabilities, which makes longer paths (in terms of hops) involved.

Recall that in order to compute the results shown in Figs. 4.10a, 4.11a, and 4.12a, we computed these parameters once for each test case and then averaged them over 1000 test cases. In order to better understand the relationship between the maximum hop count and the minimum hop count for the anypath obtained in each test case, we use a new metric, called *anypath hop count disparity ratio*. Specifically, we first compute the ratio of the maximum hop count over the minimum of all the single paths existing in the anypath obtained in each test case, and then compute the anypath hop count disparity ratio, by averaging these ratios over the 1000 test cases. We observe that this ratio is no more than 1.5 for the practical setting and no more than 3 for the random setting. This implies that in the anypath so computed the longest single path is not too much longer than the shortest single path in terms of hop counts.

Figs. 4.10c, 4.11c, and 4.12c show the number of single paths existing in the anypath computed by MAP. We observe that the number of single paths

increases fast as the number of nodes increases. This is because the number of single paths within a given anypath may be exponential. Although there may exist a large number of single paths in an anypath, the maximum anypath hop count is no more than 12 and the average anypath hop count is no more than 8 in all the cases studied. This shows that a packet may have many path choices in an unreliable wireless environment but the number of hops it will take to reach the destination is still small. Note that the path taken by a packet is determined on-the-fly, depending on which nodes in the forwarding sets successfully receive the packet at each hop.

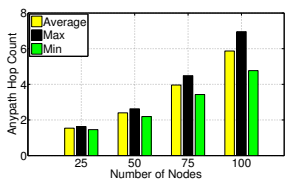
Note that there may be many single paths within a given anypath. For a given value of k, one of these single paths is shortest with respect to the kth weight, where the kth weight of a single path $\mathcal{P}$ is defined as $\sum_{(v,w)\in\mathcal{P}} \frac{w_k(v)}{p(v,w)}$. We use Figs. 4.10d, 4.10e, 4.11d, 4.11e, 4.12d, 4.12e, and 4.12f to illustrate the distribution of the actual weights of the single paths existing in the anypath. Since different anypaths in different test cases may have different lengths, we introduce the following metric, called *single path weight disparity ratio*, which is computed in the following way. For the kth weight, we first find the one with the smallest kth path weight among all the single paths existing in the obtained anyapth. Then for each single path we compute its kth path weight disparity ratio by dividing its kth path weight by the kth path weight of the shortest one we have found. Therefore, a ratio close to 1 means that this single path is almost as good as the best single path. Fig. 4.10d, 4.10e, 4.11d, 4.11e, 4.12d, 4.12e, and 4.12f show the distributions of the disparity ratios. More specifically, in the practical setting, the distributions of the disparity ratios with respect to single path delay and single path cost are shown in Fig. 4.10d and Fig. 4.10e, respectively. The disparity ratio ranges in these two figures are $[1, 1.2), [1.2, 1.4), \cdots, [2.6, 2.8)$, and $[2.8, \infty)$. In the

random setting, the disparity ratios reported in Fig. 4.11d and Fig. 4.11e are the first and the second path weight disparity ratios for the case $K = 2$, respectively. The disparity ratios reported in Fig. 4.12d, Fig. 4.12e, and Fig. 4.12f are the first, second, and third path weight disparity ratios for the case $K = 3$, respectively. The disparity ratio ranges in these five figures are $[1, 1.4), [1.4, 1.8), \cdots, [4.2, 4.6)$, and $[4.6, \infty)$.

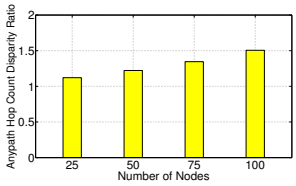
We observe that the majority of the single paths have weights close to that of the shortest one. Furthermore, the percentage of the disparity ratio decreases significantly as its value increases. The percentages of the disparity ratios located in $[2.8, \infty)$ for the practical setting and $[4.6, \infty)$ for the random setting are almost zero. This indicates that most of the single paths actually are good enough compared with the best one. If the packets take these single paths, the delay or cost is just slightly worse compared with that by taking the best single path. However, in an unreliable wireless environment, opportunistically exploiting many single paths is usually less costly and can improve the network performance. Therefore, these sub-optimal but good enough single paths provide such opportunities.

4.8 Conclusion

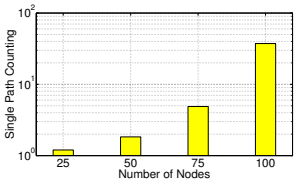
In this work, we have studied anypath routing subject to $K \geq 2$ constraints. We have proved its NP-hardness and presented a centralized polynomial time K-approximation routing algorithm and a distributed polynomial time K-approximation routing algorithm. Numerical results show that our algorithms are very fast and therefore suitable for implementation in wireless routing protocols.



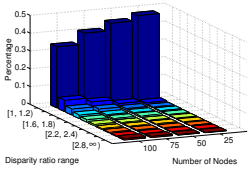
(a) Anypath hop count



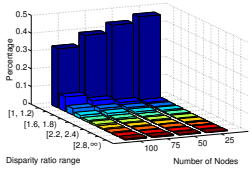
(b) Anypath hop count disparity ratio



(c) Single path counting

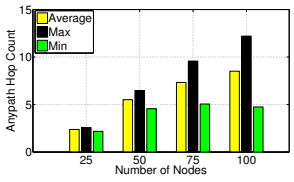


(d) Delay disparity ratio distribution

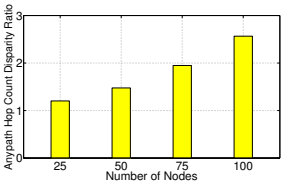


(e) Cost disparity ratio distribution

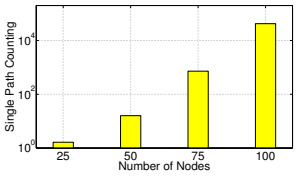
Figure 4.10: Single path property (Practical setting)



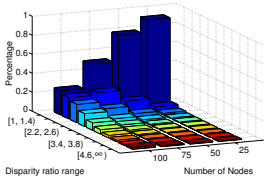
(a) Anypath hop count



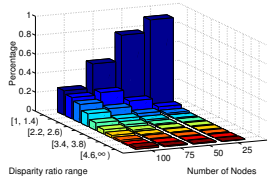
(b) Anypath hop count disparity ratio



(c) Single path counting

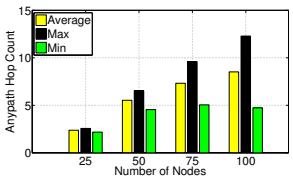


(d) Single path weight disparity ratio distribution ($k = 1$)

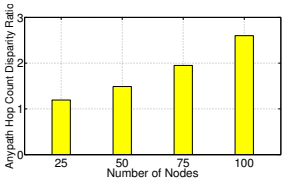


(e) Single path weight disparity ratio distribution ($k = 2$)

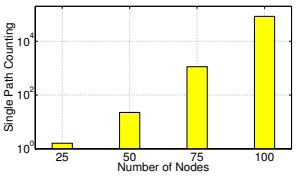
Figure 4.11: Single path property (Random setting: $K = 2$)



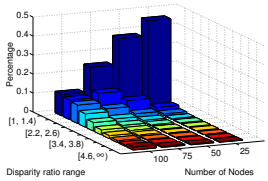
(a) Anypath hop count



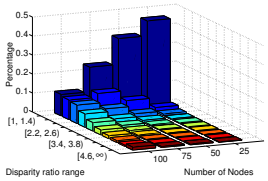
(b) Anypath hop count disparity ratio



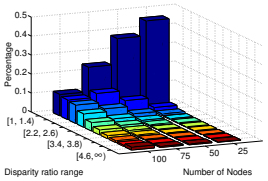
(c) Single path counting



(d) Single path weight disparity ratio distribution ($k = 1$)



(e) Single path weight disparity ratio distribution ($k = 2$)



(f) Single path weight disparity ratio distribution ($k = 3$)

Figure 4.12: Single path property (Random setting: $K = 3$)

Chapter 5

DART: Directional Anypath Routing in Wireless Mesh Networks

5.1 Introduction

5.1.1 Motivation

Prior works along this line were mainly focused on networks equipped with omnidirectional antennas. However, [110] proved that wireless networks can achieve a capacity gain when directional antennas are exploited. [18, 34, 76, 90, 98, 100] have proposed various schemes to improve performance of wireless networks by using directional communications. Thus it is of interest to design an optimally combined use of directional communications and anypath routing. Let us consider a motivating example shown in Fig. 5.1. s, v_1, v_2 and v_3's forwarding sets are $\{v_1, v_2\}$, $\{t, v_4\}$, $\{v_3\}$ and $\{v_1\}$, respectively. Obviously, if the directional transmission range of v_1 just covers t and v_4 as shown in Fig. 5.1, it will not interfere with the transmission from v_2 to v_3, and thus the delay from v_2 to v_3 is reduced. Although [70] touched on the opportunistic routing with directional antennas, it is based on the node distribution models and cannot compute an actual anypath for a source-destination pair in a wireless network. In order to bridge this gap, in this work we study anypath routing for wireless networks equipped with directional antennas.

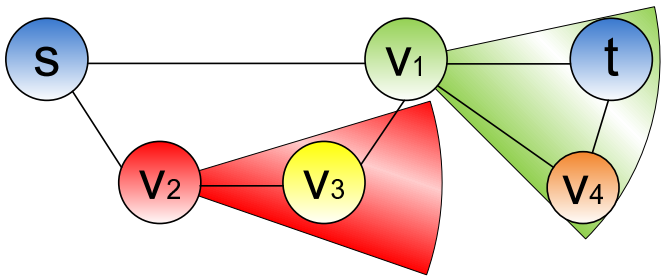


Figure 5.1: A motivating example of directional anypath routing.

From the perspective of *the MAC layer*, the challenge is how to realize *directional anycast* with relay priority guaranteed. A reliable *anycast MAC* even in the omnidirectional antenna system is an active area of research [46]. From the perspective of *the routing layer*, looking for an optimal anypath can reduce to the selection of an optimal forwarding set for each node. However, the challenge is that there is an exponential number of possible forwarding sets for each node, which makes it intractable to try out all the forwarding sets. Furthermore, the selected forwarders should not violate the beamwidth constraint of the directional antenna.

We hence present DART, a cross-layer design of MAC and routing layers for addressing the issues above, which represents the first attempt towards the practical design for directional anypath routing.

5.1.2 Contribution

Our contribution is two-fold:

1. *For the MAC layer, we present a directional anycast MAC (DAM). DAM is an enhancement to the 802.11 MAC [41], and reduces to the 802.11 MAC when there is only one forwarder for each node and only omnidirectional antenna is being used. DAM thus makes DART suitable for integration into the current 802.11 systems.*

2. *For the routing layer, we propose two polynomial time routing algorithms to compute directional anypaths based on two antenna models, and prove their optimality based on the packet delivery ratio metric.*

Simulation results show that compared with omnidirectional anypath routing, DART can reduce the average packet transmission delay by 28%-70%.

5.2 Model

5.2.1 Antenna Model

The antenna system has two separate modes: *Directional* and *Omni*, which could be *envisioned* as two separate antennas: a steerable single beam antenna and an omnidirectional antenna [18]. The boresight of the main lobe can be pointed towards any specified direction in the Directional mode. This kind of antennas is called electrically steerable antenna [18,34,77,98,100]. According to [77], a typical beam switching latency is about $150\mu s$. In principle, both the Directional and the Omni modes can be used to transmit or receive packets. As in [18], *the Omni mode in this work is used for reception, while the Directional mode may be used for transmission as well as reception.* While idle (i.e. not transmitting or receiving) a node stays in the Omni mode. This kind of antenna can be implemented by integrating directional transmitters and omni/directional receivers [19].

The main lobe in the Directional mode is approximated as a circular sector with radius r, called *transmission range* (i.e. the yellow area in Fig.5.2), where $\mathbb{A}_s$ and $\mathbb{A}_e$ denote the start and end angles of the sector (measured counterclockwise from the horizontal axis), and $\mathbb{A}_b$ denotes its beamwidth. The line segments L_s and L_e are called the *start* and *end boundaries*. All the points in this transmission range with the same distance from the center have the same *directional transmission gain*. The radiation pattern of the side lobes is also approximated as a sector. The gain of the side lobes is assumed to be very low, and is thus negligible. Note that this approximation is widely used in the literature [18, 70, 98, 110], and is often called *ideal directional antenna model*. For the beamwidth in the Directional mode, we consider two models. In the *fixed beamwidth* model [34, 90], the beamwidth of each transmitter is fixed although different antennas can use

different beamwidths. In the *variable beamwidth* model [70, 76, 90], each transmitter has an adjustable beamwidth $\mathbb{A}_b \in [A_l, A_u]$, where $0 < A_l \leq A_u \leq 2\pi$. Note that in practice usually A_l cannot be too small to avoid the side effect of creating a too sparse topology or creating too strong side lobes. As in [76], we consider that nodes automatically control the transmit power such that the directional transmission gain is maintained a constant when nodes adjust their beamwidths.

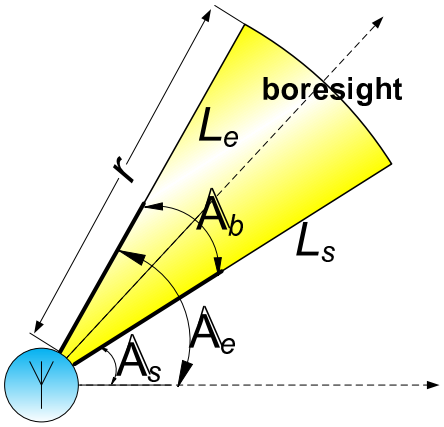


Figure 5.2: Directional antenna model

5.2.2 Network Model

We model a wireless mesh network by a directed graph $G = (V, E)$, where V is the set of vertices, and E is the set of edges. If a sender v can reach a receiver u directly by steering its boresight, we say that there exists a directed edge (v, u) in G. We use the following terms interchangeably: edge and link, vertex and node.

In order to find a high-quality directional anypath, the first problem we are facing is what path metric should be used for routing computation. However, this problem is still an open issue when directional antenna is taken into account. A widely used metric in wireless networks is *expected transmission count* (ETX) [22], which is adopted by traditional and opportunistic routing [6, 9, 25, 27, 58]. Each node computes a *packet delivery ratio* (PDR) for each outgoing link, and based

on that calculates the ETX (i.e. the inverse of PDR) for each link. The PDR is a measured probability that a data packet sent by an omnidirectional transmitter successfully arrives at the receiver. We still use PDR as the metric for our routing computation considering the following two reasons:

1. As discussed in [22], this metric is an approximation. It reveals the link quality itself, but cannot reflect how much actual interference can be reduced for a certain packet transmission by using directional antenna. However, only when other nodes are actually carrying loads, the advantage of using directional communications for interference reduction can be fully appreciated. Generally speaking, in an actual network this load information or pattern is difficult to obtain in advance. *Using PDR can avoid the oscillations that sometimes plague load-adaptive routing metrics, and the route is thus selected independently of network loads.* How to resolve interference, collisions and scheduling is actually handled by the MAC.

2. From a practical point of view, a simple routing metric is preferable and important in real life networks due to the computation and measurement complexity. *PDR, which facilitates the metric measurement and practical routing computation, includes the tractable information that a user can exploit to induce a better route.*

In graph G, each directed edge $(v, u) \in E$ is associated with a PDR $p(v, u) > 0$. We will describe how to measure PDR using probes in Section 5.4. Since a busy link may cause a probe broadcast to be deferred, but ordinarily does not cause it to be lost [22], we assume that the average PDR of edge (v, u) is constant if the directional transmission gain of v at u does not change. This implies that we can steer the boresight or adjust the beamwidth of a transmitter

without changing the PDRs of the edges in G. Thus PDR $p(v, u)$ can be considered as a constant property of each edge for routing computation. Note that this assumption holds for the ideal directional antenna model (see Section 5.2.1). However, this assumption may not hold for some real antenna implementations. Consider two locations with the same distance far away from a directional antenna. The gain at the location close to the boundary of the transmission range may be smaller than the gain at the location close to the boresight. For a real directional antenna, if it satisfies that the obtained gain drops quickly when a location is approaching the boundary, the ideal directional antenna is a good approximate for such real direction antenna.

The *hyperlink delivery ratio* $p(v, F)$ is defined as a measured probability that a packet sent by node v is received by at least one of the nodes in forwarding set F. Since [58] indicates that the loss of a packet at different receivers occurs independently, it can be computed as $p(v, F) = 1 - \prod_{u \in F} (1 - p(v, u))$.

In directional anypath routing, each node is equipped with an antenna system following the antenna model in Section 5.2.1. Each transmitter has a steerable transmission range, called *forwarding sector*. The forwarders of a node must be in its forwarding sector. We use $F \in \Lambda$ to indicate that forwarding set F is located in the forwarding sector Λ. Fig. 5.3 illustrates this concept. The directional anypath is still shown in bold arrows. The forwarding sector of each node is also shown. All the forwarders of a node must be located in its forwarding sector. In brief, compared with traditional anypath routing, each node should choose both its forwarding sector (boresight and beamwidth) and the forwarding set of this forwarding sector rather than just forwarding set.

Now we are ready to define the *directional anypath distance* (DAD) based on PDR. We will discuss the effectiveness of this PDR-based metric in Section

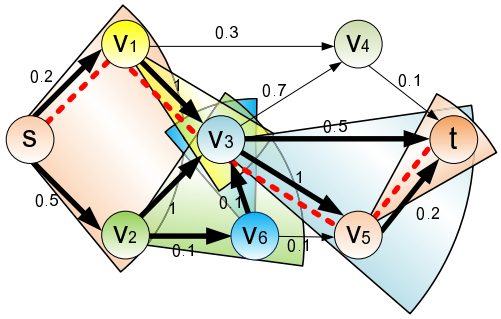


Figure 5.3: Illustration of anypaths. The edge weight denotes its packet delivery ratio. The sector on each node represents its forwarding sector.

5.3. The distance $d(v, F)$ from node v to a set of nodes F is defined as $\frac{1}{p(v,F)}$, which represents the expected number of transmissions for a packet sent by v to be received by at least one node in F. The DAD from v to t along a directional anypath is recursively defined as:

$$\mathbb{D}(v) = \min_{F \in \Lambda} (\mathbb{D}_F(v)) = \min_{F \in \Lambda} (d(v, F) + \mathbb{D}(F)), \tag{5.1}$$

where Λ is the forwarding sector of v, $\mathbb{D}_F(v) = d(v, F) + \mathbb{D}(F)$ is the *anypath distance* of v via forwarding set F, and $\mathbb{D}(F)$ is the *remaining DAD* from F to the destination. It is intuitively defined as a weighed average of the DADs from the nodes in the forwarding set F to the destination:

$$\mathbb{D}(F) = \sum_{j_\beta \beta \in F} \alpha(j_\beta, j_\beta \beta) \mathbb{D}(j_\beta \beta), \tag{5.2}$$

with $\sum_{j_\beta \beta \in F} \alpha(j_\beta, j_\beta \beta) = 1$, where the coefficient $\alpha(j_\beta, j_\beta \beta)$ represents the probability of a node with priority $\beta \in [1, |F|]$ forwarding a received packet. The value of the node priority (i.e. β) is called its *priority index*. For example, the highest priority node u_1 has priority index 1.Thus, a forwarder with a larger DAD should have a lower priority or a larger priority index [25]. Obviously, a node with priority β forwards a packet only when it receives this packet and none of the higher priority nodes receives it, which happens with probability $p(v, j_\beta \beta) \prod_{q=1}^{\beta-1} (1 - p(v, u_q))$.Thus

$\alpha(j_\beta, j_\beta\beta)$ can be computed as

$$\alpha(j_\beta, j_\beta\beta) = \frac{p(v, j_\beta\beta)\prod_{q=1}^{\beta-1}(1-p(v, j_\beta q))}{p(v, F)} = \frac{p(v, j_\beta\beta)\prod_{q=1}^{\beta-1}(1-p(v, j_\beta q))}{1-\prod_{q=1}^{|F|}(1-p(v, j_\beta q))},$$

with the denominator being the normalizing constant. Obviously, according to this definition, $\mathbb{D}(v)$ represents the expected total number of transmissions necessary for a packet sent by v to be received by destination t along this directional anypath. The forwarding set in Λ, which can provide the minimum $\mathbb{D}_F(v)$, is called the *optimal forwarding set of* Λ.

As an example, consider the network depicted in Fig. 5.3 to compute $\mathbb{D}(v_3)$. The forwarding sectors of v_3 and v_5 are shown as the two sectors. The anypath distance from v_3 to t via the forwarding set $F = \{t, v_5\}$ (note that t and v_5 are located in the forwarding sector of v_3) is calculated as

$$\begin{aligned}\mathbb{D}_F(v_3) &= d(v_3, F) + \mathbb{D}(F) \\ &= \frac{1}{1-(1-0.5)(1-1)} + \frac{0.5\times 0 + (1-0.5)\times 1\times(\frac{1}{0.2}+0)}{1-(1-0.5)(1-1)} = 3.5.\end{aligned}$$

Note that t can be considered as a forwarder with DAD 0, and that the DAD of v_5 is calculated as $(\frac{1}{0.2}+0)$. Likewise, we can compute $\mathbb{D}_{\{v_5\}}(v_3) = 6$ and $\mathbb{D}_{\{t\}}(v_3) = 2$. Thus, $\mathbb{D}(v_3) = \min\{2, 3.5, 6\} = 2$, and $\{t\}$ is the optimal forwarding set of v_3's current forwarding sector. As explained in [25, 58], adding an extra node to the forwarding set is not always beneficial even if it reduces the forwarding delay from a node to any of its forwarders. For instance, clearly the anypath distance of v_3 via $\{t\}$ is smaller than that via $\{t, v_5\}$.

Remark: Note that the above routing metric does not consider boresight or beamwidth adjustment delay. This is because we consider the routing metric from the perspective of the number of transmissions. In an unreliable wireless environment, compared with the total transmission time (including retransmission

time) of a packet, the boresight or beamwidth adjustment delay is usually much smaller. Therefore, our routing metric does not consider boresight or beamwidth adjustment delay.

5.3 Problem Formulation

MAC layer For directional anycast, the MAC should address the following issues.

1. It must solve directional transmissions, interference, collisions and the corresponding scheduling problems existing in a directional antenna system.
2. It must realize anycast with relay priority guaranteed.
3. It must know whether a packet has been delivered to one of its forwarders, and when to retransmit if failed.
4. It must consider the compatibility with the currently existing wireless networks (such as 802.11 networks).

Routing layer Routing layer aims to find a shortest directional anypath from a source to a destination. We need to point out that the shortest directional anypath defined on PDR may not be the real optimal one for a certain packet transmission since the metric PDR cannot reflect the actual reduction on interference when directional antenna is being used. However as discussed in Section 5.2.2, we remark that this formulation is a practical and helpful approximation, largely due to the undetermined load pattern and probabilistic nature of lossy wireless networks.

We study two versions of the routing problem based on the two beamwidth models described in Section 5.2.1. The first version is the *fixed beamwidth shortest*

directional anypath routing (FBDAR). Finding a shortest directional anypath is equivalent to finding an optimal forwarding sector for each node, which can provide the shortest DAD, and the optimal forwarding set of this forwarding sector. Note that since the beamwidth is fixed for each node, we only need to determine the boresight of the optimal forwarding sector for each node. The second version is the *variable beamwidth shortest directional anypath routing* (VBDAR). Similar to FBDAR, we need to find an optimal forwarding sector and the optimal forwarding set of this forwarding sector for each node. Note that in this version the optimal forwarding sector refers to the one with the minimum beamwidth among all the forwarding sectors which can provide the shortest DAD.

5.4 MAC Layer Design

In this subsection, we present a MAC design to support the practical implementation of DART.

5.4.1 Directional Anycast MAC

We present DAM, a directional anycast MAC, which is an enhancement to the 802.11 MAC [41]. A similar handshaking scheme (i.e. RTS/CTS/DATA/ACK [41]) is used in DAM. We use a directional network allocation vector (DNAV) [18] to keep track of the directions (and the corresponding durations) towards which a node must not initiate a transmission. The well-known network allocation vector (NAV) used in 802.11 indicates the duration for which the node must defer transmission to avoid interfering with some other transmission. The difference between DNAV and NAV is that if a node receivers an RTS/CTS from a certain direction, it needs to defer only those transmissions that are directed in that direction. The entry in DNAV is updated based on the "duration" field and the incoming direction of the overheard packets.

When the routing layer passes a packet and the information about forwarders to DAM, DAM requests the physical layer to beamform to the forwarding set, and performs carrier sensing for at least a DIFS (defined in 802.11 [41]) to detect whether the channel is idle. Carrier sensing is performed by both physical and virtual mechanisms. Virtual carrier sensing is performed based on DNAV. More specifically, if node u has a packet to send to v, it must check its DNAV table to decide if it is safe to transmit in the direction of v. If idle, DAM enters the backoff phase as in 802.11. Once its backoff counter counts down to zero, the transmitter will broadcast *directionally* to these forwarders an RTS (called MRTS) with all the forwarder addresses (ordered according to their priorities). If the medium is busy, the transmitter waits until it becomes idle.

Recall that all the idle nodes stay in the Omni mode. Once they receive a signal arriving from a particular direction, they lock on to this direction and receive it. If an intended forwarder receives the MRTS packet, it responds by a CTS *directionally*. However, these CTS transmissions must be staggered in time in order of their priorities. The highest priority forwarder replies the CTS after an SIFS (defined in [41]), the second one replies after (CTS+3×SIFS) time, the third one after (2×CTS+5×SIFS) time and so on. When the transmitter receives a CTS, it *directionally* sends the DATA to the sender of this CTS after an SIFS interval. This can guarantee that the lower priority forwarders hear this DATA before they transmit CTSs, thus they suppress any further CTS transmissions. *This is the key point to enforce the prioritization among forwarders.* All *such* forwarders set their DNAVs (i.e. this direction is busy) until the end of the ACK period, and switch back to Omni mode. Note that it is possible that an intended forwarder has locked on to another direction, and is thus "deaf" to this MRTS. DAM does not require all the intended forwarders to be idle when a node is trying

to send an MRTS. The "deafness" will be discussed in Section 5.4.1. If DATA is successfully received, this forwarder replies an ACK *directionally*.

In practice the channel status could change from the point when CTS is transmitted to the point when DATA or ACK is transmitted, which causes the exchange to fail. However, [46] proves that the probability is low, since for static or slowly changing dynamic wireless networks the coherence period is expected to be large enough for the DATA transmission to succeed, if the transmissions of RTS and CTS are indeed successful. If the DATA transmission fails indeed (i.e. ACK-timeout), DAM just repeats the whole procedure (i.e. MRTS/CTS/DATA/ACK).

Any other node that overhears the MRTS or the DATA (i.e. exposed node) sets its DNAV for the entire duration specified in the MRTS or DATA. Specifically, if the MRTS is received, the DNAV is set to (k×CTS+(2k+1)×SIFS+DATA+ACK) time, where k is the number of forwarders. If the header of the DATA is received, the DNAV is set to (SIFS+DATA+ACK) time. Likewise, any node that overhears any of the CTSs (i.e. hidden node) sets its DNAV until the ACK period. That is to say, a node upon receiving the i-th CTS, must set its DNAV to $((2(k-i)+1)\times$SIFS+$(k-i)\times$CTS+DATA+ACK) time.

If no CTS comes back within a CTS-timeout duration, the transmitter's DAM will increase its contention window (if the maximum has not been reached), enter the backoff phase as in 802.11, and schedule a retransmission of the MRTS.

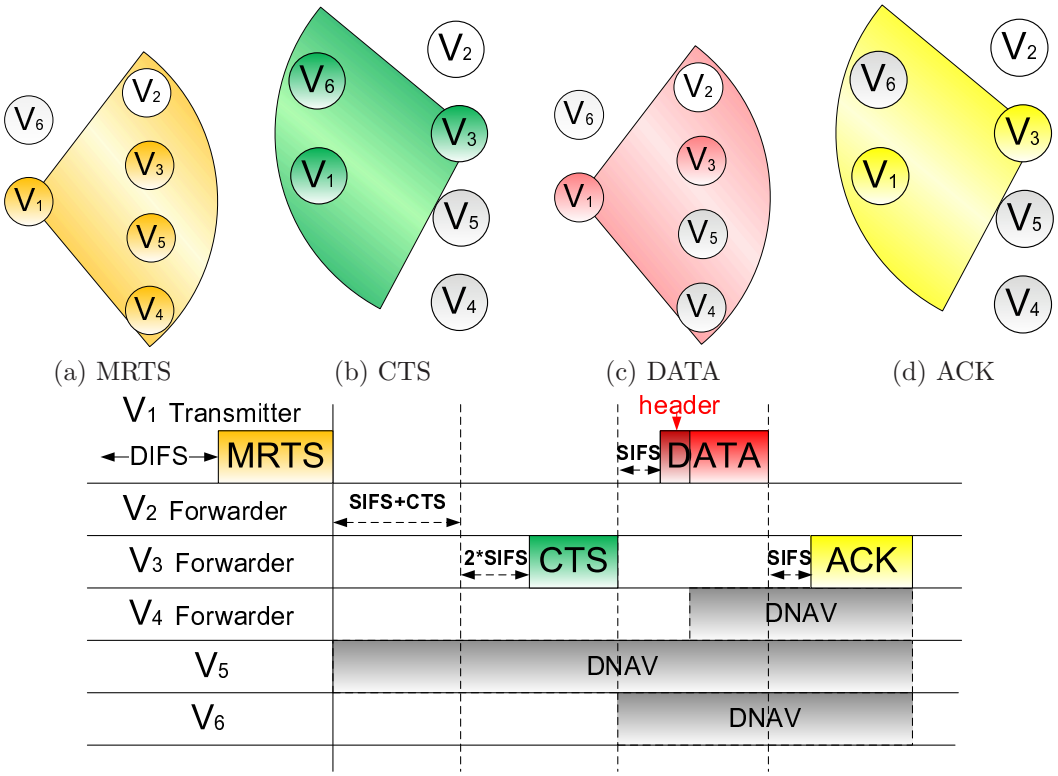
Note that DAM actually slightly deviates from the basic operations of anypath routing in [25]. [25] requires some forwarders to suppress redundant forwardings. DAM realizes this by suppressing redundant CTS replies to guarantee that only one forwarder will be responsible for forwarding this packet.

We use Fig.5.4 to illustrate the basic operation of DAM. Suppose that

v_1 has three forwarders v_2, v_3, and v_4. v_2 has the highest relay priority, v_3 has the second highest one, and v_4 has the lowest one. v_1 performs both physical and virtual carrier sensing for at least a DIFS to detect whether the channel is idle. Once it finds the channel is idle, it broadcasts an MRTS directionally to its forwarders v_2, v_3, and v_4 (see Fig.5.4a). Suppose that currently the channel between v_1 and v_2 is bad. Thus v_2, who has the highest relay priority, does not receive the MRTS from v_1. However, v_3, v_4 and v_5 receive this MRTS. Since v_5 is not the intended forwarder, it sets its DNAV and thus will not initiate a transmission towards the direction of v_1 until the ACK period (see Fig.5.4e). v_3 waits for (CTS+3×SIFS) time (see Fig.5.4e), and replies a CTS directionally (see Fig.5.4b). Suppose that both v_1 and v_6 receive this CTS. Thus, v_6 sets its DNAV and will not initiate a transmission towards the direction of v_3 until the ACK period (see Fig.5.4e), and v_1 starts to transmit DATA directionally to v_3 (see Fig.5.4c). After having received the header of the DATA, v_4 suppresses its CTS reply and sets its DNAV (see Fig.5.4e). When the DATA is received, v_3 replies an ACK directionally (see Fig.5.4d). Therefore, a successful priority-enforced directional anycast is done.

5.4.2 Practical Issues

PDR measurement Although we suggest using the traditional PDR [22] as our routing metric, considering that only the Directional mode is used for transmission, we slightly modify its operation for adapting to DART. In order to measure the average PDR, each transmitter keeps track of actually used beamwidths, and uses the moving average as the antenna beamwidth (if the beamwidth is adjustable) when it is measuring PDRs. In addition, each transmitter selects several measure directions with directional transmission ranges covering the whole 360° area, and broadcasts dedicated link probe packets directionally and periodically



(e) Timeline showing a successful directional anycast

Figure 5.4: Illustration of the basic operation of DAM: suppose that currently the channel between v_1 and v_2 is bad; (a)v_1 broadcasts an MRTS to v_2 (first highest relay priority), v_3 (second highest relay priority) and v_4 (third highest relay priority); the MRTS is heard by v_3, v_4 and v_5; v_5 sets its DNAV and thus will not initiate a transmission towards the direction of v_1; (b) since v_2 (with the highest relay priority) does not receive the MRTS, v_3 replies a CTS directionally after waiting for (3×SIFS+CTS) time; v_3's CTS is heard by v_1 and v_6; v_6 sets its DNAV and thus will not initiate a transmission towards the direction of v_3; (c) v_1 starts transmitting data directionally to v_3; v_3 locks on to the direction of v_1 and receives DATA; after having received the header of the DATA, v_4 suppresses its CTS reply, sets DNAV and thus will not initiate a transmission towards the direction of v_1; (d) once the DATA is successfully received, v_3 replies an ACK directionally.

if this direction is not busy. Every node remembers the probes it receives during the last several seconds and calculates the PDR [22].

Dynamic networks and route computation We mainly consider static or slowly changing dynamic mesh networks and thus use centralized routing algo-

rithms. In other word, network topology and link status are assumed to be static. The MAC of each node periodically measures PDRs, and keeps certain neighborhood status information dynamically. This status information from each node is propagated periodically through the whole network. This helps each node have the approximate knowledge of the network status periodically, and thus each node can adaptively compute an anypath towards destinations.

Location information Each node can use the Angle of Arrival technique [100] or the Direction of Arrival technique [18] to estimate the relative angle between each of its neighbors and itself based on a virtual axis. We will see that this relative angle information is enough for our routing algorithms.

Scheduling Since DAM is an enhancement to the 802.11 MAC, for the medium access, DART respects the 802.11 Distributed Coordination Function (DCF) [41]. For the problem of deciding which forwarder should be scheduled to receive the packet, DAM utilizes MRTS broadcasts and prioritized CTS replies. DAM does not require all the intended forwarders to be idle when it is trying to send an MRTS to fully utilize transmission opportunities if the transmission is safe.

Compatibility with existing wireless networks Our DAM reduces to the Basic Directional MAC protocol in [18] to some extent when there is only one forwarder for each node, and further reduces to 802.11 if only Omni mode is allowed.

Deafness (or failed carrier sense) problems DAM could suffer from deafness. Note that these issues also exist in other directional MACs, such as [18]. How to completely solve these issues is out of the scope of this dissertation. In ad-

dition, during the switching between omni and directional mode, the node cannot perform carrier sense and thus may transmit and interfere with other nodes' transmissions. If these transmissions fail due to this interference, DAM just repeats the whole procedure (i.e. MRTS/CTS/DATA/ACK). Considering the duration of the switching operation between omni and directional mode is short, this case does not occur too frequently.

5.5 Routing Algorithm for FBDAR

5.5.1 Algorithm Description

In this subsection we present an efficient algorithm F-DART for FBDAR, which computes a shortest directional anypath from each node to the given destination t.

From a high-level perspective, this algorithm is a Dijkstra-like algorithm. However, as we study directional anypath routing, we need to use the formulas given in Section 5.2.2 to update distances and the corresponding data structures.

For each node $v \in V$, we keep a variable $\mathbb{D}(v)$ to store the currently computed shortest DAD from v to t. Let $\mathbb{A}_s(v)$ and $\mathbb{F}(v)$ denote the start angle of the corresponding forwarding sector and the corresponding forwarding set of this forwarding sector. Recall that in FBDAR for each node, its beamwidth is its property and known as a fixed number. Therefore, the start angle can uniquely determine the boresight of its forwarding sector. We use a list $\mathbb{R}(v)$ to keep track of all the candidate forwarding sectors for v. Each element (called a *record*) in $\mathbb{R}(v)$ keeps the corresponding information of this forwarding sector, including a set F and two variables A_s and D, which denote its forwarding set, its start angle and the anypath distance of v via this forwarding set, respectively. Additionally, we keep two data structures: L and Q. L is a list that stores all the *settled nodes*

for which we have already found the shortest directional anypaths. We store all the other nodes in a priority queue Q keyed by their current $\mathbb{D}(v)$ values.

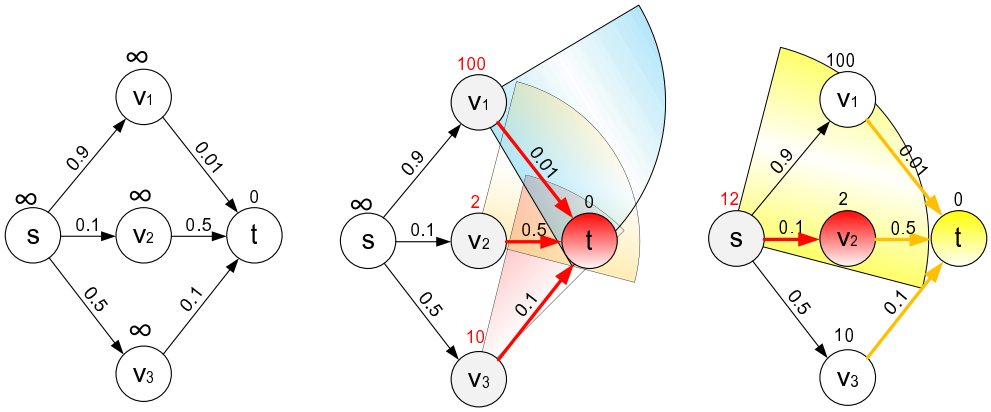
Algorithm 3 F-DART

1: **for** each node v in V **do**
2: $\mathbb{D}(v) \leftarrow \infty$, $\mathbb{A}_s(v) \leftarrow 0$, $\mathbb{F}(v) \leftarrow \emptyset$, $\mathbb{R}(v) \leftarrow \emptyset$.
3: **end for**
4: $\mathbb{D}(t) \leftarrow 0$, $L \leftarrow \emptyset$, $Q \leftarrow V$.
5: **while** $Q \neq \emptyset$ **do**
6: $u \leftarrow$ Extract-Min(Q), $L \leftarrow L \cup \{u\}$.
7: **if** $\mathbb{D}(u) = \infty$ **then** break.
8: **for** each unsettled incoming neighbor v **do**
9: **for** each record R in $\mathbb{R}(v)$ **do**
10: **if** u is in the forwarding sector of R **then**
11: $F \leftarrow R.F \cup \{u\}$, $D \leftarrow \frac{1}{p(v,F)} + \mathbb{D}(F)$.
12: **if** $D < R.D$ **then**
13: $R.F \leftarrow F$, $R.D \leftarrow D$.
14: **end if**
15: **end if**
16: **end for**
17: Construct a record R_c for v such that u is on the start boundary of the forwarding sector of R_c. Set $R_c.F$ to the set of all the settled outgoing neighbors located in this forwarding sector. Calculate $R_c.D$.
18: $\mathbb{D}(v) \leftarrow \min_{R \in \mathbb{R}(v)} R.D$, $\mathbb{A}_s(v) \leftarrow \arg\min_{R.D} R.A_s$, $\mathbb{F}(v) \leftarrow \arg\min_{R.D} R.F$; update Q.
19: **end for**
20: **end while**

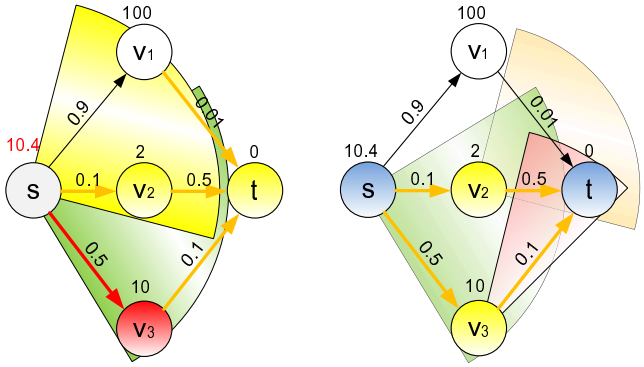
5.5.2 Algorithm Illustration

Now we briefly describe how Algorithm 3 works using the example shown in Fig.5.5. Suppose that the beamwidths of s, v_1 and v_2 are 90°, and the beamwidth of v_3 is 30°. Lines 1-4 initialize all the data structures, and set $\mathbb{D}(t)$ to 0. In the main while-loop, each time Line 6 extracts from Q a node with the minimum $\mathbb{D}$, and inserts it into L. If its $\mathbb{D}$ is ∞ already, all the nodes still in Q cannot reach the destination t and thus the algorithm terminates. In the first iteration,

t is extracted in Line 6. Lines 8-18 then perform relaxations for all its incoming neighbors v_1, v_2 and v_3. We take v_2 as an example. Since now for v_2, its $\mathbb{R}(v_2)$ is empty, Lines 9-13 are skipped. Line 17 constructs a forwarding sector for v_2 (as shown in Fig.5.5b), and a corresponding record which stores the information of this forwarding sector. Note that now t is added into the forwarding set associated with this forwarding sector since t is the only settled outgoing neighbor located in it. Then for v_2, Line 18 updates the corresponding data structures by choosing the one which can provide the shortest DAD among all the constructed records (note that now v_2 has only one record). Thus its currently computed shortest DAD is 2. Similar relaxations are performed on v_1 and v_3, and their currently computed shortest DADs are updated to 100 and 10, respectively. In the second iteration, v_2 is extracted in Line 6 since its $\mathbb{D}(v_2)$ is the smallest among all the unsettled nodes. Algorithm 3 repeats the similar procedure, relaxes its incoming neighbor s and constructs the first record for s (as shown in Fig.5.5c) with the forwarding set $\{v_2\}$ (since v_2 is the only settled outgoing neighbor located in it.). In the third iteration, v_3 is extracted, and Lines 8-18 perform relaxations for its incoming neighbor s. Note that now $\mathbb{R}(s)$ is not empty. Lines 9-13 check for the first forwarding sector (the yellow one) in $\mathbb{R}(s)$ whether v_3 is located in it. Since this forwarding sector only has v_2 in it, and v_3 is not located in it, this if-branch is skipped. Then Line 17 constructs the second forwarding sector for s (the green one) as shown in Fig.5.5d, and sets its forwarding set of this record to the set of all the settled outgoing neighbors located in this forwarding sector (i.e. v_2 and v_3). Since the second record leads to a smaller DAD, Line 18 chooses this forwarding sector and updates the corresponding data structure. As shown in Fig.5.5e, in the fourth iteration s is extracted and thus settled.



(a) An example network (b) t is settled (c) v_2 is settled



(d) v_3 is settled (e) s is settled

Figure 5.5: Algorithm illustration: the weight on each link denotes its PDR. The weight associated with each node denotes its currently computed shortest DAD $\mathbb{D}(v)$. The sectors associated with each node denote its forwarding sectors in its $\mathbb{R}(v)$.

5.5.3 Algorithm Analysis

Theorem 11. *Algorithm 3 finds a shortest directional anypath for FBDAR in* $O\left(|E|\left(\log|V| + \Delta(G)\log\Delta(G)\right)\right)$ *time, where* $\Delta(G)$ *is the maximum out-degree of graph* G. □

We need to give Lemmas 13-17 before proving Theorem 11. Let $\theta(v)$ denote the optimal DAD of v. Note that Lemmas 13-15 are similar to Lemmas 7- 9,

which studied multi-constrained anypath routing. We can use similar techniques to prove Lemmas 13-15 and thus omit the proofs for these three lemmas.

Lemma 13. *If the forwarders of node v can only be chosen from a nonempty subset of its outgoing neighbor nodes, denoted by $N(v) = \{u_1, u_2, \ldots, u_z\}$ (with $\theta(u_1) \leq \theta(u_2) \leq \cdots \leq \theta(u_z)$), there must exist an optimal forwarding set of the form $\{u_1, u_2, \ldots, u_b\}$ for some $b \in \{1, 2, \ldots, z\}$, called a full optimal forwarding set (FOFS) in $N(v)$ for node v.* □

Suppose that there are four optimal forwarding sets in $N(v)$ for node v: $\{u_1, u_2\}$, $\{u_1, u_2, u_3\}$, $\{u_1, u_2, u_3, u_4\}$, and $\{u_1, u_2, u_4\}$. This could happen, for example, when $p(v, u_2) = 1$. We can easily verify that these three forwarding sets lead to the same DAD. $\{u_1, u_2\}$, $\{u_1, u_2, u_3\}$, $\{u_1, u_2, u_3, u_4\}$ are FOFSs. Recall that a node's forwarders must be located in its forwarding sector. Once the forwarding sector is determined, the set of nodes, from which we choose forwarders, is therefore determined. Let $N(v) = \{u_1, u_2, ..., u_z\}$ denote the set of these nodes. Lemma 13 implies that we only need to check forwarding sets $\{u_1\}, \{u_1, u_2\}, ...$ *Thus the time complexity of finding the optimal forwarding set of a forwarding sector can be reduced to a polynomial time from an exponential time.* The proof of Lemma 13 will construct such an FOFS (refer to the proof of Lemma 7) that among all the optimal forwarding sets in $N(v)$, the largest priority index node in this FOFS has the smallest priority index. Thus, if we check the forwarding sets in this order $\{u_1\}$, $\{u_1, u_2\}$, ..., this constructed FOFS would be found earliest among all the optimal forwarding sets in $N(v)$. We call this FOFS a *minimum FOFS* (MFOFS) in $N(v)$. Consider the example above. We have three FOFSs $\{u_1, u_2\}$, $\{u_1, u_2, u_3\}$, and $\{u_1, u_2, u_3, u_4\}$. The MFOFS is $\{u_1, u_2\}$, since the priority index of the largest priority index node in $\{u_1, u_2\}$ is 2, which is smaller than 3 in $\{u_1, u_2, u_3\}$, and 4 in $\{u_1, u_2, u_3, u_4\}$.

However, since there are an infinite number of possible boresights of the forwarding sector, we may ask which is the optimal. Lemma 16 will show that we only need to check a polynomial number of boresights. We call the MFOFS of the optimal forwarding sector *an optimal MFOFS* (OMFOFS). For example, we have two forwarding sectors available. The MFOFS in the first one leads to a DAD of 5, while the MFOFS in the second one leads to a DAD of 10. The MFOFS in the first one is an OMFOFS.

If all the nodes on a shortest (i.e. optimal) directional anypath from v to t use their OMFOFSs, we call it a *full shortest anypath* or a *full optimal anypath*.

Lemma 14. *For an arbitrary nonempty subset $N(v)=\{u_1, u_2, \ldots, u_z\}$ of the outgoing neighbors of a node v, its MFOFS is $\{u_1, u_2, \ldots, u_b\}$ with optimal distances $\theta(u_1) \leq \theta(u_2) \leq \cdots \leq \theta(u_b)$. Let S_μ denote the set $\{u_1, u_2, \ldots, u_\mu\}, 1 \leq \mu \leq b$. Then we have $\mathbb{D}_{S_1}(v) > \mathbb{D}_{S_2}(v) > \cdots > \mathbb{D}_{S_b}(v)$.* □

Lemma 15. *The optimal DAD $\theta(v)$ of a node v is always larger than that of any node (other than v) on the full shortest anypath from node v to destination t.* □

Lemma 16. *Let $\Delta(v)$ denote the out-degree of node v in G. Although there exist an infinite number of possible forwarding sectors for v, in order to find the optimal one, it is sufficient to check $O(\Delta(v))$ particular forwarding sectors.* □

Proof. We construct the $O(\Delta(v))$ forwarding sectors for v as follows. For each outgoing neighbor node of v, we construct a forwarding sector of v such that this node is on its start boundary. We can therefore construct $O(\Delta(v))$ forwarding sectors, called *forwarding sector candidates*. We now consider an arbitrary forwarding sector Λ' of v, and prove that the DAD of v via Λ' cannot be smaller than that via one particular forwarding sector candidate. If there is an outgoing neighbor node of v on the start boundary of Λ', Λ' is a forwarding sector candidate.

Otherwise we rotate Λ' counter-clockwise until the start boundary first touches an outgoing neighbor node of v, which is originally in the interior of Λ'. Λ' now has been transformed to a forwarding sector candidate, denoted by Λ. Obviously, the outgoing neighbor nodes of v, which are covered by Λ', are also covered by Λ. By (5.1), we therefore know that the DAD of v via Λ' cannot be smaller than that via Λ. Thus we do not need to check Λ'. Therefore it is sufficient to check all the forwarding sector candidates. ■

Consider v's OMFOFS $\psi = \{u_1, u_2, ..., u_b\}$ and an optimal forwarding sector of v, which covers all the nodes in ψ. As in the proof of Lemma 16, we rotate it counterclockwise until its start boundary touches a node in ψ and we thus obtain a forwarding sector candidate, denoted by Λ_o. Obviously, Λ_o is also an optimal forwarding sector. An outgoing neighbor node of v is called a *start boundary construct node (SBCN)* of Λ_o, if it is the one with the smallest optimal DAD, among all the nodes in ψ, which are located on the start boundary of Λ_o. Intuitively, if we check nodes in the order of $u_1, u_2, ..., u_b$, and construct forwarding sector candidates with these nodes on their start boundaries, the SBCN is the first node which can correctly determine the start boundary of Λ_o.

Lemma 17. *Let v be any node that can reach the destination t. The anypath from v to t computed by Algorithm 3 is a shortest directional anypath.* □

Proof. We show that for each node v, when it is inserted into L, we have $\mathbb{D}(v) = \theta(v)$. For the purpose of contradiction, let v_a be the first node added to L for which $\mathbb{D}(v_a) > \theta(v_a)$.

Since there could exist more than one OMFOFS for v_a, at first we prove by contradiction a claim that all the OMFOFSs contain at least one unsettled node at the time v_a is inserted into L. We assume that there exists an OMFOFS

ψ, in which all the nodes have been settled and added into L, and we will derive that Algorithm 3 must find ψ and thus $\mathbb{D}(v_a) = \theta(v_a)$. Lemma 16 indicates that we only need to check the forwarding sector candidates of v. Now we prove ψ will be constructed and found, as well as the corresponding optimal forwarding sector Λ_o. We know Λ_o must exist. Let $\{u_1, u_2, ..., u_z\}$ denote the set of all the outgoing neighbor nodes of v located in Λ_o, with $\theta(u_1) \leq \theta(u_2) \leq \cdots \leq \theta(u_z)$. By Lemma 13, we know ψ is of the form $\{u_1, u_2, ..., u_b\}$. Since v_a is the first node added to L for which $\mathbb{D}(v_a) > \theta(v_a)$ and we assume all the nodes in ψ have been added into L, all the nodes in ψ must be settled in the increasing order of their optimal DADs. The SBCN of Λ_o, denoted by u_c, is therefore settled before v_a is added into L. When u_c is settled, Λ_o is constructed in Line 17. Therefore, Λ_o must be constructed before v_a is added into L, and all the nodes in Λ_o with optimal DADs not greater than $\theta(u_c)$ (i.e. $u_1, u_2, ..., u_{c-1}$) are settled before Λ_o is constructed. Recall that ψ is of the form $\{u_1, u_2, ..., u_b\}$. This implies that we need to add into the forwarding set of Λ_o all the settled outgoing neighbor nodes of v in Λ_o (i.e. $u_1, u_2, ..., u_c$) (Line 17). After that, $\{u_1, u_2, ..., u_{c+1}\}, \{u_1, u_2, ..., u_{c+2}\}$... will be checked by Lines 10-15, since $u_{c+1}, u_{c+2}, ...u_b$ are settled in the increasing order of their optimal DADs. By Lemma 14, if $(c + 1) \leq b$, the DAD of v via $\{u_1, u_2, ..., u_{c+1}\}$ is always smaller than that via $\{u_1, u_2, ..., u_c\}$. Thus the condition in Line 12 evaluates true and the forwarding set of Λ_o is updated to $\{u_1, u_2, ..., u_{c+1}\}$. This procedure is repeated until the forwarding set of Λ_o is updated to ψ. So far, we have proved that Λ_o and ψ can be constructed by Algorithm 3. However, how to find Λ_o among all the forwarding sectors constructed by Algorithm 3? After updating the forwarding sector records of v_a, Line 18 computes the currently best DAD of v_a by comparing the DADs via all the forwarding sectors which have been constructed. Since the DADs via non-optimal forwarding sectors cannot be smaller than that via Λ_o, Line 18 can find Λ_o and ψ when the

forwarding set of Λ_o is updated to ψ. Recall that since there might exist multiple OMFOFSs and optimal forwarding sectors, Algorithm 3 finds the optimal forwarding sector, whose forwarding set is updated to its MFOFS earliest. In brief, when v_a is settled, if all the nodes in ψ have been settled, ψ and Λ_o must have been found by Algorithm 3. This implies that $\mathbb{D}(v_a) = \theta(v_a)$, which contradicts $\mathbb{D}(v_a) > \theta(v_a)$. Thus when v_a is inserted into L, all the OMFOFSs of v_a contain at least one unsettled node.

We arbitrarily select one of these nodes, denoted by j. By Lemma 15, we have $\theta(j) < \theta(v_a)$. Since we assume $\mathbb{D}(v_a) > \theta(v_a)$, we must have $\theta(j) < \mathbb{D}(v_a)$. Let us consider the full shortest anypath P_j from j to t. Without loss of generality, assume that node j_1 has the smallest optimal DAD to t among all the unsettled nodes on P_j. Thus $\theta(j) \geq \theta(j_1)$. We **claim** (1) *all the nodes in the OMFOFS of* j_1 *along* P_j *must have been settled when* v_a *is added into* L. To prove this claim, let us assume that j_2, which is in this OMFOFS, is unsettled. By Lemma 15, we know that $\theta(j_1) > \theta(j_2)$. However, since we assume j_1 has the smallest optimal DAD (note that j_2 is also on P_j), then $\theta(j_1) \leq \theta(j_2)$, which contradicts $\theta(j_1) > \theta(j_2)$. Thus, all the nodes in this OMFOFS are settled. Since v_a is also in L, we next consider whether v_a is in this OMFOFS.

We **claim** (2) v_a *is not in the OMFOFSs of* j_1. We also prove this claim by contradiction. Assume that v_a is in one of its OMFOFSs. By Lemma 15, we therefore know $\theta(j_1) > \theta(v_a)$. On the other hand, since we have deduced that $\theta(j_1) \leq \theta(j)$ and $\theta(j) < \theta(v_a)$, we know $\theta(j_1) < \theta(v_a)$, which contradicts $\theta(j_1) > \theta(v_a)$. Thus v_a is not in the OMFOFSs of j_1.

The *two claims* above imply that at the time just before v_a is settled, all the nodes in one of the OMFOFSs of j_1 have been settled. Recall how we proved that if all the nodes in one OMFOFS of v_a have been added into L, this set must

be found. We can use the same proof to prove that an OMFOFS of j_1 must have been found. Thus, at that time $\mathbb{D}(j_1) = \theta(j_1)$.

We now derive the contradiction based on our first assumption that v_a is the first settled node for which $\mathbb{D}(v_a) > \theta(v_a)$. Since we have deduced $\theta(j_1) \leq \theta(j)$ and $\theta(j) < \mathbb{D}(v_a)$, we have $\theta(j_1) < \mathbb{D}(v_a)$. Additionally, we deduced $\mathbb{D}(j_1) = \theta(j_1)$. We therefore know $\mathbb{D}(j_1) < \mathbb{D}(v_a)$. However this is a contradiction, since this inequality implies that j_1, which is a node outside of L at the time v_a is inserted into L, should be inserted into L before v_a.

We hence conclude that for each node v in L, we have $\mathbb{D}(v) = \theta(v)$. This implies that when a node is settled, its shortest directional anypath has been found by Algorithm 3. ■

Proof of Theorem 11. Lemma 17 has proved the optimality of Algorithm 3. We now analyze its running time. Assuming that Q is a Fibonacci heap [21], each of the Extract-Min operations in Line 6 takes $O(\log |V|)$ time, with a total of $O(|V| \log |V|)$. Note that the for-loop of Lines 8-19 is executed $O(|E|)$ times in total, since there are a total number of $|E|$ incoming neighbors of all the nodes. For each unsettled incoming neighbor v, the for-loop of Lines 9-16 is executed $O(\Delta(v))$ times, since Algorithm 3 constructs $O(\Delta(v))$ forwarding sectors for node v. If we store some other status variables as in [58], computing D in Line 11 takes a constant time. Thus Lines 11-13 take constant time. The for-loop of Lines 9-16 therefore takes $O(\Delta(v))$ time, with a total of $O(\Delta(G)|E|)$. In Line 17, to obtain the anypath distance requires first sorting all the settled outgoing neighbors of v located in the forwarding sector in the increasing order of their optimal DADs, which takes $O(\Delta(v) \log \Delta(v))$ time [21], and then $\Delta(v)$ calculations. Obviously, it dominates the other operations in this line. Thus Line 17 takes $O(\Delta(v) \log \Delta(v))$ time, with a total of $O(|E|\Delta(G) \log \Delta(G))$ time. Since each node keeps $O(\Delta(v))$ records,

Line 18 takes $O(\Delta(v))$ time to finish comparison operations and $O(\log|V|)$ time to update priority queue Q [21], with a total of $O((\Delta(G)+\log|V|)|E|)$. Thus the total running time is $O(|E|(\log|V|+\Delta(G)\log\Delta(G)))$. ■

5.6 Routing Algorithm for VBDAR

5.6.1 Algorithm Description

In this section we present an efficient algorithm V-DART for VBDAR. In addition to the data structures used in Algorithm 3, for every node we keep another variable $\mathbb{A}_b(v)$ to store the beamwidth of the currently computed best forwarding sector. Each element (record) in $\mathbb{R}(v)$ has another component A_b to denote the beamwidth of the forwarding sector of this record. We say a node u is in the *extendible area* of a forwarding sector of v, if this forwarding sector covers u, or can cover u by rotating its start boundary clockwise (we say that u is in the *start boundary extendible area* (SBEA)) or end boundary counterclockwise (we say that u is in the *end boundary extendible area* (EBEA)) without exceeding the beamwidth constraint $A_u(v)$.

Actually the high-level framework of Algorithm 4 is similar to that of Algorithm 3. The difference is that for a forwarding sector Algorithm 4 tries out all the useful combinations of the start boundary and the end boundary rather than just all the possible start boundaries as in Algorithm 3. We briefly describe how Algorithm 4 works. Lines 1-4 initialize all the data structures, and set $\mathbb{D}(t)$ to 0. In the main while-loop, each time Line 6 extracts a node with the minimum $\mathbb{D}$ from Q, and inserts it into L. If its $\mathbb{D}$ is ∞ already, all the nodes still in Q cannot reach t and thus the algorithm terminates. In the first iteration, t is extracted. Lines 8-26 then perform relaxations for all its incoming neighbors. Since now for each incoming neighbor v, its $\mathbb{R}(v)$ is empty, Lines 9-19 will be skipped. Then

Algorithm 4 V-DART

1: **for** each node v in V **do**
2: $\mathbb{D}(v) \leftarrow \infty$, $\mathbb{A}_s(v) \leftarrow 0$, $\mathbb{A}_b(v) \leftarrow A_l(v)$, $\mathbb{F}(v) \leftarrow \emptyset$, $\mathbb{R}(v) \leftarrow \emptyset$.
3: **end for**
4: $\mathbb{D}(t) \leftarrow 0$, $L \leftarrow \emptyset$, $Q \leftarrow V$.
5: **while** $Q \neq \emptyset$ **do**
6: $u \leftarrow$ Extract-Min(Q), $L \leftarrow L \cup \{u\}$.
7: **if** $\mathbb{D}(u) = \infty$ **then** break.
8: **for** each unsettled incoming-neighbor v **do**
9: **for** each record R in $\mathbb{R}(v)$ **do**
10: **if** u is in the *extendible* area of the forwarding sector of R **then**
11: $F \leftarrow R.F \cup \{u\}$, $D \leftarrow \frac{1}{p(v,F)} + \mathbb{D}(F)$.
12: **if** $D < R.D$ **then**
13: $R.F \leftarrow F$, $R.D \leftarrow D$.
14: **if** u is not in the forwarding sector of R **then**
15: **if** u is in the SBEA **then**
16: construct a new record R' by copying R, rotate its start boundary clockwise such that u is on the start boundary, and update $R'.A_b$ and $R'.A_s$.
17: **end if**
18: **if** u is in the EBEA **then**
19: construct a new record R' by copying R, rotate its end boundary counterclockwise such that u is on the end boundary, and update $R'.A_b$.
20: **end if**
21: **end if**
22: **end if**
23: **end if**
24: **end for**
25: Construct two records for v with forwarding sector beamwidth of $A_l(v)$ such that u is on one's start boundary and the other's end boundary. For each record, set the forwarding set to the set of all the settled outgoing neighbor nodes of v in its forwarding sector, and calculate the corresponding D.
26: $\mathbb{D}(v) \leftarrow \min_{R \in \mathbb{R}(v)} R.D$; select the one with the smallest beamwidth among all the forwarding sectors which can provide $\mathbb{D}(v)$, and update the corresponding $\mathbb{A}_s(v)$, $\mathbb{F}(v)$ and $\mathbb{A}_b(v)$; update Q.
27: **end for**
28: **end while**

based on t Line 25 constructs two forwarding sectors for v, and two corresponding records which store the information of these two forwarding sectors. Then for v, Line 26 chooses the one with minimum beamwidth among all the forwarding sectors which can provide the shortest DAD, and updates the corresponding data structures. Algorithm 4 repeats this procedure. If for an incoming neighbor v of a currently settled node u, its $\mathbb{R}(v)$ is not empty, Lines 9-20 check for each existing forwarding sector of v whether u is in its extensible area and whether adding u into its forwarding set can lead to a smaller D. If so, Algorithm 4 adds u into the forwarding set and extends the forwarding sector if u is not in it. Since the original forwarding sector could be used later, Line(s) 16 and/or 19 make(s) a copy of the original one.

5.6.2 Algorithm Analysis

Theorem 12. *Algorithm 4 can find a shortest directional anypath for VBDAR in* $O(|E|(\log|V| + \Delta(G)^2))$ *time.* □

We need to prove Lemmas 18-19 before proving Theorem 12. Obviously, Lemmas 13-15 still hold for Algorithm 4.

Lemma 18. *Although for node v there exist an infinite number of possible forwarding sectors, in order to find the optimal one, it is sufficient to check* $O(\Delta(v)^2)$ *particular forwarding sectors.* □

Proof. We construct the $O(\Delta(v)^2)$ forwarding sectors as follows. First, for each outgoing neighbor of v, we construct two forwarding sectors centered at v with beamwidth $A_l(v)$ such that this node is on one's start boundary and the other's end boundary. Second, for each pair (denoted by $\{u_1, u_2\}$) of outgoing neighbors of v, we construct two forwarding sectors of v such that u_1 is on one's start

boundary and the other's end boundary, and u_2 is on one's end boundary and the other's start boundary, if the beamwidths of these two forwarding sectors are in the range of $[A_l(v), A_u(v)]$. Third, since the first step and the second step could construct the same forwarding sectors, we delete the redundant forwarding sectors. We therefore constructed $O(\Delta(v)^2)$ forwarding sectors, called *forwarding sector candidates*. We now consider an arbitrary forwarding sector Λ' of v with beamwidth in the range of $[A_l(v), A_u(v)]$. We will prove that the DAD via Λ' cannot be smaller than that via one particular forwarding sector candidate (denoted by Λ), and that the beamwidth of Λ cannot be larger than that of Λ'. We first rotate Λ' counterclockwise (or clockwise) until its start boundary (or end boundary) touches an outgoing neighbor node of v. Then we rotate the end boundary of Λ' clockwise (or the start boundary counterclockwise) until the beamwidth of Λ' reaches $A_l(v)$ or it touches an outgoing neighbor node of v. Λ is therefore constructed. We have three observations: 1)Λ is a forwarding sector candidate; 2) the beamwidth of Λ is not larger than that of Λ'; 3) all the outgoing neighbor nodes of v in Λ' are also covered by Λ. Thus by (5.1) the DAD via Λ' cannot be smaller than that via Λ. Thus we do not need to check Λ'. This implies that it is sufficient to check all the forwarding sector candidates. ■

Consider an OMFOFS ψ of a node v and its corresponding optimal forwarding sector. As in the proof of Lemma 18, we rotate it counterclockwise until its start boundary touches a node in ψ, and then rotate its end boundary clockwise until it touches a node in ψ or its beamwidth reaches $A_l(v)$. We thus obtain a forwarding sector candidate, denoted by Λ_o. Obviously, it is an optimal forwarding sector. As in Section 5.5, we define the *start boundary construct node (SBCN)* of Λ_o, which is the one with the smallest optimal DAD, among all the nodes in ψ, which are located on the start boundary of Λ_o.

Lemma 19. *Let v be any node that can reach the destination t. The anypath from v to t computed by Algorithm 4 is a shortest directional anypath.* □

Proof. The proof is similar to the proof of Lemma 17 except the part of proving the OMFOFS can be found when all the nodes in the OMFOFS are settled. We hence concentrate on this part and omit the proof for the other parts.

We claim that *an OMFOFS ψ of v and its corresponding optimal forwarding sector Λ_o must be found if all the nodes in ψ are settled in the increasing order of their optimal DADs*. Recall how we constructed Λ_o above.

We prove this claim by proving a series of sub-claims in a recursive manner. By Lemma 18, we know we only need to check the forwarding sector candidates. We know ψ and Λ_o must exist. Let $\{u_1, u_2, ..., u_z\}$ denote the set of all the outgoing neighbors of v in Λ_o, with $\theta(u_1) \leq \theta(u_2) \leq \cdots \leq \theta(u_z)$. By Lemma 13, we know ψ is of the form $\{u_1, u_2, ..., u_b\}$.

We first consider the SBCN of Λ_o, denoted by u_{c_1}. When it is settled, Line 25 constructs a forwarding sector (denoted by Λ_a) of beamwidth A_l with u_{c_1} on its start boundary. Recall that when u_{c_1} is settled, $u_1, u_2, ..., u_{c_1-1}$ have been settled. Now we consider two cases. The **first case** is that $u_1, u_2, ..., u_{c_1-1}$ are located in Λ_a. Line 25 adds $u_1, u_2, ..., u_{c_1}$ into the forwarding set of Λ_a. After that when $u_{c_1+1}, u_{c_1+2}, ..., u_b$ are settled in order, by Lemma 14 Lines 12-13 will add them into the forwarding set of Λ_a one by one until ψ is constructed. Note that Λ_a could be extended counterclockwise by Line 19 if some newly settled nodes are not in it. The **second case** is that not all $u_1, u_2, ..., u_{c_1-1}$ are located in Λ_a. Among them we select the counterclockwise farthest node from u_{c_1}(measured by angle), denoted by u_{c_2}. If there is more than one such node, we select the one with the minimum optimal DAD. ***Sub-claim (1)***: *Algorithm 4 must have*

constructed a forwarding sector Λ_1 *with* u_{c_1} *on its start boundary and* u_{c_2} *on its end boundary, whose forwarding set is* $\{u_1, u_2, ..., u_{c_1}\}$. If this is true, after that when $u_{c_1+1}, u_{c_1+2}, ..., u_b$ are settled, by Lemma 14 Lines 12-13 will add them into the forwarding set of Λ_1 one by one until ψ is constructed, and Λ_1 will keep being extended by Lines 18-19 until it is updated to Λ_o. *In brief, if sub-claim (1) is true,* ψ *and* Λ_o *will be constructed by Algorithm 4.*

Now we consider u_{c_2}. When u_{c_2} is settled, Line 25 constructs a forwarding sector (denoted by Λ_b) of beamwidth A_l with u_{c_2} on its end boundary. Similar to the discussion before, we consider two cases. The **first case** is that $u_1, u_2, ...u_{c_2-1}$ are located in Λ_b. Line 25 adds $u_1, u_2, ...u_{c_2}$ into the forwarding set of Λ_b. After that when $u_{c_2+1}, u_{c_2+2}, ..., u_{c_1}$ are settled, by Lemma 14 Lines 12 -13 will add them into the forwarding set of Λ_b one by one, and Λ_b will keep being extended by Lines 15-16. Eventually Λ_b will have u_{c_1} on its start boundary and u_{c_2} on its end boundary, and its forwarding set is $\{u_1, u_2, ..., u_{c_1}\}$. In other words, the current Λ_b is Λ_1. Thus sub-claim (1) is true in this case. The **second case** is that not all $u_1, u_2, ...u_{c_2-1}$ are located in Λ_b. Among them we select the clockwise farthest node from u_{c_2} (measured by angle), denoted by u_{c_3}. ***Sub-claim (2)***: *Algorithm 4 must have constructed a forwarding sector* Λ_2 *with* u_{c_3} *on its start boundary and* u_{c_2} *on its end boundary, whose forwarding set is* $\{u_1, u_2, ...u_{c_2}\}$. If sub-claim (2) is true, similar to the discussion before, after that when $u_{c_2+1}, u_{c_2+2}, ..., u_{c_1}$ are settled, by Lemma 14 Lines 12 -13 will add them into the forwarding set of Λ_2 one by one, and Λ_2 will keep being extended in Lines 15-16 until it is updated to Λ_1. *In brief, if sub-claim (2) is true, sub-claim (1) will be true.*

This "zigzag" procedure is repeated until we consider such a node u_{c_x} that $u_1, u_2, ..., u_{c_x}$ are all located in a forwarding sector Λ_x, where Λ_x is a forwarding sector of beamwidth A_l and has u_{c_x} on its start boundary (if x is odd) or its

end boundary (if x is even). This node must exist, since at least u_1 satisfies this. When u_{c_x} is settled, Line 25 constructs Λ_x and adds $u_1, u_2, ..., u_{c_x}$ into its forwarding set. Now consider the sub-claim associated with $u_{c_{x-1}}$. ***Sub-claim*** $(x-1)$: *Algorithm 4 must have constructed a forwarding sector* Λ_{x-1} *with* $u_{c_{x-1}}$ *on its start boundary and* u_{c_x} *on its end boundary (if* x *is even) or with* u_{c_x} *on its start boundary and* $u_{c_{x-1}}$ *on its end boundary (if* x *is odd), whose forwarding set is* $\{u_1, u_2, ...u_{c_{x-1}}\}$. Now come back to consider Λ_x. Without loss of generality, we assume x is odd. After u_{c_x} is settled, $u_{c_x+1}, u_{c_x+2}, ..., u_{c_{x-1}}$ will be settled in order. Thus by Lemma 14 Lines 12-13 will add them into the forwarding set of Λ_x one by one, and Λ_x will keep being extended in Lines 18-19. Eventually, Λ_x becomes a forwarding sector with u_{c_x} on its start boundary and $u_{c_{x-1}}$ on its end boundary, whose forwarding set is $\{u_1, u_2, ...u_{c_{x-1}}\}$. In other words, the current Λ_x is Λ_{x-1}. Thus **sub-claim** $(x-1)$ **is proved**. Repeat this procedure and sub-claim (1) is hence correct. Recall that if sub-claim (1) is true, ψ and Λ_o must be constructed. Moreover, Λ_o either has a beamwidth of A_l, or has u_{c_1} on its start boundary and a node on its end boundary. Thus its beamwidth cannot be reduced anymore.

How to find Λ_o among all the forwarding sectors constructed by Algorithm 4? Like Algorithm 3, after updating the forwarding sector records, Line 26 computes the currently best DAD by comparing the DADs via all the forwarding sectors which have been constructed. Since the DADs via non-optimal forwarding sectors cannot be smaller than the optimal, Line 26 can find the optimal DAD. Furthermore, if there exist multiple forwarding sectors associated with this shortest DAD, Line 26 chooses the one with the minimum beamwidth and thus finds the optimal forwarding sector. The claim is hence proved. ■

Proof of Theorem 12. Lemma 19 has proved the optimality of Algorithm 4. The time complexity analysis is similar to that of Algorithm 3 and is omitted. ■

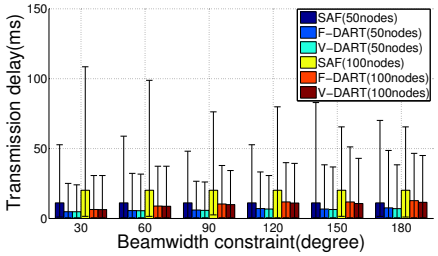
5.7 Performance Evaluation

In this subsection, we evaluate the performance of DART. Since DART represents the first attempt towards the practical design for directional anypath routing, we compare it with the shortest anypath routing proposed in [58] for omnidirectional antenna, which presented the Shortest Anypath First (SAF) algorithm to compute the shortest anypath. In the implementations of F-DART and V-DART, all the nodes adopted directional transmission, and in the implementation of SAF, all the nodes adopted omnidirectional transmission.

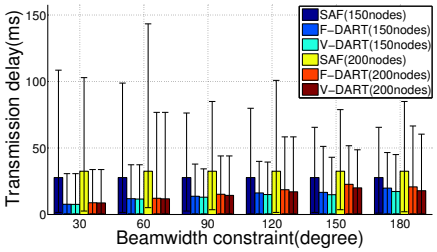
In the simulation, we uniformly distributed 802.11 nodes in a $1000m \times 1000m$ square region. The numbers of nodes were chosen to be 50, 100, $\cdots$, 300. The transmit rate of each node was set to $2Mbps$, and both the directional and omnidirectional transmission ranges were set to $200m$. As in [27], we assume that the PDR is inversely proportional to the distance with a random Gaussian deviation of 0.1. For each network size, we evaluated 6 test cases by choosing 6 different *beamwidth constraints*: $\frac{\pi}{6}, \frac{\pi}{3}, ...\pi$. For example, when the constraint was $\frac{\pi}{6}$, the beamwidths of transmitters in the implementation of F-DART and A_u in the implementation of V-DART were set to $\frac{\pi}{6}$. A_l was always set to $\frac{A_u}{6}$. For each test case, we randomly generated 100 subcases by randomly choosing a source-destination pair and the coordinates of the nodes. In each subcase, the source sent 100 packets of size 1024 bytes to its destination. Thus the results were averaged over 10,000 packets. All tests were performed on a 1.8GHz Linux PC with 2G bytes of memory. We simulated a random network traffic pattern. More specifically, the nodes, which were not on the anypath returned by DART or SAF, carried random loads, and their beamwidths in the implementation of F-DART and V-DART were set to the beamwidth constraint.

Since DART looks for the shortest directional anypath, we study the improvement on the average packet transmission delay from the source to the destination in our experiments. From Fig.5.6a-5.6c, we can make the following observations. When the network is exploiting directional antennas, the packet transmission delay is reduced by 28%-70%. This significant improvement clearly shows the importance of the directional communication and justifies the necessity of directional anypath routing. Additionally, we can observe that as the beamwidth constraint increases, the average packet transmission delay also increases. For the implementation of F-DART, this is because, with the increase of the beamwidth constraint, the beamwidths of all the nodes also increase, and as a result the mutual interference increases. For the implementation of V-DART, although the nodes on the anypath from the source to the destination can adjust their beamwidths to reduce the interference, considering the simulation setup that the beamwidths of the other nodes are increased with the increase of the beamwidth constraint, the packet transmission delay still increases. However, note that with the increase of the beamwidth constraint, the gap between the packet transmission delays of F-DART and V-DART is enlarged up to 20%. This is expected, because V-DART can keep using smaller beamwidths to decrease interference.

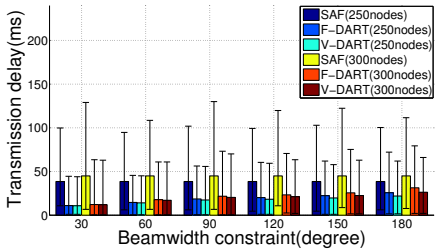
Fig.5.7 compares the running times of computing a source-destination anypath when the beamwidth constraint is set to $\frac{\pi}{2}$. Although F-DART and V-DART take more time, their running time is still satisfactory in practice. In all cases studied, the average running times of F-DART and V-DART are no more than 30ms and 140ms, respectively. Considering the significant improvement on the packet transmission delay, the sacrificed negligible running time is worthwhile.



(a) Transmission delay(50/100nodes)



(b) Transmission delay(150/200nodes)



(c) Transmission delay(250/300nodes)

Figure 5.6: Simulation results

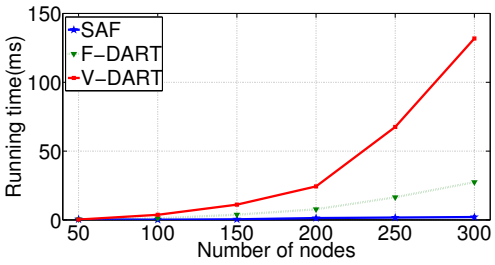


Figure 5.7: Running time

5.8 Conclusion

In this work, we have proposed DART, a cross-layer design for anypath routing in wireless networks with directional antennas. For the MAC layer, we have presented a directional anycast MAC, which is an enhancement to the IEEE 802.11 MAC protocol, making DART suitable for integration into current systems. For the routing layer, we have proposed two polynomial time routing algorithms based on two antenna models, and have proved their optimality. Simulation results show that DART can significantly reduce the packet transmission delay.

Chapter 6

Conclusion and Future Work

In this dissertation, we have concentrated on optimization for resource-constrained wireless networks, and have studied two fundamental resource-allocation problems: 1) distributed routing optimization and 2) anypath routing optimization.

6.1 Optimizing Distributed Routing for Multihop Wireless Networks

6.1.1 Conclusion

The study on the distributed routing optimization problem is composed of two main thrusts, targeted at understanding distributed routing and resource optimization for multihop wireless networks. The first thrust is dedicated to understanding the impact of full-duplex transmission on wireless network resource optimization. We have proposed two provably good distributed algorithms to optimize the resources in a full-duplex wireless network. The second thrust is dedicated to understanding the influence of network entity load constraints on network resource allocation and routing computation. We have proposed a provably good distributed algorithm to allocate wireless resources. In addition, we have proposed a new subgradient optimization framework, which can provide find-grained convergence, optimality, and dual space information at each iteration. This framework can provide a useful theoretical foundation for many networking optimization problems.

6.1.2 Future Work

As for the future work, one interesting research task is to investigate the impact of full-duplex radios on end-to-end delay. The constraints and problem formulation used before might be still applicable since our problem is formulated from a

perspective of stable network flows. However, this formulation cannot effectively capture the delay improvement that can be brought by using full-duplex radios. We expect that the network delay reduction is a non-decreasing function of the percentage of the nodes equipped with full-duplex radios. Therefore, it would be interesting to exploit the potential impact of full-duplex radios on end-to-end delay.

Another interesting research topic is extending our preliminary results described above to the mixed-duplex and multi-channel case. Some of the constraints we used before may still hold, but we need to specifically handle the problems in the MAC layer once multiple channels are being used.

6.2 Optimizing Anypath Routing for Multihop Wireless Networks

6.2.1 Conclusion

This study on the anypath routing optimization problem is composed of two main thrusts. The first thrust is dedicated to understanding the computational complexity of multi-constrained anypath routing and designing approximate solutions. We have shown that the problem is NP-hard when the number of constraints is larger than one. We have presented two polynomial time K-approximation algorithms. One is a centralized algorithm while the other one is a distributed algorithm. For the second thrust, we have studied directional anypath routing and present DART, a cross-layer design of MAC and routing layers, which represents the first attempt towards the practical design for directional anypath routing. For the MAC layer, we have presented a directional anycast MAC, an enhancement to the 802.11 MAC. For the routing layer, we have proposed two polynomial time routing algorithms to compute directional anypaths based on two antenna models, and proved their optimality based on the packet delivery ratio metric.

6.2.2 Future Work

As for the future work, possible future research topics along the line of research on multi-constrained anypath routing include either designing better approximation algorithms or establishing stronger hardness results. This is because our NP-hardness proof proves the problem to be weakly NP-hard, as opposed to strongly NP-hard.

A possible future research topic along the line of research on directional anypath routing is designing distributed algorithms. Although we have presented two centralized routing algorithms, in practice distributed algorithms might be preferable. Essentially speaking, our routing algorithms are Dijkstra-like algorithms. We strongly believe that our algorithms can be converted to Bellman-Ford-like algorithms. In this way, it is likely that we can decentralize the routing algorithms.

[1] A. Abdulla, H. Nishiyama, and N. Kato. Extending the lifetime of wireless sensor networks: A hybrid routing algorithm. *Computer Communication*, 35(9):1056–1063, 2012.

[2] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the ACM SIGCOMM*, pages 121–132, 2004.

[3] M. Alicherry, R. Bhatia, and L. E. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of the ACM MOBICOM*, pages 58–72, 2005.

[4] S. Basagni. Finding a maximal weighted independent set in wireless networks. *Telecommunication Systems*, 18:155–168, 2001.

[5] R. Bhatia, A. Segall, and G. Zussman. Analysis of bandwidth allocation algorithms for wireless personal area networks. *Wireless Networks*, 12(5):589–603, 2006.

[6] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *Proceedings of the ACM SIGCOMM*, pages 133–144, 2005.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[8] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu. Joint asynchronous congestion control and distributed scheduling for multihop wireless networks. *IEEE/ACM Transactions on Networking*, 2008.

[9] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of the ACM SIGCOMM*, pages 169–180, 2007.

[10] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of the IEEE INFOCOM*, pages 22 –31, 2000.

[11] J.-H. Chang and L. Tassiulas. Fast approximate algorithms for maximum lifetime routing in wireless ad-hoc networks. In *Proceedings of the IFIP-TC6 / European Commission International Conference on Broadband Communications, High Performance Networking, and Performance of Communication Networks*, pages 702–713, 2000.

[12] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar. Throughput and fairness guarantees through maximal scheduling in wireless networks. *IEEE Transactions on Information Theory*, 54(2):572 –594, 2008.

[13] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle. Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 1 –13, 2006.

[14] S. Chen and K. Nahrstedt. On finding multi-constrained paths. In *Proceedings of the IEEE ICC*, pages 874 –879, 1998.

[15] S. Chen and K. Nahrstedt. An overview of quality of service routing for next-generation high-speed networks: problems and solutions. *IEEE Network*, 12(6):64 –79, 1998.

[16] M. Chiang, S. Low, A. Calderbank, and J. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255 –312, 2007.

[17] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *Proceedings of the ACM MOBICOM*, pages 1–12, 2010.

[18] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya. Using directional antennas for medium access control in ad hoc networks. In *Proceedings of the ACM MOBICOM*, pages 59–70, 2002.

[19] Cisco. Cisco aironet antennas and accessories reference guide.

[20] Cisco. Cisco aironet 802.11a/b/g wireless lan client adapters (CB21AG and PI21AG) installation and configuration guide. 2007.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms. 2001.

[22] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the ACM MOBICOM*, pages 134–146, 2003.

[23] M. Duarte and A. Sabharwal. Full-duplex wireless communications using off-the-shelf radios: Feasibility and first results. In *2010 Conference Record of*

the Forty Fourth Asilomar Conference on Signals, Systems and Computers, pages 1558 –1562, 2010.

[24] H. Dubois-Ferrieàndre, M. Grossglauser, and M. Vetterli. Valuable detours: Least-cost anypath routing. *IEEE/ACM Transactions on Networking*, 19(2):333 –346, 2011.

[25] H. Dubois-Ferrière. Anypath routing. *Ph.D dissertation, EPFL*, 2006.

[26] A. Eryilmaz and R. Srikant. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1514 –1524, 2006.

[27] X. Fang, D. Yang, P. Gundecha, and G. Xue. Multi-constrained anypath routing in wireless mesh networks. In *Proceedings of the IEEE SECON*, pages 1 –9, 2010.

[28] X. Fang, D. Yang, and G. Xue. Consort: Node-constrained opportunistic routing in wireless mesh networks. In *Proceedings of the IEEE INFOCOM*, pages 1907 –1915, 2011.

[29] F. Forgo, J. Szep, and F. Szidarovszky. Introduction to the theory of games. *Kluwer Academic*, 1999.

[30] Y. Ganjali and A. Keshavarzian. Load balancing in ad hoc networks: single-path routing vs. multi-path routing. In *Proceedings of the IEEE INFOCOM*, pages 1120 – 1125, 2004.

[31] J. Gao and L. Zhang. Load-balanced short-path routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(4):377 – 388, 2006.

[32] A. Goldsmith. Wireless communications. *Cambridge University*, 2005.

[33] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. In *Proceedings of the ACM SIGCOMM*, pages 159–170, 2008.

[34] S. Guo and O. W. Yang. Antenna orientation optimization for minimum-energy multicast tree construction in wireless ad hoc networks with direc-

tional antennas. In *Proceedings of the ACM MOBIHOC*, pages 234–243, 2004.

[35] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. *IEEE/ACM Transactions on Networking*, 17(6):1846–1859, 2009.

[36] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388 –404, 2000.

[37] B. Hajek and G. Sasaki. Link scheduling in polynomial time. *IEEE Transactions on Information Theory*, 34(5):910 –917, 1988.

[38] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *Proceedings of the ACM MOBICOM*, pages 339–350, 2008.

[39] M. Harchol-Balter, T. Leighton, and D. Lewin. Resource discovery in distributed networks. In *Proceedings of the ACM PODC*, pages 229–237, 1999.

[40] J. Hoepman. Simple distributed weighted matchings. *http://arxiv.org/abs/cs/0410047*.

[41] IEEE. IEEE Standard 802.11, http://standards.ieee.org/getieee802/download/802.11-2007.pdf.

[42] J. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14(1):95 – 116, 2006.

[43] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the ACM MOBICOM*, pages 66–80, 2003.

[44] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha. Practical, real-time, full duplex wireless. In *Proceedings of the ACM MOBICOM*, pages 301–312, 2011.

[45] R. Jain, D.-M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *CoRR*, 1998.

[46] S. Jain and S. R. Das. Exploiting path diversity in the link layer in wireless ad hoc networks. *Ad Hoc Networks*, 6(5):805–825, 2008.

[47] L. Jang-Won, T. Ao, H. Jianwei, M. Chiang, and A. Robert. Reverse-engineering mac: A non-cooperative game model. *IEEE Journal on Selected Areas in Communications*, 25(6):1135 –1147, 2007.

[48] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[49] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose atm switch chip. *IEEE Journal on Selected Areas in Communications*, 9(8):1265 –1279, 1991.

[50] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level network coding for wireless mesh networks. In *Proceedings of the ACM SIGCOMM*, pages 401–412, 2008.

[51] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8(1):33–37, 1997.

[52] M. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *Proceedings of the ACM MOBICOM*, pages 42–54, 2003.

[53] T. Korkmaz and M. Krunz. A randomized algorithm for finding a path subject to multiple QoS requirements. *Computer Networks*, 36(2-3):251–268, 2001.

[54] D. Koutsonikolas, C.-C. Wang, and Y. Hu. Efficient network-coding-based opportunistic routing through cumulative coded acknowledgments. *IEEE/ACM Transactions on Networking*, 19(5):1368 –1381, 2011.

[55] F. Kuipers, A. Orda, D. Raz, and P. Van Mieghem. A comparison of exact and e-approximation algorithms for constrained routing. In *Proceedings of the 5th international IFIP-TC6 conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 197–208, 2006.

[56] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Communications Magazine*, 40(12):50 – 55, 2002.

[57] S. Kwon and N. Shroff. Paradox of shortest path routing for large multihop wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 1001 –1009, 2007.

[58] R. Laufer, H. Dubois-Ferriere, and L. Kleinrock. Multirate anypath routing in wireless mesh networks. In *Proceedings of the IEEE INFOCOM*, pages 37 –45, 2009.

[59] R. Laufer and L. Kleinrock. Multirate anypath routing in wireless mesh networks. *UCLA Computer Science Department, Tech. Rep. UCLA-CSD-TR080025*, 2008.

[60] M. Leconte, J. Ni, and R. Srikant. Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks. *IEEE/ACM Transactions on Networking*, 19(3):709–720, 2011.

[61] X.-Y. Li, Y. Wang, H. Chen, X. Chu, Y. Wu, and Y. Qi. Reliable and energy-efficient routing for static wireless ad hoc networks with unreliable links. *IEEE Transactions on Parallel and Distributed Systems*, 20(10):1408 –1421, 2009.

[62] X. Lin and N. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proceedings of the IEEE CDC*, volume 2, 2004.

[63] X. Lin and N. Shroff. The impact of imperfect scheduling on cross-layer rate control in wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 1804 – 1814, 2005.

[64] X. Lin and N. Shroff. Utility maximization for communication networks with multipath routing. *IEEE Transactions on Automatic Control*, 51(5):766 – 781, 2006.

[65] M. Liu, J. Cao, G. Chen, and X. Wang. An energy-aware routing protocol in wireless sensor networks. *Sensors*, 9(1):445–462, 2009.

[66] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28:213–219, 1999.

[67] S. Low. A duality model of tcp and queue management algorithms. *IEEE/ACM Transactions on Networking*, 11(4):525 – 536, 2003.

[68] S. Low and D. Lapsley. Optimization flow control. i. basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861 –874, 1999.

[69] M. Lu and J. Wu. Opportunistic routing algebra and its applications. In *Proceedings of the IEEE INFOCOM*, pages 2374 –2382, 2009.

[70] C. P. Luk, W. C. Lau, and O. C. Yue. Opportunistic routing with directional antennas in wireless mesh networks. In *Proceedings of the IEEE INFOCOM*, pages 2886 –2890, 2009.

[71] D. Lun, M. Medard, and R. Koetter. Network coding for efficient wireless unicast. In *International Zurich Seminar on Communications*, pages 74 –77, 2006.

[72] L. Ma, Q. Zhang, and X. Cheng. A power controlled interference aware routing protocol for dense multi-hop wireless networks. *Wireless Networks*, 14(2):247–257, 2008.

[73] X. Mao, S. Tang, X. Xu, X.-Y. Li, and H. Ma. Energy-efficient opportunistic routing in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(11):1934 –1942, 2011.

[74] A. Mei and J. Stefa. Routing in outer space: fair traffic load in multi-hop wireless networks. In *Proceedings of the ACM MOBIHOC*, pages 23–32, 2008.

[75] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *Proceedings of SIGMETRICS'06/Performance'06*, pages 27–38, 2006.

[76] A. Nasipuri, K. Li, and U. Sappidi. Power consumption and throughput in mobile ad hoc networks using directional antennas. In *Proceedings of the IEEE ICCCN*, pages 620 – 626, 2002.

[77] V. Navda, A. P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. Mobisteer: using steerable beam directional antenna for vehicular network access. In *Proceedings of MOBISYS*, pages 192–205, 2007.

[78] A. Nedić and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009.

[79] A. Orda and A. Sprintson. Precomputation schemes for QoS routing. *IEEE/ACM Transactions on Networking*, 11(4):578–591, 2003.

[80] D. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, pages 1439 –1451, 2006.

[81] D. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Transactions on Automatic Control*, 52(12):2254 –2269, 2007.

[82] G. Parissidis, M. Karaliopoulos, T. Spyropoulos, and B. Plattner. Interference-aware routing in wireless multihop networks. *IEEE Transactions on Mobile Computing*, 10(5):716 –733, 2011.

[83] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 90 –100, 1999.

[84] P. P. Pham and S. Perreau. Increasing the network performance using multi-path routing mechanism with load balance. *Ad Hoc Networks*, 2(4):433 –459, 2004.

[85] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica. Balancing traffic load in wireless networks with curveball routing. In *Proceedings of the ACM MOBIHOC*, pages 170–179, 2007.

[86] S. Prabhavat, H. Nishiyama, N. Ansari, and N. Kato. On load distribution over multipath networks. *IEEE Communications Surveys Tutorials*, (99):1–19, 2011.

[87] B. Radunovic, C. Gkantsidis, P. Key, and P. Rodriguez. An optimization framework for opportunistic multipath routing in wireless mesh networks. In *Proceedings of the IEEE INFOCOM*, pages 2252 –2260, 2008.

[88] B. Radunovic, D. Gunawardena, P. Key, A. Proutiere, N. Singh, V. Balan, and G. Dejean. Rethinking indoor wireless mesh design: Low power, low frequency, full-duplex. In *2010 Fifth IEEE Workshop on Wireless Mesh Networks*, pages 1 –6, 2010.

[89] L. Rao, X. Liu, J.-J. Chen, and W. Liu. Joint optimization of system lifetime and network performance for real-time wireless sensor networks. In *Quality of Service in Heterogeneous Networks*, volume 22 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 317–333. Springer Berlin Heidelberg, 2009.

[90] S. Roy, Y. C. Hu, D. Peroulis, and X.-Y. Li. Minimum-energy broadcast using practical directional antennas in all-wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 1 –12, 2006.

[91] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu. Soar: Simple opportunistic adaptive routing protocol for wireless mesh networks. *IEEE Transactions on Mobile Computing*, 8(12):1622 –1635, 2009.

[92] A. Sahai, G. Patel, and A. Sabharwal. Pushing the limits of full-duplex: Design and real-time implementation. *CoRR*, 2011.

[93] A. Sankar and Z. Liu. Maximum lifetime routing in wireless ad-hoc networks. In *Proceedings of the IEEE INFOCOM*, pages 1089 – 1097, 2004.

[94] G. Schaefer, F. Ingelrest, and M. Vetterli. Potentials of opportunistic routing in energy-constrained wireless sensor networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 118–133, 2009.

[95] S. Shakkottai and R. Srikant. Network optimization and control. *Found. Trends Networks*, 2(3):271–379, 2007.

[96] G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the complexity of scheduling in wireless networks. In *Proceedings of the ACM MOBICOM*, pages 227–238, 2006.

[97] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of the ACM MOBICOM*, pages 181–190, 1998.

[98] A. Spyropoulos and C. Raghavendra. Energy efficient communications in ad hoc networks using directional antennas. In *Proceedings of the IEEE INFOCOM*, pages 220 – 228, 2002.

[99] A. Subramanian, M. Buddhikot, and S. Miller. Interference aware routing in multi-radio wireless mesh networks. In *IEEE Workshop on Wireless Mesh Networks*, pages 55 –63, 2006.

[100] M. Takai, J. Martin, R. Bagrodia, and A. Ren. Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In *Proceedings of the ACM MOBIHOC*, pages 183–193, 2002.

[101] J. Tang, G. Xue, and W. Zhang. Interference-aware topology control and QoS routing in multi-channel wireless mesh networks. In *Proceedings of the ACM MOBIHOC*, pages 68–77, 2005.

[102] L. Tassiulas and S. Sarkar. Maxmin fair scheduling in wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 763 – 772, 2002.

[103] G. Tsaggouris and C. Zaroliagis. Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. *Theor. Comp. Sys.*, 45(1):162–186, 2009.

[104] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder. Efficient interference-aware tdma link scheduling for static wireless networks. In *Proceedings of the ACM MOBICOM*, pages 262–273, 2006.

[105] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228 –1234, 1996.

[106] J. Wu, M. Lu, and F. Li. Utility-based opportunistic routing in multi-hop wireless networks. In *International Conference on Distributed Computing Systems*, pages 470 –477, 2008.

[107] G. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman. Finding a path subject to many additive QoS constraints. *IEEE/ACM Transactions on Networking*, 15(1):201 –211, 2007.

[108] G. Xue and W. Zhang. Multiconstrained QoS routing: Greedy is good. In *Proceedings of IEEE GLOBECOM*, pages 1866 –1871, 2007.

[109] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman. Polynomial time approximation algorithms for multi-constrained QoS routing. *IEEE/ACM Transactions on Networking*, 16(3):656 –669, 2008.

[110] S. Yi, Y. Pei, and S. Kalyanaraman. On the capacity improvement of ad hoc wireless networks using directional antennas. In *Proceedings of the ACM MOBIHOC*, pages 108–116, 2003.

[111] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *IEEE/ACM Transactions on Networking*, 15(1):133 – 144, 2007.

[112] X. Yuan and X. Liu. Heuristic algorithms for multi-constrained quality of service routing. In *Proceedings of the IEEE INFOCOM*, pages 844–853, 2001.

[113] K. Zeng, W. Lou, and H. Zhai. On end-to-end throughput of opportunistic routing in multirate and multihop wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 816 –824, 2008.

[114] K. Zeng, Z. Yang, and W. Lou. Opportunistic routing in multi-radio multi-channel multi-hop wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 1 –5, 2010.

[115] X. Zhang and B. Li. Dice: a game theoretic framework for wireless multipath network coding. In *Proceedings of the ACM MOBIHOC*, pages 293–302, 2008.

[116] Z. Zhong, J. Wang, S. Nelakuditi, and G.-H. Lu. On selection of candidates for opportunistic anypath forwarding. *SIGMOBILE Mobile Computing and Communications Review*, 10(4):1–2, 2006.

[117] M. Zorzi and R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance. *IEEE Transactions on Mobile Computing*, 2(4):349 – 365, 2003.