DISTRIBUTED SOLUTIONS FOR RATE CONTROL AND MAXIMUM LIFETIME IN WIRELESS NETWORKS

By

LIANG ZHANG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

To my family

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Prof. Shigang Chen, for his constant guidance, support, and insightful advice throughout my graduate study. He is a terrific advisor, a passionate researcher and a critical thinker. Without the numerous discussions with him, the work presented in this dissertation would never have existed.

I am grateful to Prof. Sartaj Sahni, Prof. Randy Chow, Prof. Jonathan Liu, Prof. Tan Wong, and Prof. Liuqing Yang, for their instructive comments and support during my study. I would also like to thank all my colleagues in Prof. Chen's research group, including Zhan Zhang, MyungKeun Yoon, Ying Jian, Ming Zhang and Tao Li, for providing valuable feedback and high level of research support.

I would also like to take this chance to express my endless love to my wife Xiaojie Sun, my parents, and my brother. Without their love, understanding, encouragement and support, none of these would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

DISTRIBUTED SOLUTIONS FOR RATE CONTROL AND MAXIMUM LIFETIME IN WIRELESS NETWORKS

By

Liang Zhang

August 2009

Chair: Shigang Chen
Major: Computer Engineering

This study focuses on end-to-end flow rate fairness and lifetime fairness in wireless networks.

In recent years, the advent of multihop wireless networks has greatly accelerated the research on bandwidth management in such networks to support new applications. While much research concentrates on the MAC layer, the users perception on these networks is however determined mainly based on the networks end-to-end effectiveness. It is important for us to develop flexible tools for traffic engineering in multihop wireless networks. In this study, two solutions are proposed to achieve end-to-end maxmin flow rate fairness in such networks.

A cross-layer design is firstly proposed for achieving end-to-end maxmin fairness in wireless mesh networks. In this approach, a generalized maxmin model is first proposed for multihop wireless networks. At the network layer, our design allocates network capacity to end-to-end flows for maxmin bandwidth allocation. At the MAC layer, our design achieves the allocated bandwidth shares for flows through a two-level weighted fair queuing algorithm. The proposed design is able to equalize the end-to-end bandwidth allocation to competing flows that share common bottlenecks, while fully utilizing the network capacity. Results of simulations are presented to demonstrate the effectiveness of the proposed solution in enhancing end-to-end fairness.

We also propose a fully distributed solution that is compatible with IEEE 802.11 DCF for achieving end-to-end maxmin fairness. We transform the global maxmin objective to four local conditions and prove that, if the four local conditions are satisfied in the whole network, then the global maxmin objective must be achieved. We then design a distributed rate adaptation protocol based on the four conditions. Whenever a local condition is tested false at a node, the node informs the sources of certain selected flows to adapt their rates such that the condition can be satisfied. Comparing with previous work, our protocol has a number of advantages. First, it does not modify the backoff scheme of IEEE 802.11. Second, it replaces per-flow queueing with per-destination queueing. Packets from all flows to the same destination is queued together. Third and most important, our protocol achieves far better fairness (or weighted fairness) among end-to-end flows than previous work.

Wireless sensor networks have a wide range of applications in habitat observation, seismic monitoring, battlefield sensing, etc. As another type of multihop wireless network, a sensor network consists of battery-powered sensor nodes that are limited in energy supply. An important problem of wireless sensor networks is maximizing the operational lifetime of a sensor network. The lifetime of a sensor network is defined as the lifetimes of all sensors that produce useful data. A centralized solution proposed by previous work requires solving a sequence of linear programming problems. The computation overhead can be prohibitively high for large sensor networks. Collecting the complete information about the network and uploading the complete forwarding policies to all nodes require significant amount of transmissions, particularly for nodes around the sink. We propose a fully distributed progressive algorithm which iteratively produces a series of lifetime vectors, each better than the previous one. Instead of giving the optimal result in one shot after lengthy computation, the proposed distributed algorithm has a result at any time, and the more time spent gives the better result. We show that when the algorithm

stabilizes, its result produces the maximum lifetime vector. Furthermore, the algorithm is able to converge rapidly towards the maximum lifetime vector with low overhead.

CHAPTER 1
INTRODUCTION

The technology of wireless networking has been widely adopted due to its advantages on accessibility and portability. In a multihop wireless network, each node operates both as an end host and as a router, forwarding packets for other nodes that cannot communicate directly. Multihop wireless networks provide more flexibility as they operate in a decentralized and self-organizing manner and do not rely on fixed network infrastructure. In recent years, the advent of various multihop wireless networks, including wireless mesh networks and wireless sensor networks, has greatly intensified research on such networks to improve their applicability in practice.

1.1 End-to-End Flow Rate Fairness

A major problem of multihop wireless networks is to fairly allocate the scarce wireless bandwidth to all users. Much research concentrates on the MAC layer. Researchers have proposed some algorithms to achieve fair bandwidth allocation for *single-hop flows* while maximizing network throughput [26, 37]. "Fair" is defined differently in those algorithms. One common feature of those algorithms is that at least certain amount of bandwidth is guaranteed for every single-hop flow in the network.

The user's perception on multihop networks is however determined mainly based on the networks' end-to-end effectiveness. For example, for new users to participate in a wireless mesh network, they want to be sure that their end-to-end traffic is treated fairly as everyone else. Moreover, if a user contributes more to the network, she may demand that her traffic is given more weight than others' traffic. In order to meet diverse user requirements, it is important for us to develop flexible tools for traffic engineering in multihop wireless networks.

However, the solutions for single-hop flow fairness cannot be extended to achieve end-to-end flow fairness because they ignore the relationship among the subflows from the same multihop flow. If an upstream subflow is allocated more bandwidth than its

downstream subflow from the same multihop flow, the router in the middle will receive packets at a faster rate than it can forward. The buffer of the router may be overflowed by the continuously increasing packets. Among all subflows of a multihop flow, the one with the lowest rate becomes the bottleneck. Furthermore, the bandwidth consumed by dropped packets could be allocated to other flows to increase their throughput.

Previous work [32] has pointed out the above problem existing in multihop flows if simply applying the above single-hop flow algorithms to multihop flows by breaking each multihop flow into multiple single-hop flows. In [32], Li pointed out a relationship among all subflows of a multihop flow, which is that all subflows from the same multihop flow are expected to have the same rate and to receive equal amount of bandwidth. However, the basic fairness model proposed in [32] has serious limitation that hinders its applicability in WMNs. The model ensures a *basic share* of bandwidth for each end-to-end flow in a *contending flow group* and then tries to maximize the overall network throughput. The basic share is calculated as the channel capacity divided by the total effective length of the routing paths of the flows in the group. The effective length of a path is the smaller one of the path length and 3. Two flows, f_1 and f_n, belong to the same contending flow group if there exists a sequence of flows, f_2 through f_{n-1}, such that f_i contends with f_{i+1}, $1 \leq i < n$. By this definition, two remote flows may belong to the same group even though they do not contend with each other. In a well-connected WMN, all flows in the network may even form a single large contending flow group. In this case, the basic share of each flow will be very small and unable to provide sufficient fairness. For example, in Fig. 1-1, the $2n$ flows belong to a single contending flow group, which includes all flows that are *transitively* related via contentions. According to the formulation in [32], the basic share is $\frac{c}{3n}$, where c is the channel capacity. Note that a two-hop flow consists of two one-hop subflows, consuming $\frac{2c}{3n}$ of local bandwidth. The network is required to ensure the basic share for each flow and then maximize the network throughput, which means the rest of bandwidth will all be assigned to one-hop flows. Consequently, each two-hop flow receives

a final share of $\frac{c}{3n}$, and each one-hop flow receives a final share of $\frac{c}{2} - \frac{2c}{3n}$. When n is large, all two-hop flows are starved.

We study a fundamental problem, how to support weighted bandwidth allocation among all end-to-end flows in a multihop wireless network based on IEEE 802.11 DCF. A more precise but less intuitive definition of the problem is how to adapt the flow rates to achieve the *global maxmin objective* [6]: the rate of any flow in the network cannot be increased without decreasing the rate of another flow which has an equal or smaller normalized rate, where the *normalized rate* is defined as the flow rate divided by the flow weight. Two solutions are proposed to achieve the global maxmin objective.

The first solution is a cross-layer design. A generalized maxmin model is proposed for multihop wireless networks. At the network layer, it allocates network capacity to end-to-end flows for maxmin bandwidth allocation. At the MAC layer, our design achieves the allocated bandwidth shares for the flows through a two-level weighted fair queuing algorithm. The proposed design is able to equalize the end-to-end bandwidth allocation to competing flows that share common bottlenecks, while fully utilizing the network capacity.

The second solution proposed is a fully distributed solution that is compatible with IEEE 802.11 DCF. We transform the global maxmin objective to four local conditions and prove that, if the four local conditions are satisfied in the whole network, then the global maxmin objective must be achieved. We then design a distributed rate adaptation protocol based on the four conditions. Whenever a local condition is tested false at a node, the node informs the sources of certain selected flows to adapt their rates such that the condition can be satisfied. Comparing with [32], which we believe is the most related work, our protocol has a number of advantages. First, it does not modify the backoff scheme of IEEE 802.11. Second, it replaces per-flow queueing with per-destination queueing. Packets from all flows to the same destination is queued together. Third and most important, our protocol achieves far better fairness (or weighted fairness) among end-to-end flows than the basic fair scheme in [32].

1.2 Lifetime Fairness in Sensor Networks

Wireless sensor networks have a wide range of applications in habitat observation, seismic monitoring, battlefield sensing, etc. As another type of multihop wireless network, a sensor network consists of battery-powered sensor nodes that are limited in computation capability, memory space, communication bandwidth, and above all, energy supply.

The network cannot carry out its task after the nodes' energy is exhausted. Hence, maximizing the operational lifetime of a sensor network is a critical problem.

What is exactly the lifetime of a sensor network? Many prior works [7, 11, 27, 33, 38, 46, 48, 54, 58, 59] define the network's lifetime as the time before the first sensor in the network runs out of energy, or before the first loss of coverage [8]. This definition simplifies the problem of maximizing lifetime to a linear programming problem or an NP-hard non-polynomial programming problem if the sink is allowed to move [48]. However, in reality, the operational lifetime of the network is not limited to the smallest lifetime of all nodes. When one sensor dies, the rest of the network can still work, as long as useful data generated by other sensors can still reach the sink. It is not true that, since sensors around the sink forward others' data, they will always exhaust their energy first and prevent the rest of the network from reaching the sink. One can deploy more sensors around the sink, use larger batteries to boost the energy level there, or perform in-network data aggregation.

An appropriate definition for the lifetime of a sensor network should include the lifetimes of all sensors that produce useful data. A sensor's lifetime is the duration from the time when it begins to generate the first data packet to the time when it generates the last packet that is deliverable to the sink. The network's lifetime can be defined as the vector of all sensors' lifetimes sorted in ascending order, which is called the *lifetime vector*. The value of the lifetime vector is determined by the nodes' *packet forwarding policies* that specify how packets are forwarded from the sensors through the network to the sink. More

specifically, for every node, its forwarding policy specifies the proportion of packets that should be forwarded on each outgoing link towards the sink.

Hou et al. [24, 25] define the problem of maximizing a sensor network's lifetime as to find the packet forwarding policies for all nodes that collectively produce the lexicographically largest lifetime vector, called the *maximum lifetime vector*. In less precise terms, it first maximizes the smallest lifetime of all nodes, then maximizes the second smallest lifetime of all nodes, and so on. Hou et al. show that this problem can be modeled as a series of linear programming (LP) problems. After solving the LP problems, the sink uploads the optimal packet forwarding policies to the sensors. Based on its forwarding policy, each sensor forward its packets. Such a solution is however a centralized one. It requires solving $O(|N|)$ LP problems of size $O(|E|)$, where $|N|$ is the number of sensors in the network, $|E|$ is the number of links, and LP has high-order polynomial complexity. The computation overhead can be prohibitively high for large sensor networks that need to be operational soon after deployment. Collecting the complete information about the network and uploading the complete forwarding policies to all nodes require significant amount of transmissions in the network, particularly for nodes around the sink. To avoid these problems, a distributed algorithm that spreads the overhead evenly on all nodes becomes important.

We propose the first distributed solution for the problem of maximizing the lifetime vector of a sensor network. Our strategy is to design a distributed progressive algorithm that works in a series of iterations, each producing a result (in our case, a lifetime vector and its corresponding forwarding policies) that is better than the previous one. The sequence of results approaches to the optimal solution. A distributed progressive algorithm is practically attractive because a result is available at any time and is getting better as more time is spent. We show that when the algorithm stabilizes, its result produces the maximum lifetime vector. We have performed thousands of simulation runs on random networks of various sizes, and compared with Hou's centralized algorithm as well as other

related algorithms. The results demonstrate that our algorithm rapidly converges to the maximum lifetime vector and its overhead is small. For networks of thousands of nodes, it produces near optimal results in 10 to 30 iterations — one iteration requires each node to transmit two small control messages. The algorithm scales well as its overhead increases slowly with respect to network size. When used as a centralized algorithm, it is two to three orders of magnitude faster than Hou's linear programming solution for random networks of thousands of nodes; the performance gap increases for larger networks. We also compare the proposed algorithm with other existing algorithms that maximize the smallest sensor lifetime in the network or perform minimum-power routing. DPA produces much better lifetime vector.

1.3 Maximizing Lifetime Vector and Maximizing Rate Vector in Sensor Networks

There is another rate control problem with lifetime requirement in wireless sensor networks. A *rate vector* is the vector of all sensors' local rates sorted in ascending order. For a given node lifetime requirement T for all nodes, the problem of *maximizing rate vector* is to find the lexicographically largest rate vector. Hou et al. prove in [25] that there exists an underlying duality relationship between the problem of maximizing the lifetime vector of a sensor network and the problem of maximizing the rate vector in a sensor network. The duality relationship is summarized in Table 1-1, in which, g_i is the local data rate of node i and t_i is the lifetime of node i.

1.4 Related Work

1.4.1 Flow Rate Fairness

The maxmin solutions on wired networks [9, 18, 31] not only require per-flow queueing but also assume a fixed bandwidth capacity for each link, which makes them not applicable in random-access wireless networks.

It is well known that TCP does not perform well in wireless networks [4, 57]. Much research has been done to improve TCP's performance, and a recent survey can be

found in [15]. Most existing solutions employ heuristic mechanisms for better congestion signaling. However, they are not designed for solving the problem of *provable weighted bandwidth allocation* as this work does.

Utility-based solutions on wired networks [21, 30, 35, 39] also require each link to have a fixed capacity. Efforts have been made to adapt utility-based solutions in wireless networks by considering only single-hop flows [19, 52], eliminating contention among neighboring nodes by using separate CDMA/FDMA channels for wireless links [56], modeling resources as maximal contention cliques instead of wireless links [55], relying on cross-layer design to integrate end-to-end rate adaptation with MAC-layer packet scheduling [12], assuming all wireless links share the same channel [2], or assuming a fixed bandwidth capacity for each wireless node [43]. Assigning one separate channel for each contending wireless link [56] requires a large number channels for a dense network, causing low capacity for each channel. This approach does not work well with widely-deployed IEEE 802.11b/g that has only three non-overlapping channels. The maximal clique approach [55] requires that each clique's *effective* capacity is known, but it is not clear how to accurately measure such capacity, which is a complex function of nearby contention and environmental noise. The cross-layer approach [12] requires the nodes to dynamically establish globally coordinated (or locally approximated [34]) time-slotted transmission schedules at the MAC layer, which does not fit well with IEEE 802.11's random access model. More importantly, the utility function that approximates maxmin fairness contains an exponent approaching to infinity [40], which makes the system hard to stabilize. In summary, existing utility-based approaches do not provide a maxmin solution for IEEE 802.11 DCF.

There are other works that are not utility-based. Most of them are designed to achieve MAC-layer fairness [28, 36, 37] or maxmin fairness [26, 51] among one-hop flows. While some study multihop flows, each has its limitation. Basic end-to-end fairness in wireless ad-hoc networks is achieved in [32]. However, the basic fair share guaranteed

for each flow is highly conservative; it can be far below the maxmin rate. End-to-end maxmin is investigated in [47], which assumes a separate CDMA/FDMA channel for each contending wireless link. The temporal fairness in multi-rate wireless networks is studied in [20], which however does not provide an algorithm that computes the temporally-fair rates. A distributed algorithm that achieves aggregate fairness in sensor networks is proposed in [14], assuming that all flows are destined to the same base station. To the best of our knowledge, no distributed algorithm has been proposed to provide weighted maxmin bandwidth allocation in a multihop wireless network based on IEEE 802.11 DCF.

IEEE 802.11e [1] has been under development to support QoS, primarily for WLAN. Its EDCA provides prioritized channel access only for four access categories (background, best effort, video, and voice). It does not provide fine-level control for weighted bandwidth allocation among end-to-end flows.

1.4.2 Lifetime Fairness in Sensor Networks

Many researchers design energy-efficient routing algorithms to maximize the network lifetime. Many prior works [3, 7, 11, 27, 33, 38, 46, 58, 59] define the network's lifetime as the time before the first sensor in the network runs out of energy, or before the first loss of coverage [8].

Hou et al. show in [24, 25] that the problem of maximizing the lifetime vector of a sensor network can be modeled as a series of centralized linear programming problems. Hou et al. also prove in [25] that there exists an underlying duality relationship between the problem of maximizing the lifetime vector of a sensor network and the problem of maximizing the rate vector of a sensor network with a global node lifetime requirement.

Some researchers also design energy-efficient routing algorithms to achieve the goal of minimizing energy consumption [22, 45, 49, 50, 53]. The typical approach [22, 45] is to use a shortest path algorithm in which the edge cost is the power consumed to transmit a packet along this edge. Though effectively reducing the energy consumption

rate, this approach can cause unbalanced consumption distribution. The nodes on the minimum-energy path are quickly drained of energy, causing network partition.

The rest of this study is organized as follows. Chapter 2 and Chapter 3 propose two solutions to achieve the global end-to-end flow rate maxmin objective in multihop wireless networks: a cross-layer solution and a fully distributed solution. Chapter 4 proposes a distributed progressive algorithm for maximizing lifetime vector in wireless sensor networks. Chapter 5 concludes our study.

Maximizing rate vector	Maximizing lifetime vector
g_i (optimization variable)	$g_i = R$ (constant)
$t_i = T$ (constant)	t_i (optimization variable)
Total data volume at node i: $g_i \cdot T = t_i \cdot R$	

Table 1-1. Duality relationship between the two problems proved by Hou et al. in [25]

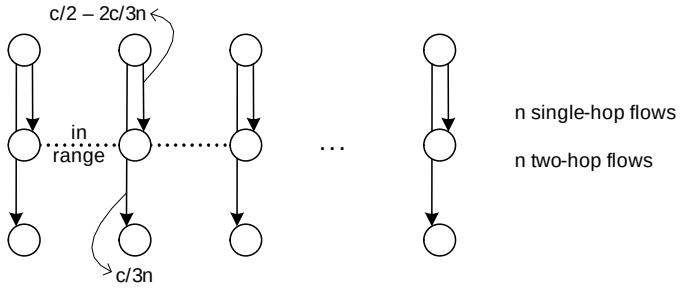


Figure 1-1. Two-hop flows are starved.

CHAPTER 2
CROSS-LAYER DESIGN FOR ACHIEVING END-TO-END MAXMIN

In this chapter, a cross-layer design is proposed for achieving end-to-end maxmin in wireless mesh networks (WMNs). A generalized maxmin model is first proposed for multihop wireless networks. At the network layer, it allocates network capacity to end-to-end flows for maxmin bandwidth allocation. At the MAC layer, our design achieves the allocated bandwidth shares for the flows through a two-level weighted fair queuing algorithm. The proposed design is able to equalize the end-to-end bandwidth allocation to competing flows that share common bottlenecks, while fully utilizing the network capacity.

This chapter is organized as follows. Section 2.1 describes the network model and our objective. Section 2.2 presents a generalized maxmin model, based on which we design the maxmin bandwidth allocation algorithms for WMNs. Section 2.3 presents the two-level weighted fair queuing scheduling algorithm. Section 2.4 evaluates the performance of our solution. Section 2.5 summarizes the chapter.

2.1 Network Model and maxmin Model

2.1.1 Network Model

We adopt infrastructure (backbone) WMNs as our network model. In an infrastructure WMN, mesh routers interconnect through wireless links to form a communication backbone. Clients are connected to mesh routers via wired or wireless means. Clients connected to different routers communicate with one another through multi-hop wireless paths. To simplify the discussion, we assume that two different channels are used for wireless communication *between mesh routers* and wireless communication *between a router and a client*. Therefore, router-router communication does not interfere with router-client communication. We focus on router-router communication. We assume the existence of the IEEE 802.11 DCF MAC protocol. Two wireless links (between routers) *contend* if they cannot transmit simultaneously.

A WMN connects to the Internet through one or multiple *gateway mesh routers*. All communication traffic from clients of one router to clients of another router constitutes an *internal flow*. All communication traffic from clients of one router to the Internet or in the reverse direction constitutes an *external flow*. Internal flows are common in WMNs deployed for campus communication. External flows are common in WMNs deployed in residential areas for Internet access. In our abstract model, we consider the beginning router of a flow as the data source of the flow and the ending router as the destination. We assume the existence of a routing protocol that establishes a routing path for each flow.

We study end-to-end flows, which are referred to simply as *flows*. A flow consists of one or more single-hop flows, which are called *subflows*. Two subflows *contend* if they are carried by the same link or two contending links. Two flows *contend* if any of their subflows contend.

2.1.2 Maxmin Model

Each end-to-end flow is assigned a *nominal flow weight*. The network is expected to allocate bandwidth to the flows in proportion to their nominal weights whenever possible. The nominal flow weights can be decided based on administrative policies (e.g., higher weights for more important flows), commercial policies (e.g., higher weights for customers who pay more), or incentive policies (e.g., higher weights for flows whose source routers contribute more in carrying others' traffic).

Each flow is entitled to a fair share of network bandwidth in proportion to its weight. It is well known that maintaining fairness and maximizing network throughput are contradictive goals [37]. Stricter fairness can be achieved often at the expense of lower network throughput. Some previous studies focused more on throughput optimization under certain basic, relaxed fairness criteria [32, 36]. We put more focus on fairness. Specifically, we want to achieve the classical maxmin fairness among end-to-end flows in

WMNs. The maxmin fairness requires the network to first maximize the smallest flow rate, then maximize the second-smallest flow rate, and so forth.

The classic maxmin model *for wired networks* is described as follows. Given a set Q of resources (i.e., links), a capacity b_q for each resource $q \in Q$ (i.e., bandwidth), a set F of flows, a nominal weight w_f for each flow $f \in F$, and a routing path p_f for each flow f, the problem is to assign a rate r_f for each flow f such that

1. $\forall q \in Q,\ \sum_{f \in F,\ q \in p_f} r_f \leq b_q,$ and
2. for any flow $f \in F$, its rate r_f cannot be increased without decreasing the rate $r_{f'}$ of another flow f', for which $r_{f'}/w_{f'} \leq r_f/w_f$.

The set of rates $R = \{r_f \mid f \in F\}$ that satisfy the above conditions are called the *maxmin rates*.

The above model assumes that each resource has a fixed capacity and that a resource can appear in a flow's routing path at most once. In order to apply this model to WMNs, we have to identify what the resources are. Wireless links cannot be used as the resources because they do not have individually fixed capacities. Following Huang and Bensaou's work [26] which considers only one-hop flows, we shall use "cliques" from the contention graph as the resources, which will be explained in detail in Section 2.2. However, in order to accommodate the "clique resources" in the context of end-to-end flows, we must generalize the maxmin model first in the following section to allow a resource to appear in a flow's routing path for multiple times.

2.2 A generalized maxmin model

In this section, a generalized maxmin model is introduced. By applying this model, each flow in the network is assigned a maxmin fair share that will be used by the packet scheduling algorithm presented in Section 2.3.

2.2.1 Resources in WMNs

In wired networks, all flows that pass a link between two routers compete for the link bandwidth. The links serve as the resources in the classical maxmin model. In a WMN,

the medium is shared by a group of nearby mesh routers. Not only flows passing the same wireless link but also those passing nearby wireless links compete for the shared channel capacity.

A *wireless-link contention graph* can be employed to describe the spatial contention relationship among contending links. Vertices in a wireless-link contention graph represent wireless links in the corresponding network topology. Two vertices are connected if the corresponding links contend with each other. A wireless link is *idle* if there is no flow passing it. A *simplified* wireless-link contention graph can be constructed from a network topology with all idle links removed. An example of simplified wireless-link contention graph is given by Fig. 2-2 (b).

A *clique* is a complete subgraph with a link between every pair of nodes. A *maximum clique* is a clique that is not contained in another clique. In this chapter, we refer to maximum cliques as cliques henceforth. A clique in a wireless-link contention graph represents a group of mutually contending wireless links in which only one link can be in transmission at any time. The channel bandwidth is shared by all wireless links of a clique. The cliques from the wireless-link contention graph can be used as resources.

Following the routing path of a flow, we can obtain a sequence of cliques that the flow passes. When a flow passes multiple links of a clique, we consider the flow passes the clique multiple times. For a wireless link belonging to multiple cliques, if a flow passes this link, we consider the flow passes those cliques in turn.

2.2.2 Generalized Maxmin Model

In classic maxmin model, a resource can appear in a flow's routing path at most once. Motivated by the above characteristics of WMNs, in this subsection, we generalize the classic maxmin model and then apply the generalized model to WMNs.

In the generalized model, a resource is allowed to appear in a flow's routing path for multiple times in different positions.The number of appearances of resource $q \in Q$ in flow f's path p_f is denoted by n_f^q. We have the following feasibility constraint for a set of flow

rates $R = \{r_f \mid f \in F\}$.

$$\sum_{f \in F,\ q \in p_f} n_f^q \times r_f \leq b_q \tag{2–1}$$

A set of rates that satisfies the above constraint is said to be *feasible*. It is *maxmin fair* if it is feasible and, for each $f \in F$, r_f cannot be increased while maintaining feasibility without decreasing $r_{f'}$ for another flow f', for which $r_{f'}/w_{f'} < r_f/w_f$. Our goal is to find a set of flow rates that is maxmin fair.

An algorithm that calculates the maxmin rates of the flows can be found in [6]. Below we adapt it for the generalized maxmin model. For each resource q, compute the average capacity share available for a unit weight of one appearance of each passing flow, which is $\frac{b_q}{\sum_{f \in F,\ q \in p_f} n_f^q \times w_f}$. Find the global bottleneck resource that has the smallest capacity share. Assign an equal share of the resource's capacity to a unit weight of one appearance of each passing flow f. It can be proved that the equal share is the maxmin normalized rate r_f/w_f of this flow. Remove the bottleneck resource and the flows passing it from the network. When a flow f is removed, the capacity of each resource q on its routing path is reduced by $n_f^q \times r_f$. Repeat the above process until every flow is assigned a rate and removed from the network.

The algorithm described above can be used to compute the maxmin flow rates. The rate of a flow reflects the amount of the bandwidth a flow should received at each node on its routing path. The larger rate a flow has, the more bandwidth a flow should receive at each node on its path. The flow rates calculated by the generalized maxmin model are also called flows' *maxmin fair shares*. By applying the generalized maxmin model, the flows in Fig. 2-2 can be redrawn in Fig. 2-3, where the resources (circles) are cliques. If we assume the bandwidth capacities of all cliques are the same and are normalized to *one* unit, then the maxmin fair shares of f_1, f_2, f_3, f_4 are $1/5, 1/5, 1/3, 1/3$, respectively.

Some previous works, e.g., [26, 32], use channel capacity as clique capacities. If the same way is followed, for some link contention graphs, flow fair shares calculated by the maxmin algorithm are upper bounds of the true maxmin fair shares. One example is the

odd cycles of length at least 5 without chords in link contention graph [19]. To solve this problem, we use the *effective channel capacity* of a clique q as b_q in our model. A node measures the *effective bit rates* of its incident links. The concept of effective bit rate is similar to the one in [5], which incorporates link layer details. A clique's effective channel capacity can be obtained by summing up the effective bit rates of all links of that clique.

In infrastructure WMNs, mesh routers have relatively strong computing capability and stable positions which make the centralized implementation of the algorithm feasible. The implementation can also be distributed. Nodes only work on local link contention graph which is much smaller than the global one. The work of clique decomposition is reduced remarkably. Some distributed maxmin algorithms for wireline networks (e.g., [41]) could be customized to calculate flow maxmin fair shares.

2.3 Packet Scheduling Algorithm

We have discussed how to calculate maxmin flow fair shares. These fair shares replace the nominal flow weights and become the *effective flow weights* used by the packet scheduling algorithm that will be described in this section.

2.3.1 Overview

The basic idea of our scheduling algorithm is to let each subflow receive bandwidth proportionally to its effective weight, which is equal to the effective weight of the flow it belongs to. This idea is similar to the scheduling in wired networks. However, scheduling in multihop wireless networks is more complex. In wired networks, contending subflows are backlogged in the same router. All scheduling work could be done within this router. In a wireless network, contending subflows may reside in different nodes, which could be as far as several hops away. They need to cooperate with each other to guarantee each subflow receive appropriate bandwidth. Our scheduling method includes two components:

- Inter-node scheduling. If we consider all packets in a router form a virtual queue, the weight of this virtual queue equals to the router's effective weight, which is the sum of the effective weights of all backlogged flows in the router. Our method

schedules the transmissions of the packets from virtual queues to guarantee that the bandwidth each virtual queue obtains is proportional to its weight.

- Intra-node scheduling. Inside a router, packets from different flows are queued separately. The intra-node scheduling allocates the bandwidth obtained by the router to the backlogged flows proportionally to their effective weights. Some queuing algorithms proposed for wired networks (e.g., [**?**]) can be adopted to achieve this.

The rest of this section describes the inter-node scheduling algorithm that is based on the 802.11 DCF with RTS-CTS-DATA-ACK handshake.

2.3.2 Inter-node Scheduling

Let B be the set of all backlogged flows in the network, B_i the set of backlogged flows at router i. The effective weight of flow f_i is denoted by w_i' and then the effective weight of router i is $\hat{w}_i = \sum_{j \in B_i} w_j'$. Each router i maintains a counter C_i. When packet P_i^k (the kth packet from router i) becomes the next-to-send packet of router i, it is assigned a tag $T_i = C_i$. Then $C_i = C_i + L_i^k/\hat{w}_i$, where L_i^k is the length of packet P_i^k. In order to achieve the short-term fairness, all counters are reset to zero every ϕ seconds at the same time, assuming clocks of all routers are loosely synchronized.

Fig. 2-4 shows all possible contending packet transmissions within two hops away from router x and router y. In Fig. 2-4, circles represent routers. A line between two routers means they are within the transmission range of each other. An arrow from router x to y means the next-to-send packet of x need to be transmitted to y. Given the example in Fig. 2-4, the transmission from x to y conflicts with all other transmissions indicated by the arrows in Fig. 2-4.

If a node x has a packet to transmit, the *contending node set* of x, denoted by Ω_x, is defined as the group of nodes that are competing for the media access with x. In Fig. 2-4, $\Omega_x = \{i, j, m, y, n, v, w\}$. Let $\Omega_x^+ = \Omega_x \cup \{x\}$. When x has a packet to transmit and its backoff timer becomes zero, it should compare its tag with those of the nodes in Ω_x. Ideally, the packet from x should be transmitted immediately if its tag is the smallest. Otherwise, x's transmission should be withheld until all packets from Ω_x with smaller tags

are transmitted first. To describe how to determine if a transmission should be withheld, we need first define three variables for each router x:

- T_x^s: *sending tag* of x, which is the tag of the next-to-send packet of x. If x does not have any packet to send, T_x^s is set to a very large value MAXTAG.
- T_x^r: *receiving tag* of x, which is the smallest tag of the packets to be received by x from its neighbors.
- T_x^n: $T_x^n = min_{i \in N_x}\{T_i^s, T_i^r\}$, where N_x is x's neighbor set.

When x has a packet to be transmitted to y, if $T_x^s > T_x^n$ or $T_x^s > T_y^n$, x can know it does not have the smallest tag in Ω_x^+ and the transmission should be withheld. In order to obtain most up-to-date T_x^n and T_y^n, RTS, CTS, DATA and ACK packets can piggyback necessary tags. Each router maintains a table to keep track of its neighbors' tags.

However, it is difficult to enforce the above strict conditions for each transmission. The reason is that sender x cannot always have the fresh tags of its neighbors, especially T_y^n from receiver y, which are based on the tags of nodes as far as three hops away from x. Stale tags may cause deadlocks. To avoid potential deadlocks, a heuristic method is used by x to estimate T_y^n. The basic idea is to estimate the increment rate of T_y^n, denoted by r_y^n. For each $i \in N_x$, besides T_i^n, x also records r_i^n it estimates and the time t_i when T_i^n gets updated. When x needs to transmit a packet to y, x uses $\hat{T_y^n}$ instead of T_y^n to check the second condition, where

$$\hat{T_y^n} = T_y^n + r_y^n \times (t - t_y) \tag{2–2}$$

t is the current time. Once T_y^n gets updated and becomes larger, the new r_y^n is computed as:

$$r_y^n = \alpha \times r_y^n + (1 - \alpha) \times \frac{\Delta T_y^n}{t - t_y} \tag{2–3}$$

where α is a parameter to control the influence of T_y^n's new increment rate on r_y^n. If above approach is employed, $\hat{T_y^n}$ will eventually be increased large enough such that the second withholding condition will become false.

We have discussed that ideally router x should withhold its transmission until its sending tag becomes the smallest in Ω_x^+. Actually, we do not have to enforce such strict conditions. x should be allowed to transmit a packet as long as its sending tag is not "very large" compared to the tags of the nodes in Ω_x. Now the two transmission withholding conditions can be formally given as follows. When router x need transmit a packet to y, the transmission should be withheld if:

1. $T_x^s > T_x^n + \beta \times L/\hat{w}_x$, or
2. $T_x^s > \hat{T_y^n} + \beta \times L/\hat{w}_x$

where L is the packet length, β is a parameter greater than zero. β specifies how many packets x is allowed to transmit ahead of its contending nodes. By introducing β, x do not have to wait until it has the smallest sending tag in Ω_x^+, but will withhold its transmission when its sending tag is "much larger" than those of the nodes in Ω_x.

2.4 Performance Evaluation

In this section, the proposed solution that achieves end-to-end maxmin fairness (referred to as MMF) will be evaluated through simulations. The simulation environment settings are described as follows. The channel capacity is 11Mbps. The transmission range of a mesh router is 250 meters. Each data packet is 1024 bytes long. Per-flow queuing is adopted by each router. We assume each source sends data at a constant bit rate (CBR) of 700 packets per second. The length of each simulation session is 200 seconds. The parameters of MMF are set as follows: ϕ is 10 seconds, α is 0.9, and β is 6.

We compare the performance of MMF with (1) 802.11 DCF (abbreviated as 802.11); and (2) the two-phase protocol (abbreviated as 2PP) proposed in [32]. We compare the algorithms from two aspects: *end-to-end flow fairness* and *spatial reuse of spectrum*.

To evaluate the end-to-end fairness, we adopt the maxmin fairness index [6] (denoted by I_{mm}) and the equality fairness index [16] (denoted by I_{eq}).

$$I_{mm} = \frac{min_{f \in F}\{r_f\}}{max_{f \in F}\{r_f\}}, \qquad I_{eq} = \frac{(\sum_{f \in F} r_f)^2}{|F| \sum_{f \in F} (r_f)^2}$$

I_{mm} measures the ratio of the smallest flow rate to the largest flow rate. I_{eq} measures the overall equality among the flow rates; its value approaches to one if the rates of all flows approach toward equality.

To measure the spatial reuse of spectrum, we employ the *effective network throughput* U, which is defined as $\sum_{f \in F} r_f \times l_f$, where l_f is the number of hops on the routing path of flow f. The packets dropped by the intermediate nodes do not count towards the effective network throughput as they do not contribute to end-to-end throughput. The effective network throughput gives us a measurement for network bandwidth utilization and the efficiency of a protocol.

We present simulation results in two network scenarios: a simple network topology shown in Fig. 2-2 and a complex network topology that will be described later. All flows in both scenarios have the equal nominal weights. In the rest of this section, the unit of the flow or network throughput is packets per second (PPS).

The simulation results of the example in Fig. 2-2 are shown by Table 2-1. The length of a flow is the number of its subflows. MMF shows good end-to-end fairness and comparable bandwidth utilization. In 2PP, the objective of the basic fairness model is to maximize the total end-to-end throughput. Thus single hop flow $\langle 9, 8 \rangle$ has much higher rate than other flows.

The complex scenario simulates the traffic in the backbone of a WMN. 27 nodes are placed in a 900×900 region, in which 25 are non-gateway nodes and 2 are gateway nodes. Gateway nodes are evenly placed in the horizontal midline of the region. The region is divided into 25 grids. Each non-gateway node is placed into a grid. The location of a non-gateway node in its grid is randomly chosen. A non-gateway node connects to the Internet through the nearest gateway node. Every non-gateway node has a download flow from its gateway node. 5 non-gateway nodes are randomly picked to have 5 upload flows to their gateway nodes. We also randomly create 5 internal flows among non-gateway nodes. The simulation results are shown in Table 2-2.

For both 802.11 and 2PP, many flows have very low rates. For 802.11, all high rate flows are no longer than 2 hops. The reason is that for a long flow, many packets are dropped before arriving the destination due to buffer overflow caused by inconsistent subflow rates. For 2PP, all high rate flows have only one hop. The reason is that the basic fair share guaranteed for each flow in such large network is very small. Bandwidth is allocated to single hop flows whenever possible to maximize total end-to-end throughput. MMM shows much better fairness than the other two and also achieves good bandwidth utilization.

2.5 Summary

In this chapter, we have studied the problem of end-to-end fairness in WMNs. A generalized maxmin model is presented. This model is applied to WMNs by considering the unique characteristics of wireless networks to provide end-to-end maxmin fairness. A two-level packet scheduling algorithm is proposed to make each flow receive bandwidth at each node on its routing path proportionally to its maxmin fair share calculated by the model. Simulation results have demonstrated the effectiveness of the proposed solution in enhancing end-to-end fairness.

		802.11	2PP		MMF	
flow	length	thro.	effe. weight	thro.	effe. weight	thro.
$\langle 1, 6\rangle$	4	114.44	1.00	115.60	1.00	173.74
$\langle 3, 1\rangle$	2	198.82	2.50	288.65	1.00	173.74
$\langle 7, 10\rangle$	2	272.21	1.00	114.36	1.67	291.49
$\langle 9, 8\rangle$	1	414.32	6.00	682.03	1.67	292.24
effe. network thro.		1814.13	1950.46		1917.66	
I_{mm}		0.276	0.168		0.595	
I_{eq}		0.837	0.627		0.940	

Table 2-1. Simulation results on the topology in Fig. 2-2

	802.11	2PP	MMF
effe. network thro.	1550.15	998.86	1528.45
I_{mm}	0.004	0.026	0.500
I_{eq}	0.136	0.453	0.895

Table 2-2. Simulation results of the complex scenario

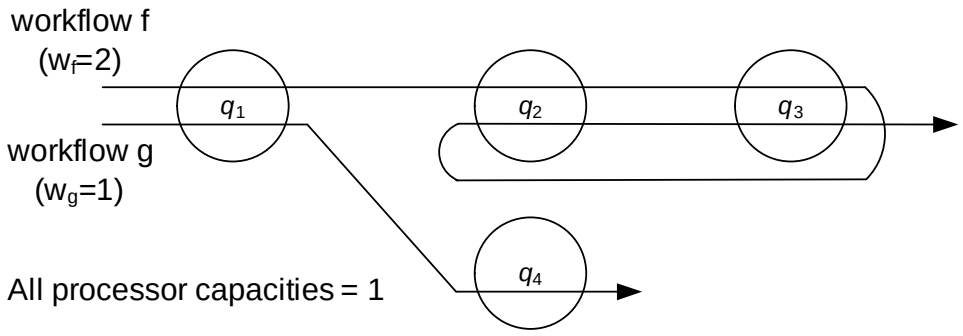


Figure 2-1. A simple example of the generalized maxmin model

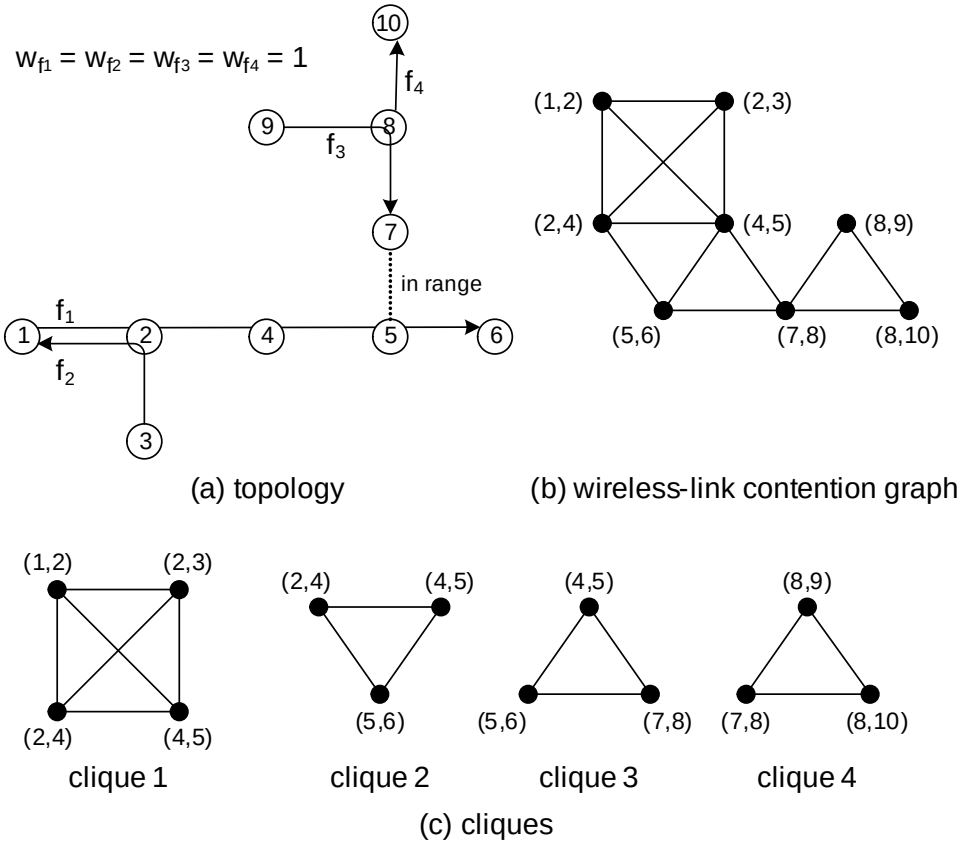


Figure 2-2. An example of wireless-link contention graph and cliques

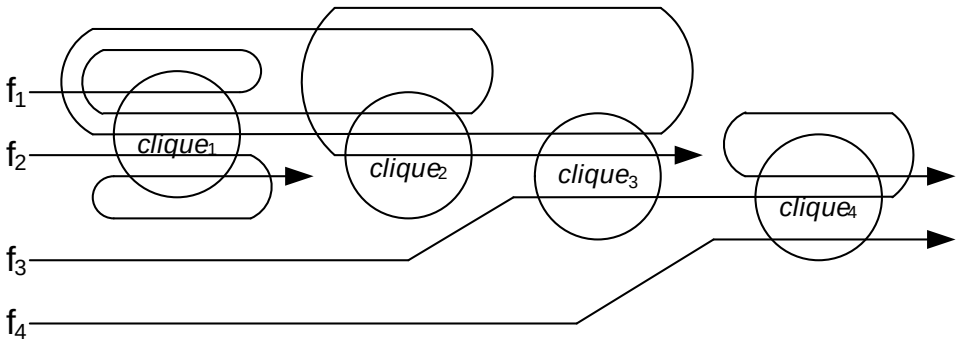


Figure 2-3. Flows described by the generalized maxmin model

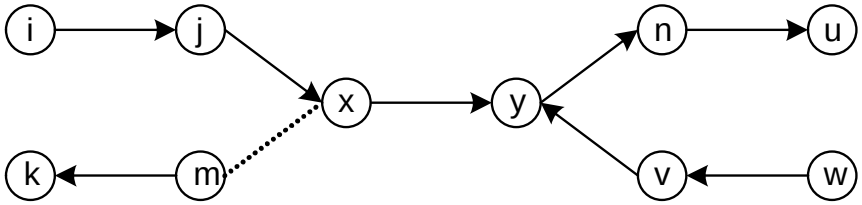


Figure 2-4. Scheduling among contending nodes

CHAPTER 3
FULLY DISTRIBUTED SOLUTION FOR ACHIEVING GLOBAL END-TO-END MAXMIN

In this chapter, we present a fully distributed approach to support weighted bandwidth allocation among all end-to-end flows in a multihop wireless network. Our goal is to enable the network to adapt the flow rates such that global maxmin can be achieved. In order to design a fully distributed solution that is compatible with IEEE 802.11 DCF, we transform the global maxmin objective to four local conditions and prove that, if the four local conditions are satisfied in the whole network, then the global maxmin objective must be achieved. We then design a distributed rate adaptation protocol based on the four conditions. Whenever a local condition is tested false at a node, the node informs the sources of certain selected flows to adapt their rates such that the condition can be satisfied. Comparing with [32], which we believe is the most related work, our protocol has a number of advantages. First, it does not modify the backoff scheme of IEEE 802.11. Second, it replaces per-flow queueing with per-destination queueing. Packets from all flows to the same destination is queued together. Third and most important, our protocol achieves far better fairness (or weighted fairness) among end-to-end flows than the basic fair scheme in [32].

The rest of the chapter is organized as follows. Section 3.1 defines the network model. Section 3.2 classifies wireless links into three categories. Section 3.3 presents the local conditions for global maxmin in wireless networks with a single destination. Section 3.4 presents the local conditions for networks with multiple destinations. Section 3.5 designs a distributed global maxmin protocol based the local conditions. Section 3.6 evaluates the protocol by simulations. Section 3.7 summarizes the chapter.

3.1 Preliminaries

3.1.1 Network Model and Problem Statement

We consider a *static* multihop wireless network (such as wireless mesh network with external power supply for each node) based on IEEE 802.11 DCF with congestion

avoidance enhancement [13]. Mobile ad-hoc networks are beyond the scope of this study. Two nodes are neighbors of each other if there are able to perform RTS/CTS/DATA/ACK exchange. Two nodes that are not neighbors communicate via a multihop wireless path. Time is not slotted. Radio interference is resolved by random backoff. Two wireless links contend if they cannot transmit simultaneously. Based on the most popular MAC protocol, this model excludes the majority of related works [12, 20, 34, 43, 47, 56].

Let F be a set of end-to-end flows in the network. Each flow f has a desirable rate $d(f)$ and a weight $w(f)$. But the flow source will generate new packets at a smaller rate if the network cannot deliver its desirable rate. The actual rate of flow f is denoted as $r(f)$ ($\leq d(f)$). The *normalized rate* of flow f is defined as

$$\mu(f) = r(f)/w(f) \tag{3–1}$$

In this chapter, when we refer to "flow rate" or "normalized rate of a flow", we mean "end-to-end rate". The *global maxmin objective* is defined as follows: The normalized rate $\mu(f)$ of any flow f cannot be increased without decreasing the normalized rate $\mu(f')$ of another flow f', for which $\mu(f') \leq \mu(f)$.

In a more intuitive but less precise description, our goal is to equalize the normalized rates of all flows as much as possible, particularly, raising the smallest ones. Directly competing flows tend to receive bandwidth in proportional to their weights. Achieving global maxmin is a fundamental function of end-to-end traffic engineering in multihop wireless networks. It adds a new entry in the existing tool box (which includes price-based and other solutions) for traffic differentiation among applications. For example, we may establish several service classes in the network and assign larger weights to applications belonging to higher classes. How to enforce a certain weight assignment scheme through service contract or other means is beyond the scope of this study.

We assume there exists a routing protocol that establishes a routing table at each node. The routing table may be implicit under geographic routing [10, 29], or explicitly

established by a distance-vector [42] or link-state routing protocol. Consider a specific destination. A node may receive packets from multiple *upstream neighbors* and forward them to a *downstream neighbor* towards the destination. The links from the upstream neighbors to a node are called *upstream links* of the node, and the link from a node to its downstream neighbor is called the *downstream link*.

3.1.2 Congestion Avoidance and Buffer-Based Backpressure

Suppose packets to different destinations are queued separately. This assumption is necessary to achieve global maxmin, as we will explain in Section 3.4.1. Note that other works [32, 47] require per-flow fair queueing, which is a more stringent requirement. Now consider the packets to a single arbitrary destination. A node buffers packets received from upstream links before forwarding them the downstream link. The buffer space for the queue is limited. To avoid packet drops due to buffer overflow, we adopt the congestion avoidance scheme in [13], which allows a node i to send its downstream neighbor j a packet only when j has enough free buffer space to hold the packet. Suppose the buffer space is slotted with each slot storing one packet. To keep the neighbors updated with j's buffer state, whenever j transmits a packet (RTS/CTS/DATA/ACK), it piggybacks its current buffer state, for example, using one bit to indicate whether there is at least one free buffer slot. When an upstream neighbor i overhears a packet from j, it caches the buffer state of j. If j's buffer is not full, i transmits its packet. If j's buffer is full, i will hold its packet and wait until overhearing new buffer state from j. Note that the residual buffer at node j changes only when j receives or sends a data packet. Whenever this happens, j will send either CTS/ACK or RTS/DATA, immediately informing the neighbors of its new buffer state through piggybacking. No cyclic waiting is possible if routing is acyclic. To handle failed overhearing, i will stop waiting and attempt transmitting if it does not overhear j's buffer state for certain time. Readers are referred to [13] for discussion on other issues.

When there is a bottleneck in the routing path of a flow, the buffer at the bottleneck node will become full, forcing the upstream node to slow down its forwarding rate, which in turn makes the buffer of that node full. Such buffer-based backpressure will propagate all the way to the source of the flow. When the buffer at the source is full, the source has to slow down the flow rate (at which new packets are generated), in order to match the rate it sends our packets. Ultimately, the flow rate is determined by the forwarding rate at the bottleneck. There is no explicit signaling for the above buffer-based backpressure. The only overhead is the buffer-state bit piggybacked in each packet.

3.2 Link Classification

We classify the wireless links into different types based on the buffer state. In the discussion, we consider packets to a single arbitrary destination.

3.2.1 Saturated Buffer

When the combined rate from the upstream links of node j exceeds the rate on the downstream link, if no action is taken, the excess packets will be dropped due to buffer overflow, reducing the effective capacity of the network. With the congestion avoidance scheme [13], when the buffer at j becomes full, it forces the upstream neighbors to slow down to a combined rate that matches the rate on the downstream link.[1] *Whenever j sends out a packet*, it frees some buffer space such that the upstream neighbors can compete for transmission. *Whenever j receives a packet*, its buffer may become full again and the upstream neighbors may have to wait for the next release of buffer at j. A buffer is *saturated* if it continuously switches between full and unfull, which slows down the rates of upstream links as the upstream neighbors have to spent time waiting for buffer release. A buffer is *unsaturated* if it stays unfull (for most of the time).

[1] Slowing down the rate from upstream can even help raising the rate on the downstream link due to less contention.

3.2.2 Three Link Types

Consider an arbitrary link (i, j). If there is sufficient bandwidth to carry all packets received by i over (i, j) and other links downstream after j to the destination, then (i, j) is an unsaturated link. On the other hand, if (i, j) or any link downstream is a bottleneck that cannot carry all packets received by i, then (i, j) is called *a saturated link*. Depending on where the bottleneck is, a saturated link is either bandwidth-saturated or buffer-saturated. Before we give the formal definition, we analyze three different traffic conditions.

Case 1: (i, j) *is the bottleneck.* Both the upstream paths from flow sources to i and the downstream path from j to the destination can deliver more traffic than what (i, j) can do. Even when all available bandwidth is used, being a bottleneck, i still cannot forward all received packets. Consequently, the buffer at i will be saturated, while the downstream path has sufficient bandwidth to keep the buffer at j unsaturated.

Case 2: (i, j) *is not the bottleneck but buffer-based backpressure from a downstream bottleneck propagates through this link.* i forwards more packets to j than the downstream path can deliver. Excess packets will fill the buffer at the bottleneck, causing buffer-based backpressure and eventually saturating the buffer at j and i. Even though link (i, j) has enough bandwidth, the rate on (i, j) will be forced down because i has to wait whenever j's buffer becomes full.

Case 3: (i, j) *is not the bottleneck and no buffer-based backpressure propagates through this link.* Node i is able to forward all packets that it receives. Its buffer will be unsaturated.

Based on the above three cases, we classify wireless links into three types: bandwidth-saturated links, buffer-saturated links, and unsaturated links.

- A link (i, j) is *bandwidth-saturated* if i's buffer is saturated but j's buffer is unsaturated. The fact that j's buffer is unsaturated means the downstream path from j to the destination is able to deliver all packets that i forwards to j. The fact that i's buffer is saturated means that (i, j) does not have sufficient bandwidth to

timely deliver the packets received by i. Therefore, link (i, j) is the bottleneck. The only reason that prevents i from sending more packets to j is because the channel capacity has been fully utilized by (i, j) and its contending links. Hence, the rate on a bandwidth-saturated link cannot be increased without decreasing the rate of a contending link.

- A link (i, j) is *buffer-saturated* if both i's buffer and j's buffers are saturated. The fact that j's buffer is saturated means the downstream path has a bottleneck that cannot timely deliver the packets received by j. The backpressure from that bottleneck causes j's buffer to be saturated, which in turn causes i's buffer to be saturated. The rate on link (i, j) is limited not because the local channel capacity is fully used, but because the downstream path is bottlenecked and i has to spend a fraction of its time waiting for j to release buffer.

- A link (i, j) is *unsaturated* if i's buffer is unsaturated. Both link (i, j) and the downstream path from j to the destination are able to timely deliver all packets received by i. The buffer at j could be either unsaturated or saturated. An unsaturated buffer at j indicates that j is able to timely forward all packets it receives. There is no bottleneck in the downstream path from j to the destination. A saturated buffer at j indicates that j also receives packets from upstream neighbors other than i. Due to limited traffic supply from i, although j's buffer is saturated, i is still able to timely forward its packets and remain unsaturated.

A bottleneck link must be bandwidth-saturated as there is sufficient data to use up all bandwidth available to the link. A non-bottleneck link is either buffer-saturated link or an unsaturated link. The available bandwidth is not fully utilized because of a downstream bottleneck in the former case or shortage of data supply from upstream in the latter case.

3.2.3 Saturated Clique

A set of mutually contending wireless links forms a *contention clique* [26, 32, 55]. A proper clique is a clique that is not contained by a larger clique. In the following, when we refer to a contention clique, we already mean a proper clique. A link may belong to multiple cliques, consisting of nearby contending links. Packet transmissions on the links of a clique must be made serially. Therefore, the combined rate on all links of a clique is bounded by the channel capacity. A clique is *saturated* if the links have utilized all available bandwidth such that increasing the rate on one link will always lead to decreasing the rate on another link in the clique. Because a bandwidth-saturated link uses

up all available bandwidth that it can acquire, it must belong to one or multiple saturated cliques.[2]

3.3 Local Conditions for Global Maxmin: Single-Destination Case

We transform the global maxmin objective to several local conditions to be satisfied. Essentially our goal is to transform a global non-linear optimization problem into a fully distributed optimization problem (represented by the local conditions), which lays down the theoretical foundation for designing a distributed solution in Section 3.5. For now, we assume that all flows go to the same destination. The assumption will be removed in the next section.

3.3.1 Basic Idea

Clearly, letting IEEE 802.11 DCF decide flow rates will not achieve the global maxmin objective. Consider a network with two contending links, one carrying a single flow, f_1, and the other carrying two flows, f_2 and f_3. Suppose the weights of all flows are one. IEEE 802.11 DCF allocates channel capacity equally between the two links. Hence, f_1 can send at twice the rate of other flows. However, for this simple example, the global maxmin objective requires the rates of all three flows to be the same. One approach to meet this goal is to inform the source of f_1 to lower the flow rate by self-imposing an appropriate rate limit. The problem is how to decide which flows should have rate limits in an *arbitrary* network and, after applying rate limits, whether the resulting flow rates

[2] For a bandwidth-saturated link (i, j), node i is constantly backlogged by definition. It constantly attempts to send whenever the channel is idle, and therefore uses up all bandwidth available to it in the statistical sense under the random access framework of IEEE 802.11 DCF. Note that we assume IEEE 802.11 DCF, not a time-slotted MAC protocol with coordinated transmission schedules. The only reason that prevents the rate on (i, j) from being higher is because the rest of the channel capacity is fully occupied by contending links. In other words, if the rate on (i, j) were to increase, when other contending links access the media at randomized times, their probability of finding media occupied by transmission on (i, j) would be proportionally higher. Consequently the rate of one or more contending links would go down.

achieve global maxmin. Our solution is to establish a set of conditions that are testable based on the current network state. We shall prove that, if the conditions are all satisfied, then the global maxmin objective must be met. Moreover, if a condition is tested to be false, it should tell us which flows should increase their rates and which should decrease, so that we can inform the sources of those flows to adjust their rate limits. Finally, in order to make the solution fully distributed, the conditions have to be localized. Namely, they can be tested distributedly.

For the above example, one may argue that, although IEEE 802.11 DCF does not provide fairness, many MAC protocols [26, 28, 36, 37, 51] have been proposed to achieve that. IEEE 802.11e can also provide coarse-level rate control. But the example is a single one with only one-hop flows. These MAC solutions cannot provide *end-to-end* fairness, let alone *provable* weighted maxmin.

3.3.2 Normalized Rate

The data rate on link (i, j) is denoted as $r(i, j)$. The *normalized rate* on (i, j) is defined as the largest normalized rate of any flow that passes (i, j).

$$\mu(i,j) = \max_{f \in F,\ (i,j) \in p(f)} \{\mu(f)\} \tag{3–2}$$

where $p(f)$ is the routing path of flow f. There is an easy way for each link to know its normalized rate. When the source of a flow produces new packets, it lets the packets carry the flow's normalized rate. The nodes of a link inspect the passing packets and take the largest normalized rate carried in the packets as the link's normalized rate.

The set of flows that pass (i, j) consists of all flows passing the upstream links and all flows that begin from i. By the definition of normalized rate, we have the lemma below.

Lemma 1. *The normalized rate of link (i, j) is equal to the largest value among the normalized rates of all upstream links of i and the normalized rates of all flows whose sources are i.*

3.3.3 Local Conditions for Global Maxmin

We transform the global maxmin objective into four localized conditions below.

- *Source Condition:* For every node i with a saturated buffer, if i is the source of a flow, then the normalized rate of the flow is no less than that of any upstream link of i and no less than that of any other flow whose source is i.
- *Buffer-Saturated Condition:* For every buffer-saturated link (i, j), the normalized rate of (i, j) is no less than that of any other upstream link of j and no less than that of any flow whose source is j.
- *Bandwidth-Saturated Condition:* Each bandwidth-saturated link has the largest normalized rate in at least one saturated clique that it belongs to.
- *Rate-Limit Condition:* The rate limit at a flow source should be set the highest without violating the previous three conditions.

Checking the above conditions does not require global state of the entire network. As we will see in Section 3.5 where we design the protocol, the first two conditions can be tested by each node individually and the third condition only requires information exchange among nearby nodes, which can be efficiently done. The fourth condition requires the rate limit at a flow source to be additively increased until a source, buffer-saturated or bandwidth-saturated condition is violated in the network. When this happens, the source will be signaled to tighten its rate limit. For example, if the bandwidth-saturated condition is violated, a link l that has the highest normalized rate in the saturated clique will be asked to reduce its rate in order to give up some bandwidth for the bandwidth-saturated link. Link l will identify the packets carrying the largest normalized rate and inform the sources of those packets to reduce their rates. In response, the sources will self-impose tighter rate limits.

We illustrate the purpose of the four local conditions by a couple of examples. First, examine the simple case in Section 3.3.1, where the network has only two wireless links, (i, t) and (j, t). There are three flows, one from i to t and two from j to t. Assume both i and j have saturated buffer. Satisfying the source condition ensures that the two flows on (j, t) have the same normalized rate. Satisfying the bandwidth-saturated condition

ensures that they also have the same normalized rate as the flow on (i, t) does. Hence, the maxmin objective is achieved. Regardless of what the flows' weights are, equalizing normalized rate means that the flows' rates will be proportional to their weights. One may be puzzled by the contradictive fact that IEEE 802.11 DCF would assign equal bandwidth to (i, t) and (j, t), which means the flows on (j, t) would each have half of the bandwidth for the flow on (i, t). The answer is that, to satisfy the four local conditions, rate limits must be enforced on some flow sources. In this example, a rate limit at j will reduce its flow rate such that the flows on (i, t) can receive more bandwidth. (Detailed operations will be given in Section 3.5.)

Figure 3-1 gives a more sophisticated example. Satisfying the source condition ensures that the normalized rate of flow f_4 is as high as that of any other upstream flow. The buffer-saturated condition requires that flow f_1 has the same normalized rate as f_2, f_3 and f_4. Because f_1's weight is 2, its actual rate should be twice that of f_2, f_3 or f_4. To satisfy this condition, rate limits must be applied at v, w and x to give more bandwidth to u. Satisfying the bandwidth-saturated requirement ensures that the normalized rates of flows (f_1 through f_5) passing the bandwidth-saturated link (i, j) are as large as any contending flows (f_6). This may require a rate limit to be applied at k on f_6. The rate-limit condition makes sure that this rate limit is not set too low.

It is noted that, the rate increment of a flow source may lead to the violation of one or more local conditions but the flow source is not required by other nodes to reduce its rate. In that case, the rate increment at the flow source does not violate the local conditions. An example is given in Fig. 3-2. The three links are in a saturated clique. Link (k, t) is bandwidth-saturated. In Fig. 3-2 (c), the normalized rate of f_1 is lower than those of the other two flows. By rate-limit condition, i increases the normalized rate of f_1 from 1 to 3. Due to the limited bandwidth in the saturated clique, the normalized rate of (k, t) drops to 4, as shown in Fig. 3-2 (d). Bandwidth-saturated condition is violated as (j, t) has larger normalized rate than (k, t). Thus j is required to reduce the

rate of f_2. In this example, although the increment of f_1's rate results in the violation of the bandwidth-saturated condition, only the source of f_2, which is j, violates the local condition. Fig 3-2 (e) shows the final rate allocation of the three flows that satisfies all four local conditions. The rate limit of f_1 or f_2 cannot be further increased without violating the bandwidth-saturated condition.

3.3.4 Correctness Proof

We prove the equivalence between the global maxmin objective and the four local conditions.

The portion of a flow's routing path from the first node whose buffer is saturated to the first bandwidth-saturated link is called the *primary saturated subpath* of the flow. It is easy to see that the primary saturated subpath of a flow consists of a chain of buffer-saturated links and a bandwidth-saturated link at the end. The chain of buffer-saturated links in the primary subpath is the result of buffer-based backpressure originated from the bandwidth-saturated link, which is demonstrated in Figure 3-1 (c), where the bottleneck link (i, j) causes the upstream links buffer-saturated. It is possible that the primary saturated subpath of a flow does not have a buffer-saturated link. For a flow f, given the fact that the buffer at its destination is unsaturated,[3] there must be a node whose buffer is unsaturated on its routing path. If the buffer at the source of f is saturated, f must have a primary saturated subpath.

For a flow f with $r(f) < d(f)$, the first bandwidth-saturated link whose normalized rate is equal to $\mu(f)$ on the routing path of f is called the *primary bandwidth-saturated link* of f.

[3] We assume the destination of each flow is capable of timely dealing with incoming packets and keeping its buffer unsaturated.

Lemma 2. *For any unlimited flow f with $r(f) < d(f)$ and a saturated buffer at the source, the primary bandwidth-saturated link is the first bandwidth-saturated link on the routing path if the source condition and the buffer-saturated condition are satisfied.*

Proof: First we consider the case where the first link on the routing path of f is a bandwidth-saturated link, which forms the entire primary saturated subpath of f. By the source condition and Lemma 1, the normalized rate of this link must be equal to $\mu(f)$. Then this link is the primary bandwidth-saturated link.

Next we consider the case where the first link on the routing path is not a bandwidth-saturated link. Let the primary saturated subpath be $i_1 \rightarrow i_2 \rightarrow ... \rightarrow i_k \rightarrow i_{k+1}$, where i_1 is the source of the flow, $(i_l, i_{l+1}), 1 \leq l < k$, are all buffer-saturated and (i_k, i_{k+1}) is bandwidth-saturated. The buffers at nodes $i_l, 1 \leq l \leq k$, are all saturated.

We prove by induction that the normalized rates of links $(i_l, i_{l+1}), 1 \leq l \leq k$, are all equal to $\mu(f)$. By the source condition and Lemma 1, $\mu(i_1, i_2) = \mu(f)$. Suppose $\mu(i_{l-1}, i_l) = \mu(f), 1 < l \leq k$. Because (i_{l-1}, i_l) is buffer-saturated, by the buffer-saturated condition and Lemma 1, we must have $\mu(i_l, i_{l+1}) = \mu(i_{l-1}, i_l) = \mu(f)$, which completes the induction proof. Therefore, (i_k, i_{k+1}) is the primary bandwidth-saturated link of flow f. □

For a flow with an unsaturated buffer at the source, the portion of its routing path from the source to the first node whose buffer is saturated, or to the destination if there is no such node, is called the *primary unsaturated subpath* of the flow. Its routing path begins with the primary unsaturated subpath followed by the primary saturated subpath, as shown in Fig. 3-3. It is possible that a flow does not have a primary saturated subpath. In that case, the primary unsaturated subpath of the flow forms the entire routing path.

Lemma 3. *When the four localized conditions are satisfied in the network, for any flow f with a rate limit and an unsaturated buffer at the source, if $r(f)$ is increased by a small amount, the violation incurred by f must occur at a link on its routing path and the normalized rate of the link must be equal to $\mu(f)$ before the rate increment on f.*

Proof: By the rate-limit condition, when $r(f)$ is increased, one or more local conditions are violated and the source of f is required to reduce the rate of f. Suppose the amount of f's rate increment is very small and the buffer at the source of f is still unsaturated. The violation will not appear at the source because the source condition and the buffer-saturated condition are not applicable at a node with an unsaturated buffer. Therefore, the violation must appear on at least one link on the routing path of f. Because any small amount of rate increment on f can introduce a violation and the rate reduction request will always be sent to the source of f, the normalized rate of the link where the violation appears must be equal to $\mu(f)$ before the rate increment on f. □

Lemma 4. *For any unlimited flow f with $r(f) < d(f)$ and an unsaturated buffer at the source, if the four localized conditions are satisfied, there must be a link on the routing path of f that has the largest normalized rate which is equal to $\mu(f)$ in at least one saturated clique it belongs to.*

Proof: There must be a rate limit at the source of f because $r(f) < d(f)$ and the buffer at the source of f is unsaturated. Suppose the rate of f is increased by a small amount. By rate-limit condition, the rate increment on f will violate at least one of the first three local conditions.

The violation can occur on the primary unsaturated subpath. In this case, the bandwidth-saturated condition may be violated by f. Let (i, j) be the first link on the primary unsaturated subpath on which the bandwidth-saturated condition is violated. A bandwidth-saturated link (i', j') will not have the largest normalized rate in any saturated clique. Because any small amount of rate increment on f can introduce the violation, both (i, j) and (i', j') must have the largest normalized rate in a saturated clique before f's rate is increased. By Lemma 3, the normalized rate of (i, j) is equal to $\mu(f)$.

For a flow f with $r(f) < d(f)$, the first unsaturated link on the routing path that has the largest normalized rate which is equal to $\mu(f)$ in at least one saturated clique

it belongs to is called the *primary unsaturated link* of f. In the above case, (i, j) is the primary unsaturated link of f.

If f has a primary saturated subpath, the source condition or the buffer-saturated condition may also be violated at the last link of the primary unsaturated subpath of f (denoted by (i_0, i_1)). Let the primary saturated subpath be $i_1 \rightarrow i_2 \rightarrow ... \rightarrow i_k \rightarrow i_{k+1}$, where $(i_l, i_{l+1}), 1 \leq l < k$, are all buffer-saturated and (i_k, i_{k+1}) is bandwidth-saturated. We will prove below that (i_k, i_{k+1}) is the primary bandwidth-saturated link of f.

By Lemma 3, $\mu(i_0, i_1) = \mu(f)$. By the source condition and the buffer-saturated condition, among all upstream links and local flows of i_1, the buffer-saturated upstream links and the local flows of i_1 have the largest normalized rates. Because any small amount of rate increment on f will violate the source condition or the buffer-saturated condition, (i_0, i_1) must also have the largest normalized rate before f's rate is increased. By Lemma 1, we have $\mu(i_1, i_2) = \mu(f)$. By the buffer-saturated condition and Lemma 1, all other links on the primary saturated subpath have the same normalized rate as (i_1, i_2). Then $\mu(i_k, i_{k+1}) = \mu(f)$. Therefore, (i_k, i_{k+1}) is the primary bandwidth-saturated link of f.

If the violation happens on a link after (i_0, i_1) on the routing path, the normalized rate of (i_1, i_2) must be equal to $\mu(f)$ before $r(f)$ is increased. This can be proved by contradiction. Assume $\mu(i_1, i_2) > \mu(f)$ before $r(f)$ is increased. By Lemma 1, all links on the routing path after (i_1, i_2) also have normalized rates larger than $\mu(f)$. Among all links on the routing path from (i_1, i_2), there is a link on which the violation of flow f occurs. The normalized rate of that link is larger than $\mu(f)$ before $r(f)$ is increased, which contradicts with Lemma 3. By the buffer-saturated condition and Lemma 1, all other links on the primary saturated subpath have the same normalized rate as (i_1, i_2). Then $\mu(i_k, i_{k+1}) = \mu(f)$. Therefore, (i_k, i_{k+1}) is the primary bandwidth-saturated link of f.

By the bandwidth-saturated condition, (i_k, i_{k+1}) has the largest normalized rate in at least one saturated clique. □

The first primary bandwidth-saturated link or primary unsaturated link of f is uniformly called the *primary link* of f. By summarizing Lemma 2 and Lemma 4, we can get the lemma below.

Lemma 5. *For any unlimited flow f with $r(f) < d(f)$, if the four localized conditions are satisfied, f must have a primary link. The primary link of f has the largest normalized rate which is equal to $\mu(f)$ in at least one saturated clique it belongs to.*

Theorem 1. *When all flows have a common destination, the global maxmin objective is achieved if the four local conditions are satisfied.*

Proof: Suppose the local requirements are achieved. For an arbitrary flow f with $r(f) < d(f)$, we need to prove that, in order to increase the normalized rate $\mu(f)$, we have to decrease the normalized rate $\mu(f')$ of another flow f', for which $\mu(f') \leq \mu(f)$.

We prove it by contradiction. Assume to the contrary that there exists such a flow f that $\mu(f)$ can be increased without decreasing $\mu(f')$ for all flows f' with $\mu(f') \leq \mu(f)$.

By Lemma 5, flow f has a primary link (i, j) and $\mu(i, j) = \mu(f)$. It means that the normalized rates of all other flows passing (i, j) are not greater than $\mu(f)$. When we increase $\mu(f)$ by increasing the rate of f, based on the assumption, the normalized rates of all other flows passing (i, j) will not be decreased, which means that (i, j)'s rate will go up. By Lemma 5, (i, j) has the largest normalized rate in a saturated clique. When (i, j)'s rate goes up, the rate of another link (i', j') in the saturated clique will have to go down. Among all flows passing (i', j'), at least one flow f' has to decrease its rate (and thus $\mu(f')$). Since (i, j) has the largest normalized rate in the clique, we have $\mu(f') \leq \mu(i', j') \leq \mu(i, j) = \mu(f)$, which contradicts with the previous assumption. □

3.4 Local Conditions for Global Maxmin: Multiple-Destinations Case

Removing the assumption of a single destination, we establish local conditions that are equivalent to the global maxmin objective in a general multihop wireless network.

3.4.1 Per-Destination Packet Queueing

We argue that, when the flows passing a node are destined for different destinations, the node should allocate a separate queue for packets to each destination. Consider the network with two flows in Figure 3-4 (a). First, we show that one queue per node will unnecessarily reduce the rate of f_2 in Figure 3-4 (b), where (z, t) is a bandwidth-saturated link, causing buffer-based backpressure to saturate the buffers at j, i, x and y. Suppose the rate of f_1 is 1 due to the bottleneck (z, t). Because the source nodes, x and y, compete fairly for transmission to i whenever i's buffer is not full,[4] f_2 will have the same rate as f_1, even though there is no bottleneck on its routing path. With one queue at each intermediate node, f_2 is penalized because packets from f_1 saturate the shared queues along the path. To solve this problem, a node must be allowed to use multiple queues.

A node is said to *serve* a destination if it is on the routing path of a flow with that destination. A node should maintain a separate queue for each served destination, not for each passing flow. It should be noted that, in a mesh network, many flows may destine for the same destination, i.e., the gateway to the Internet. In Figure 3-4 (c), when i and j keep separate queues for destinations t and v, f_2 will be able to send at its desirable rate of 5.

Separate queues achieve "isolation" between packets for different destinations, which allows us to model the physical wireless network as a set of overlapping virtual networks, each for one destination. Figure 3-4 (d) shows that f_1 and f_2 are delivered in two virtual networks with separate packet queues but sharing the same channel.

3.4.2 Virtual Nodes, Virtual Links, and Virtual Networks

We model each physical node i as a set of *virtual nodes* i_t, one for each served destination t. A virtual node i_t carries one queue, storing all packets received by i for

[4] They also spend the same amount of time waiting for i's buffer becoming unfull, which wastes channel capacity.

destination t. All virtual nodes for the same destination t form a *virtual network*; there exists a *virtual link* (i_t, j_t) if j is i's next hop towards t. (i_t, j_t) is called the *downstream link* of i_t and an *upstream link* of j_t. All virtual networks together are called the *grand virtual network*, which can be viewed as a "decomposed" model of the original wireless network. A wireless link (i, j) is modeled as the aggregate of virtual links (i_t, j_t) from all virtual networks. These virtual links (i_t, j_t) mutually contend because the physical node i can only transmit a packet from one of its queues at each time. An example is given in Figure 3-4 (d), where the wireless network is modeled as two virtual networks, and (i, j) as two virtual links.

Each virtual network carries a subset of flows, which is disjoint from the subsets carried by other virtual networks. Buffer-based backpressure (Section 3.1.2) is performed independently within each virtual network. The normalized rate of a virtual link is defined as the largest normalized rate of any flow passing the link. Within a virtual network, we classify virtual links as bandwidth-saturated, buffer-saturated, or unsaturated in the same way as we did in Section 3.2.2. Other concepts can also be trivially extended to virtual networks.

3.4.3 Localized Requirements for Global Maxmin

Below we modify the local conditions in Section 3.3.3 to suit for a wireless network whose flows have different destinations.

- *Source Condition:* In the virtual network for destination t, for every node i_t with a saturated buffer, if i_t is the source of a flow, then the normalized rate of the flow is no less than that of any upstream link of i_t and no less than that of any other flow whose source is i_t.
- *Buffer-Saturated Condition:* In the virtual network for destination t, for every buffer-saturated virtual link (i_t, j_t), the normalized rate of (i_t, j_t) is no less than that of any other upstream link of j_t and no less than that of any flow whose source is j_t.
- *Bandwidth-Saturated Condition:* Each bandwidth-saturated virtual link has the largest normalized rate in at least one saturated clique that it belongs to.

- *Rate-Limit Condition:* The rate limit at a flow source should be set the highest without violating the previous three conditions.

Lemma 1 - Lemma 5 can be easily extended to virtual networks.

Lemma 6. *The normalized rate of virtual link (i_t, j_t) is equal to the largest value among the normalized rates of all upstream virtual links of i_t and the normalized rates of all flows whose sources are i_t.*

Lemma 7. *For any unlimited flow f with $r(f) < d(f)$ and a saturated buffer at the source, the primary bandwidth-saturated virtual link is the first bandwidth-saturated virtual link on the routing path if the source condition and the buffer-saturated condition are satisfied.*

Lemma 8. *When the four localized conditions are satisfied in the grand virtual network, for any flow f with a rate limit and an unsaturated buffer at the source, if $r(f)$ is increased by a small amount, the violation incurred by f must happen at a virtual link on its routing path and the normalized rate of the virtual link must be equal to $\mu(f)$ before f's rate is increased.*

Lemma 9. *For any unlimited flow f with $r(f) < d(f)$ and an unsaturated buffer at the source, if the four localized conditions are satisfied, there must be a virtual link on the routing path of f that has the largest normalized rate which is equal to $\mu(f)$ in at least one saturated clique it belongs to.*

Lemma 10. *For any unlimited flow f with $r(f) < d(f)$, if the four localized conditions are satisfied, f must have a primary virtual link. The primary virtual link of f has the largest normalized rate which is equal to $\mu(f)$ in at least one saturated clique it belongs to.*

Theorem 2. *The global maxmin objective is achieved if the four local conditions are satisfied in the grand virtual network.*

Proof: Suppose the local requirements are achieved. For an arbitrary flow f with $r(f) < d(f)$, we need to prove that, in order to increase the normalized rate $\mu(f)$, we have to decrease the normalized rate $\mu(f')$ of another flow f', for which $\mu(f') \leq \mu(f)$.

We prove it by contradiction. Assume to the contrary that there exists such a flow f that $\mu(f)$ can be increased without decreasing $\mu(f')$ for all flows f' with $\mu(f') \leq \mu(f)$.

By Lemma 10, flow f has a primary virtual link (i_t, j_t) and $\mu(i_t, j_t) = \mu(f)$. It means that the normalized rates of all other flows passing (i_t, j_t) are not greater than $\mu(f)$. When we increase $\mu(f)$ by increasing the rate of f, based on the assumption, the normalized rates of all other flows passing (i_t, j_t) will not be decreased, which means that (i_t, j_t)'s rate will go up. By Lemma 10, (i_t, j_t) has the largest normalized rate in a saturated clique. When (i_t, j_t)'s rate goes up, the rate of another virtual link (i'_v, j'_v) in the saturated clique will have to go down. Among all flows passing (i'_v, j'_v), at least one flow f' has to decrease its rate (and thus $\mu(f')$). Since (i_t, j_t) has the largest normalized rate in the clique, we have $\mu(f') \leq \mu(i'_v, j'_v) \leq \mu(i_t, j_t) = \mu(f)$, which contradicts with the previous assumption. □

3.5 Distributed Global Maxmin Protocol (GMP)

In this section, we design a distributed protocol that adapts the flow rates to satisfy the four local conditions in Section 3.4.3, which is equivalent to meeting the global maxmin objective in wireless networks with multiple destinations.

3.5.1 Overview

Our basic means is to set appropriate rate limits at flow sources such that the local conditions can be satisfied in the network. Assume the system clocks at the nodes are loosely synchronized. The time is divided into alternating measurement/adjustment periods. In each measurement period, all nodes measure the state of its adjacent (virtual) links and exchange information with close-by nodes. In each adjustment period, based on the information measured by itself and close-by nodes, each node checks the first three local conditions. If one or more conditions are false, the node issues rate adjustment

requests for selected flow sources, which adjust their rates accordingly. If a flow source does not receive a rate adjustment request, it will increase its rate limit to meet the fourth condition. After a series of measurement and adjustment periods, the rate limits of all flows are gradually modified to meet the four conditions.

Even after the conditions are satisfied, the network/traffic dynamics may cause them to be violated again. The protocol will continuously change the flow rates to restore the conditions and achieve global maxmin in the current network/traffic environment.

In the protocol description, we refer to a physical node simply as "node", denoted as "i", in contrast to a "virtual node", denoted as "i_t" for destination t. We refer to a link between two physical nodes as "wireless link", denoted as "(i, j)", which may contain multiple "virtual links", denoted as "(i_t, j_t)". We refer to the original network as "wireless network", in contrast to "virtual network" consisting of virtual links. The protocol could have been designed to work entirely on virtual nodes/links, but we optimize it by working on physical nodes and wireless links whenever possible and on virtual nodes/links only when we have to. The reason is that there are a lot more virtual nodes/links than physical ones.

Flow f is a *local flow* at node i if i is the source of f. Flow f is a local flow of virtual node i_t if f is a local flow of i and its destination is t. The *primary flow* of a (virtual) link is the flow that has the largest normalized rate among all flows passing that (virtual) link. When multiple flows have the largest normalized rate, they are all primary flows.

Below we explain the operations performed in the measurement and adjustment periods. Note that the operations by a virtual node i_t are *actually* performed by the physical node i.

3.5.2 Measurement Period

In this period, nodes measure the state of their links. At the end of the period, they exchange the link state.

Step 1: Measurement

All virtual nodes measure their buffer state, based on which they determine the types of their adjacent virtual links. The virtual nodes also measure the normalized rates of their adjacent virtual links. The physical nodes measure the channel occupancies of their adjacent wireless links; we will discuss how to determine saturated cliques based on this information. Details of measurement are given below.

Buffer State: Each virtual node i_t carries one queue for all packets received by i destined for t. A certain amount of buffer is designated for the queue. Over each measurement period, i_t measures the fraction Ω of time in which the buffer stays full. If Ω is above a threshold, i_t sets the buffer state as saturated. We find in our simulations that, if the upstream neighbors supply more packets than i_t can forward, Ω will always stay above 50%, and if the upstream neighbors supply fewer packets than i_t can forward, Ω will be almost zero. Therefore, we set the threshold to 25%.

At the end of a measurement period, for each virtual link (i_t, j_t), the end nodes exchange their buffer state, which can be piggybacked in RTS/CTS/DATA/ACK packets with one extra bit (saturated or not). Based on their buffer state, both i_t and j_t can determine the type of (i_t, j_t), which is buffer-saturated, bandwidth-saturated, or unsaturated.

Link Rate: For each virtual link (i_t, j_t), i_t measures the average data rate $r(i_t, j_t)$ on the link over each measurement period.

Normalized Rate: In the first half of each measurement period, the flows' normalized rates are measured at their sources. In the second half of the period, each flow source selects a number of data packets to piggyback the flow's current normalized rate. From the packets forwarded on a virtual link (i_t, j_t), both i_t and j_t learn the virtual link's normalized rate, which is the largest normalized rate carried in the packets. They also learn the sources of the virtual link's primary flows, which are the sources of the packets that carry the largest normalized rate. Clearly, the normalized rate of a wireless link (i, j) is equal to the largest normalized rate of its virtual links (i_t, j_t).

Channel Occupancy: The channel occupancy of a wireless link (i, j) is defined as the fraction of time in which the channel is occupied by packets forwarded by i to j, including RTS/CTS/DATA/ACK transmissions. Nodes i and j measure their transmissions over each measurement period, and exchange their measurements at the end of the period.

Step 2: Information Dissemination

Every node i has the following information at the end of a measurement period: a) the type of each adjacent virtual link, b) the data rate on each downstream virtual link, c) the normalized rate of each adjacent virtual link, d) the sources of the primary flows, e) the normalized rate of each adjacent wireless link, and f) the channel occupancy of each adjacent wireless link. In order to design an protocol that checks the bandwidth-saturated condition in Section 3.4.3, a node must also know the normalized rates and the channel occupancies of all wireless links that contend with any of its adjacent links.

We refer to the normalized rate and the channel occupancy of a wireless link as the *state of the link*. We must disseminate the state of each wireless link (i, j) to all nodes that have a link contending with (i, j). For IEEE 802.11 DCF, it includes all nodes that are within two hops from either i and j. The dissemination protocol is described as follows. Recall that we only consider static wireless networks. After deployment, we assume each node i discovers the wireless topology in its two-hop neighborhood, and identifies a minimum subset of one-hop neighbors, called i's *dominating set*, whose adjacent links reach all two-hop neighbors. Node i informs the nodes in its dominating set of their membership in the set. At the end of each measurement period, if the state of (i, j) changes from the previous period, both i and j broadcast the new state to their one-hop neighbors. When a node in their dominating sets overhears this information, the node re-broadcasts the information to its neighbors.

The state of a link is very small. Instead of making a separate transmission, such information can be disseminated by piggybacking in RTS/CTS/DATA/ACK packets, which are overheard by all nodes in one-hop neighborhood. In this design, i piggybacks

the state of (i, j) in its normal transmission, and after overhearing the information, a node in i's dominating set does the same thing. To overcome failed overhearing, the same information should be be piggybacked in a number of transmissions. We stress that the piggyback design can be applied to disseminate other information in the rest of the protocol as well.

3.5.3 Adjustment Period

When local conditions are tested false, a node proposes rate adjustments for its local flows and primary flows on the adjacent virtual links. We will discuss how to efficiently deliver rate adjustments to the sources of the primary flows at the end of this section.

In order to stabilize the flow rates quicker, we introduce a system parameter β. The data rates, normalized rates, or channel occupancies of two links (or flows, cliques when applicable) are considered to be "equal" if their difference is below β percentage (e.g., 10%). One is considered to be "smaller" than another if it is smaller by at least β percentage.

The operations performed by the nodes in this period are explained below.

- *Removing Unnecessary Rate Limits*

If a local flow's actual rate is smaller than its rate limit, the node removes the rate limit because it is unnecessary.

- *Testing Source Condition and Buffer-Saturated Condition*

If a virtual node i_t has a saturated buffer, it examines the normalized rates of its upstream virtual links and local flows. Let L_1 be the largest value among them, and S_1 be the smallest among the normalized rates of the local flows and those of buffer-saturated upstream virtual links. To satisfy both source condition and buffer-saturated condition, S_1 should be equal to L_1. Otherwise we have to adapt the rates of local and/or passing flows until S_1 is equal to L_1. More specifically, i_t transmits a rate adjustment request (carrying

L_1 and S_1) to all upstream neighbors. When j_t receives the request, it invokes a procedure Adjust(j_t, i_t, L_1, S_1), which is described as follows.

1. If $\mu(j_t, i_t)$ is equal to L_1, then a rate reduction request is issued for the primary flows on virtual link (j_t, i_t). If $L_1 > 3S_1$, it requests the primary flows to halve their rates; otherwise, it requests the primary flows to reduce their rates by β percentage. (The motivation for the above rate reduction scheme is straightforward. While reducing by β percentage is the norm, an optimization is added — when the gap between L_1 and S_1 is too big, reducing by half helps to close the gap quickly. The number 3 is artificially set.)

2. If (j_t, i_t) is a buffer-saturated link and $\mu(j_t, i_t)$ is equal to S_1, then a rate increase request is issued for the primary flows on virtual link (j_t, i_t). If $L_1 > 3S_1$, it requests the primary flows to double their rates; otherwise, it requests the primary flows to increase their rates by β percentage.

Similarly, a rate adjustment request may be issued for a local flow f for destination t. If $\mu(f) = L_1$, i_t issues a rate reduction request for f. If $\mu(f) = S_1$ and f has a rate limit, i_t issues a rate increase request for f.

- *Testing Bandwidth-Saturated Condition*

We have assumed that, after deployment, each node i discovers the wireless topology in its two-hop neighborhood. From the topology, it pre-computes the set of cliques it belongs to. Because there is a one-to-one correspondence between cliques in the original wireless network and cliques in the grand virtual network, we are able to perform most clique-related operations based on wireless links instead of their constituent virtual links (whose number is much larger). Each clique has a system-wide unique identifier, consisting of the smallest identifier of the nodes in the clique and a sequence number. A clique's identifier is assigned by the node with the smallest identifier and disseminated to other nodes in the clique via its dominating set.

At the beginning of each adjustment period, i computes the channel occupancy of each clique, which is equal to the sum of the channel occupancies of the wireless links in the clique. For a wireless link (i, j) that has at least one bandwidth-saturated virtual link, we do the following: First, among its bandwidth-saturated virtual links, we identify

the one (i_t, j_t) with the smallest normalized rate. Among all cliques that (i, j) belongs to, we treat those that have the largest channel occupancy as being saturated. Second, we check whether (i_t, j_t) satisfies the bandwidth-saturated condition. If $\mu(i_t, j_t)$ is not the largest normalized rate in any of its saturated cliques, we must increase $\mu(i_t, j_t)$ by issuing rate adjustment requests. Let L_2 be the largest normalized rate on wireless links in all saturated cliques that (i, j) belongs to. Node i disseminates L_2, $\mu(i_t, j_t)$, and the identifiers of saturated cliques via its dominating set to all nodes in two-hop neighborhood. When a node k receives this information, if a wireless link (k, m) belongs to one of those saturated cliques, k calls Adjust$(k_v, m_v, L_2, \mu(i_t, j_t))$ for each of its virtual links (k_v, m_v), with "buffer-saturated link" in the second step of the routine replaced by "bandwidth-saturated link".

- *Rate Adjustment at Sources*

  At the end of an adjustment period, the source of each flow sends a control packet that travels along the routing path to collect the rate adjustment requests for the flow. It only carries one request. If there is no rate reduction request, it keeps the rate increase request with the smallest increase. If there is a rate reduction request, it discards all rate increase requests. If there are multiple rate reduction requests for the flow, it keeps the one with the largest rate reduction. When the destination receives the control packet, it sends the packet back to the source, which will adjust its rate (by changing the rate limit) based on the request carried in the packet.

- *Meeting Rate-Limit Condition*

  If a flow source does not receive any rate adjustment request and it has a rate limit, it will additively increase its rate limit by a small amount to make sure that the flow will send at the highest possible rate.

3.6 Simulation

We perform simulations to evaluate the proposed distributed global maxmin protocol (GMP). The simulation setup is described as follows. IEEE 802.11 DCF is implemented. The channel capacity is 11Mbps. Each node has a transmission range of 250 meters. Each data packet is 1024 bytes long. The desirable rate of any flow is 800 packets per second. The buffer space at a node can hold 300 packets. The length of each simulation session is 400 seconds. Each measurement or adjustment period is 4 seconds long. β is set to be 10%.

3.6.1 Effectiveness of GMP

The network topology used in the first simulation is shown in Fig. 3-5. First, we assign all flows the same weight 1, so that a flow's normalized rate is the same as the flow rate. Wireless links $(1, 2)$, $(3, 4)$ and $(4, 5)$ mutually contend with each other and form clique 1. Links $(0, 1)$ and $(1, 2)$ form the smaller clique 0. Based on the maxmin model, flows f_2, f_3 and f_4 should have the same normalized rate, and they have equal access to the channel capacity of clique 1. Because the rate of f_2 is limited by clique 1, flow f_1 is able to send at a higher rate, fully utilizing the bandwidth not used by f_2 in clique 0. The simulation results shown in Table 3-1 are consistent with the above analysis. In the simulation, after the flow rates are stabilized, $(0, 1)$ and $(1, 2)$ are bandwidth-saturated links, while $(3, 4)$ and $(4, 5)$ are unsaturated links. The bandwidth-saturated condition ensures that, in the saturated clique 1, the normalized rate of f_2 is no less than those of f_3 and f_4. The rate-limit condition ensures that f_1 will send at the highest-possible rate as long as it does not drive the rate of f_2 too low that violates the bandwidth-saturated condition of f_2 in clique 1.

Next we test weighted maxmin on the same network topology by assigning different weights to flows. The simulation results are given in Table 3-2. The rates of the three flows in clique 1 are approximately proportional to their pre-assigned weights. Flow

f_1 has a higher rate than flow f_2 even though its weight is smaller. That is because it opportunistically utilizes all remaining bandwidth in clique 0 that cannot be used by f_2.

3.6.2 Performance Comparison

In this subsection, we compare the performance of GMP with IEEE 802.11 DCF (abbreviated as 802.11) and the two-phase protocol (abbreviated as 2PP) proposed in [32]. These three protocols use different buffer management strategies to accommodate their packet queuing algorithms. In 802.11, all flows passing a node share the same buffer space. When a packet arrives at a node whose buffer is full, it will overwrite the packet at the tail of the queue. In 2PP, each flow is allocated a separated queue that can hold 10 packets. In GMP, all flows to the same destination share a common queue that can hold 10 packets.

Since 2PP is designed to provide fairness (instead of weighted bandwidth allocation), we compare the protocols from two aspects: *end-to-end flow fairness* (when all flows have equal weights) and *spatial reuse of spectrum*.

As in Section 2.4, we adopt the maxmin fairness index I_{mm} and the equality fairness index I_{eq} to evaluate the end-to-end fairness and effective network throughput U to measure the spatial reuse of spectrum.

First we simulate the scenario in Fig. 3-6. The simulation results are shown in Table 3-3. GMP is much fairer than 2PP, which is in turn much fairer than 802.11. Due to the hidden terminal problem under 802.11, a severe unfairness in media access exists between link $(0, 1)$ and $(2, 3)$ [26]. Node 0 has much less chance to grab the channel when it has packets to be transmitted to node 1. This explains why the flow from node 0 to node 3, which passes $(0, 1)$, has the lowest rate under 802.11. The effective network throughputs of 2PP and GMP are comparable, and they are higher than that of 802.11, which drops more packets due to buffer overflow.

The design of 2PP is to ensure a basic fair share of bandwidth for all flows and then favor short flows in allocating the remaining bandwidth. The basic fair share can be very small, and there are cases in which it is outperformed by 802.11. We perform

simulations on the topology in Fig. 3-7, and the results are shown in Table 3-4. With this topology, the basic fair share calculated based on the formula in [32] is small, and the remaining bandwidth is distributed heavily biased towards f_2 and f_8 based on the linear programming approach in the same paper. Under 802.11, the flows in the middle (f_3, f_4, f_5 and f_6) have lower rates than the flows on the sides (f_1, f_2, f_7 and f_8). The reason is that a flow in the middle need compete for bandwidth with more flows than a flow on the side. With GMP, all flows have approximately equal rates regardless of their locations and lengths. The flows in the middle have slightly lower rates for two possible reasons. First, under GMP, two flow rates are considered to be "equal" if their difference is below β, which is 10% in our simulations. Second, the maximum combined rate of the four links in the middle (which form a contention clique) is slightly lower than those of other cliques due to more collisions in the middle clique.

Finally, we perform simulations on a more complex network topology shown in Fig. 3-8. The network consists of 25 nodes that are deployed in a $900 \times 900m^2$ region. We create 25 multihop flows in the network, where the source and the destination of each flow are randomly chosen. The destinations of the flows starting from a node are listed in square brackets after the source node ID. The wireless links are shown as edges in the graph. A solid line means that the link is on the routing path of at least one flow. The total number of flows passing a links is shown in parentheses beside that link. A dotted line represents an unused link. The simulation results are shown in Fig. 3-9 and Table 3-5. In Fig. 3-9, the flow rates that are under 100 pps (packets per second) use the numbers on the left vertical axis; the flow rates above 100 pps use the numbers on the right vertical axis.

Under 802.11, half of all flows have rates under 10 pps. Several flows (e.g. f_7 and f_{13}) are almost starved. Under 2PP, three one-hop flows, f_0 (from node 6 to node 5 in Fig. 3-8), f_3 (from node 6 to node 1), and f_5 (from node 12 to node 7), have very high rates and contribute more than 50% of the total end-to-end throughput. The three flows

whose rates are around 40pps (f_{11}, f_{14} and f_{24}) are also short flows that are only one-hop or two-hops long. GMP achieves far better fairness as shown in Fig. 3-9C.

3.7 Summary

In this chapter, we proposed a distributed protocol to achieve the global maxmin objective based on four local conditions. We introduced several new concepts, including link classification based on buffer state, virtual links, and virtual networks, which are essential for the development of the local conditions. We performed extensive simulations to evaluate the effectiveness of the proposed protocol and demonstrate that it works far better than existing protocols.

flow	$\langle 0,1\rangle$	$\langle 1,2\rangle$	$\langle 3,5\rangle$	$\langle 4,5\rangle$
rate	563.96	196.96	217.57	221.41

Table 3-1. Simulation results on the topology in Fig.3-5

flow	$\langle 0,1\rangle$	$\langle 1,2\rangle$	$\langle 3,5\rangle$	$\langle 4,5\rangle$
weight	1	2	1	3
rate	527.58	225.40	121.90	377.20

Table 3-2. Simulation results of weighted maxmin in Fig.3-5

flow	802.11	2PP	GMP
$\langle 0,3\rangle$	80.63	131.86	164.75
$\langle 1,3\rangle$	220.07	188.76	176.04
$\langle 2,3\rangle$	174.09	240.85	179.21
U	856.11	1013.96	1025.54
I_{mm}	0.366	0.547	0.919
I_{eq}	0.882	0.946	0.999

Table 3-3. Simulation results on the topology in Fig. 3-6

flow	802.11	2PP	GMP
f_1	221.81	43.31	145.46
f_2	221.81	347.81	145.94
f_3	107.29	43.33	134.26
f_4	107.28	86.67	132.38
f_5	106.36	43.39	135.44
f_6	106.36	86.70	133.04
f_7	223.39	43.36	141.69
f_8	223.39	346.96	149.07
U	1976.54	1214.93	1674.13
I_{mm}	0.476	0.125	0.888
I_{eq}	0.890	0.514	0.998

Table 3-4. Simulation results on the topology in Fig.3-7

	802.11	2PP	GMP
U	1665.76	1672.65	2632.74
I_{mm}	0.002	0.017	0.206
I_{eq}	0.327	0.298	0.835

Table 3-5. Simulation results on the topology in Fig. 3-8

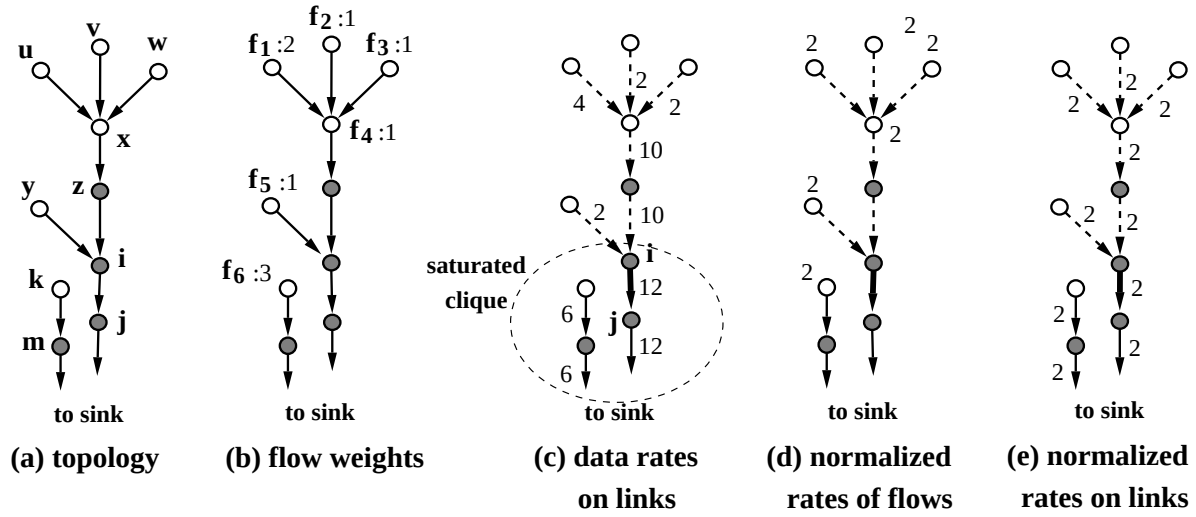


Figure 3-1. White circles represent flow sources. Grey circles represent other nodes. Thick arrows represent bandwidth-saturated links. Thin arrows represent unsaturated links. Thin dashed arrows represent buffer-saturated links. (a) A portion of the network is shown with each arrow pointing from an upstream node to its downstream neighbor. (b) There are six flows, f_1 through f_6, whose weights are shown beside their sources. (c) The actual data rates of the links are shown. (i, j) is a bandwidth-saturated link, which sends buffer-based backpressure upstream, creating buffer-saturated links all the way to the flow sources and slowing the flow rates. (d) The normalized rates of the flows are shown beside the sources. (e) The normalized rates on the links are shown.

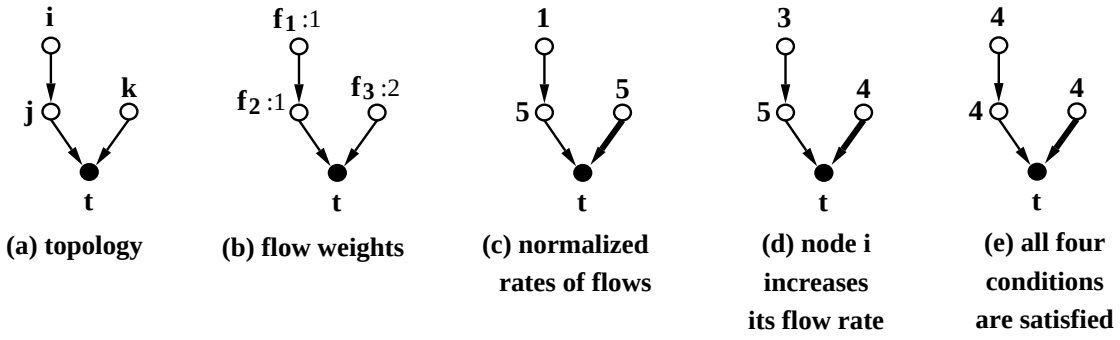


Figure 3-2. An example of rate-limit condition

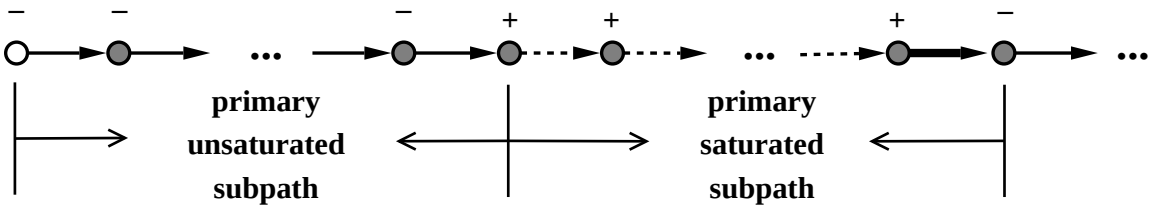


Figure 3-3. White circle represents the flow source. Grey circles represent other nodes. Thick arrows represent bandwidth-saturated links. Thin arrows represent unsaturated links. Thin dashed arrows represent buffer-saturated links. "−" on top of a node indicates an unsaturated buffer at that node. "+" indicates a saturated buffer.

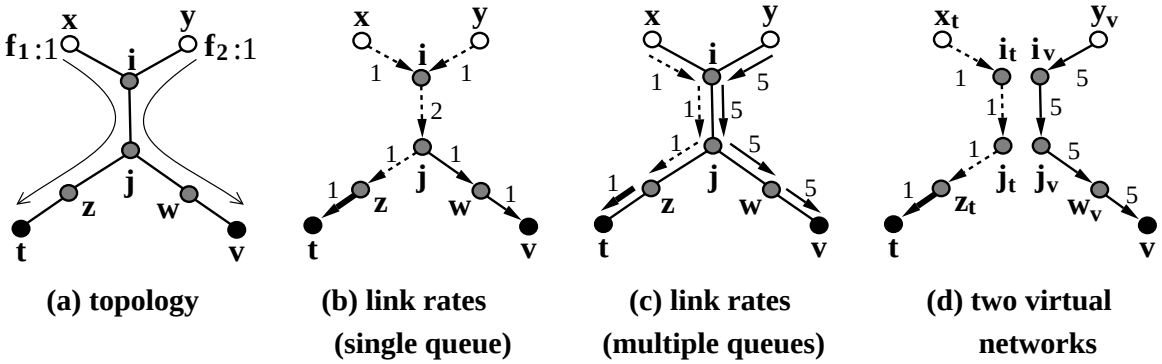


Figure 3-4. White circles represent flow sources. Black circles represent destinations. Thick arrows represent bandwidth-saturated links. Thin arrows represent unsaturated links. Thin dashed arrows represent buffer-saturated links. (a) A portion of the network with two flows whose weights are both one and desirable rates are both 5. (b) Each node has one queue for all destinations. (c) Each node has one queue per served destination. (d) The wireless network is modeled as two virtual networks.

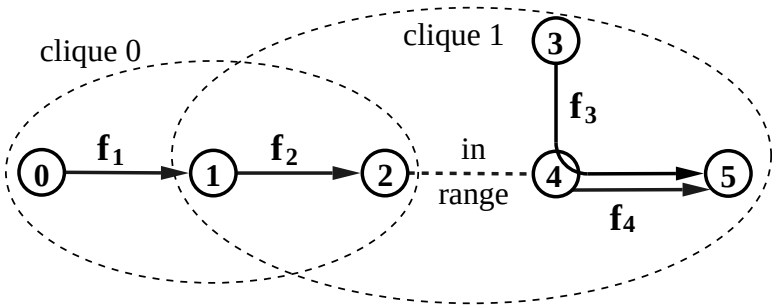


Figure 3-5. Network topology of a simple scenario

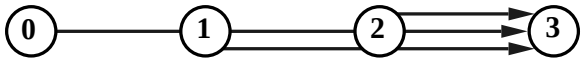


Figure 3-6. A three-links topology

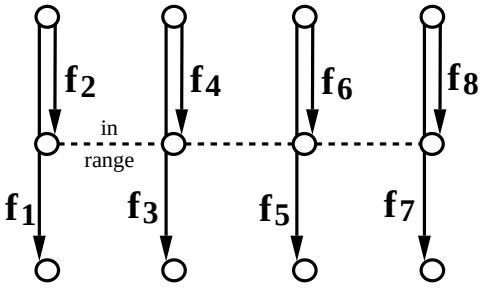


Figure 3-7. Network topology

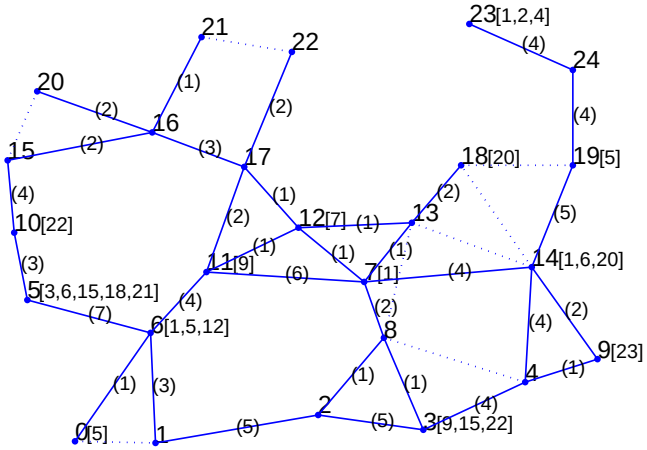
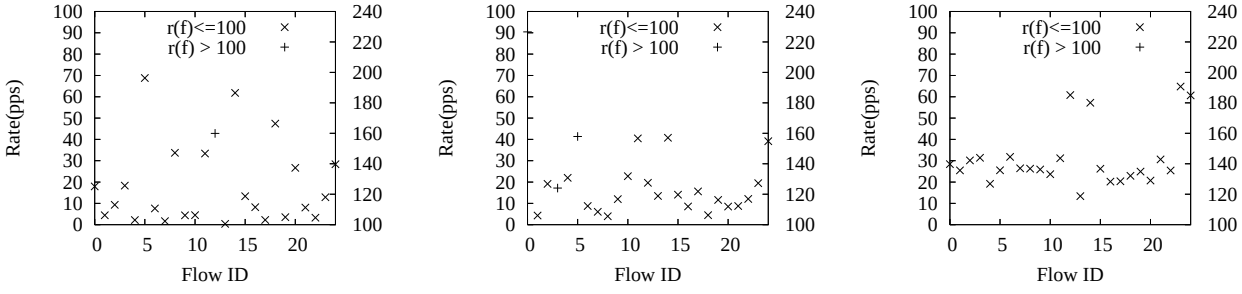


Figure 3-8. Network topology



A 802.11 B 2PP C GMP

Figure 3-9. Rates of the flows on the topology in Fig. 3-8

CHAPTER 4
DISTRIBUTED PROGRESSIVE ALGORITHM FOR MAXIMIZING LIFETIME VECTOR IN WIRELESS SENSOR NETWORKS

Maximizing the operational lifetime of a sensor network is a critical problem in practice. Many prior works define the network's lifetime as the time before the first sensor in the network runs out of energy. However, when one sensor dies, the rest of the network can still work, as long as useful data generated by other sensors can reach the sink. More appropriately, we should maximize the lifetime vector of the network, consisting of the lifetimes of all sensors, sorted in ascending order. For this problem, there exists only a centralized algorithm that solves a series of linear programming problems with high-order complexities. This chapter proposes a fully distributed progressive algorithm which iteratively produces a series of lifetime vectors, each better than the previous one. Instead of giving the optimal result in one shot after lengthy computation, the proposed distributed algorithm has a result at any time, and the more time spent gives the better result. We show that when the algorithm stabilizes, its result produces the maximum lifetime vector. Furthermore, simulations demonstrate that the algorithm is able to converge rapidly towards the maximum lifetime vector with low overhead.

The rest of this chapter is organized as follows. Section 4.1 gives the network model and the problem statement. Section 4.2 lays down the theoretical foundation for our algorithm. Section 4.3 proposes our distributed progressive algorithm for maximizing the lifetime vector. Section 4.4 presents the simulation results. Section 4.5 summarizes the chapter.

4.1 Network Model and Problem Definition

4.1.1 Sensor Network Model

We study the problem of maximizing the lifetime vector of long-term low-rate monitoring sensor networks that collect data from fields for ecosystem study, environmental monitoring, seismic measurement, etc. Operating under battery power, such sensor networks are designed to gather tens of thousands of data points from each selected

location over a period of weeks or months. When *raw information* from each location is of interest, power consumption cannot be substantially reduced by in-network aggregation techniques that are suitable for max/min/avg queries, but not for collecting temporal/spatial data of the whole field (in case that raw data must be made available for future analysis).

Let N be the set of sensor nodes, among which the subset S that generate new data are called *data sources*, which may be the aggregation nodes representing local clusters [24, 25]. Let $g_i, i \in N$, be the *source rate* at which node i generates new data packets. $g_i > 0$ if $i \in S$; $g_i = 0$ if $i \notin S$. We assume that the source rates are set low enough to not cause congestion in the network. The sink may consist of multiple geographically dispersed base stations. Assume the base stations are externally connected to a data collector. It makes no difference which base station a data packet is routed to.

Two nodes are neighbors if they can receive packets from each other (to support DATA/ACK exchange). There may be multiple routing paths from each node to the sink. Let D_i be the set of neighbors that node i use as the next hops to the sink. They are called *downstream neighbors* of node i. $\forall j \in D_i$, (i, j) is called an *outgoing link* of i. Let U_i be the set of *upstream neighbors*, which use i as the next hop on their routing paths to the sink. $\forall k \in U_i$, (k, i) is called an *incoming link* of i. If i is a downstream neighbor of k, then k must be an upstream neighbor of i. Let $E = \{(i, j) \mid \forall i \in N, j \in D_i\}$. We call the graph consisting of all these links as the *routing graph* of the sensor network, which contains all routing paths from data sources to the sink.

4.1.2 Volume Schedule

The *volume* $v(i, j)$ of a link (i, j) is defined as the number of packets transmitted on the link over the lifetime of the sensor network. The *source volume* $v(i)$ of a node i is defined as the number of new data packets generated by i. All link volumes and source volumes together form a *volume schedule*. There are many possible volume schedules, but

not all of them can be actually realized. A volume schedule is *feasible* only if it satisfies the following energy and volume conservation constraints.

Let e_i be the energy available at node i. Let α be the amount of energy that a node spends on receiving a data packet from an upstream neighbor, β_i be the amount of energy that node i spends on producing a new data packet, γ_i be the amount of energy that node i spends on sending a packet. The *energy constraint* is given below.

$$\sum_{k \in U_i} \alpha \times v(k,i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i \times v(i,j) \leq e_i, \quad \forall i \in N \tag{4–1}$$

We say a node i is *exhausted* if

$$\sum_{k \in U_i} \alpha \times v(k,i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i \times v(i,j) = e_i.$$

The *volume conservation constraint* depends on the application model. If the application requires raw data to be delivered from sources to the sink, then the number of packets sent by a node is equal to the number it receives, i.e.,

$$\sum_{j \in D_i} v(i,j) = v(i) + \sum_{k \in U_i} v(k,i), \quad \forall i \in N. \tag{4–2}$$

If it requires periodic measurement of min/max/avg among readings from sources that have not exhausted yet and remain reachable to the sink, then a node will send a packet for each set of packets received from its upstream neighbors or generated locally. The constraint becomes

$$\sum_{j \in D_i} v(i,j) = \max\{\max_{k \in U_i}\{v(k,i)\}, v(i)\} \quad \forall i \in N. \tag{4–3}$$

4.1.3 Maximum Lifetime Vector Problem

The volume schedule specifies how many packets are forwarded through each node, each link and each path, and thus it determines the lifetime vector. For an arbitrary

feasible volume schedule, we can calculate the *lifetime* of each data source $s \in S$ as follows:

$$t_s = v(s)/g_s. \tag{4–4}$$

The *lifetime vector* of the sensor network is defined as $(t_s, s \in S)$ sorted in ascending order.

Each feasible volume schedule produces a *feasible lifetime vector*. All feasible lifetime vectors form the lifetime space. One lifetime vector T is *greater* than another T' if T is lexicographically larger — for some $x \in [1..|S|]$, T and T' share the common $(x-1)$ smallest elements but the xth smallest element in T is greater than the xth smallest element in T'.

The *maximum lifetime vector problem* is to find a feasible volume schedule that produces the largest (or say, maximum) lifetime vector. Intuitively, its goal is to first maximize the smallest lifetime of all sources, then the second smallest, and so on.

Once we find the volume schedule for the maximum lifetime vector, the nodes must know their packet forwarding policies that will realize the volume schedule. To implement a volume schedule, each node i simply does the following: 1) it generates new packets at its source rate g_i for $v(i)$ packets, and 2) it forwards the received packets to downstream neighbors in weighted round robin, using the volumes on the outgoing links as the weights. Therefore, the packet rates on the outgoing links are proportional to the volumes on the links. This is called the *volume-rate property*.

$$\frac{r(i,j)}{v(i,j)} = \frac{r(i,j')}{v(i,j')}, \quad \forall j, j' \in D_i, v(i,j) \neq 0, v(i,j') \neq 0 \tag{4–5}$$

where $r(i,j)$ is the packet rate on link (i,j).

4.1.4 Routing Graph

The routing graph should be a DAG (directed acyclic graph) to avoid cyclic routing because packets that are routed in a cycle waste energy and may not be able to timely reach the sink. Moreover, any feasible volume schedule based on a routing graph with cycles can be transformed into a feasible volume schedule on the routing graph with cycles

removed, producing an equal or larger lifetime vector. This can be shown by the following procedure, assuming the application model (4–2): Identify a routing loop and find the link (i, j) with the smallest volume $v(i, j)$. Deduct the link volumes along the loop by $v(i, j)$ and remove (i, j). Repeat the above procedure until all loops are removed. The resulting volume schedule still satisfies the volume conservation constraint and the energy constraint. No source volume has been changed, and thus the lifetime vector produced by the new volume schedule on the acyclic routing graph remains the same. Furthermore, since some link volumes have been reduced, which may leave room for increasing some source volumes to produce a larger lifetime vector. The same reasoning can be applied to model (4–3) with added details on how to reduce volumes along a cycle and on other links. But the key point is the same — when removing a cycle, link volumes only need to be reduced.

The acyclic routing graph can be easily constructed when packets are forwarded based on hop counts or the nodes' geographic locations to the sink. For example, D_i may consist of all or a selected subset of neighbors that are closer to the sink (based on the hop count or Euclidean distance to the closest base station), and U_i may consist of all or a selected subset of neighbors that are further away from the sink.

4.2 Necessary and Sufficient Conditions for Maximizing Lifetime Vector

This section establishes the theoretical foundation of our distributed algorithm for maximizing the lifetime vector.

The volume of a (directed) path is defined as the minimum volume of the links on the path. A path in the routing graph is called a *forwarding path* if its volume is greater than zero. Otherwise, it is called a *non-forwarding path*.

Node $s \in S$ is a *feeding source* of node $i \in N$ if there is a forwarding path from s to i. Furthermore, node s is a *restricted* feeding source of node i if there is an exhausted node on every forwarding path from s to i. Node s is an *unrestricted* feeding source of node i if there is no exhausted node on at least one forwarding path from s to i, where the *path*

referred in this definition includes s but excludes i. Node s is a *potential source* of node i if it is not a feeding source of i, but there exists a non-forwarding path from s to i, and the path has no exhausted node.

We will establish the necessary and sufficient conditions for maximizing the lifetime vector in a theorem below. Below we explain a basic technique used in the proof, called *volume shift*. Understanding this technique will also help one to understand the design of the algorithm.

Consider the routing graph in Fig. 4-1. Suppose s and w are two unrestricted feeding sources of node i. Let P_1 and P_2 be two forwarding paths that do not have any exhausted node. We show that *the lifetime of an unrestricted feeding source can be increased at the expense of the lifetime of another*. To do so, we simply decrease the source volume of s, then decrease the volumes on the links of P_1, increase the source volume of w, and finally increase the volumes on the links of P_2, all by the same tiny amount, which should be small enough such that its addition on P_2 does not violate the energy constraint. The above operation is called a *volume shift* from s to w with respect to i. It is easy to see that, after volume shift, the volume schedule remains feasible and the lifetime of s is decreased, the lifetime of w is increased, while the lifetimes of all other sources remain unchanged. It is obvious that, to improve the lifetime vector, we shall always perform a volume shift from a node with a larger lifetime to a node with a smaller lifetime.

Not only can a volume shift be performed between two unrestricted feeding sources, but also it can be performed from a restricted feeding source u to an unrestricted feeding source s, or from an unrestricted feeding source s to a potential source z, but not the other way around — more specifically, i) *a volume shift cannot be performed from an unrestricted feeding source s to a restricted feeding source u* because we cannot add any additional volume to P_3 that has an exhausted node x; ii) *a volume shift cannot be performed from a potential source z to an unrestricted feed source s* because the volume of any path from z to i is zero and thus nothing can be shifted out.

Theorem 3. *A feasible volume schedule produces the maximum lifetime vector if and only if the following conditions are met:*

1There is an exhausted node on every path from a source to the sink.

2All unrestricted feeding sources of a node must have the same lifetime, which should be no less than the lifetimes of the restricted feeding sources of the same node, and no greater than the lifetimes of the potential sources of the same node.

Proof: First, we prove that the conditions are necessary. If a feasible volume schedule does not meet either condition, we show that, by modifying the volume schedule, we can produce a larger lifetime vector. If the first condition is not true on a path P from a source s to the sink, we can improve the lifetime of s by increasing its source volume as well as the volume of P by a tiny amount, which results in a larger lifetime vector. Next consider the second condition.

- If an unrestricted feeding source s has a greater lifetime than another unrestricted feeding source w of a node i, we can perform a volume shift (Fig. 4-1) from s to w such that the lifetime of w is slightly increased (but still below that of s), which results in a larger lifetime vector. Note that the volume shift only changes the lifetimes of two nodes, s and w.
- If an unrestricted feeding source s has a smaller lifetime than a restricted feeding source u, we can perform a volume shift from u to s to increase the lifetime vector.
- If an unrestricted feeding source s has a greater lifetime than a potential source z, we can perform a volume shift from s to z to increase the lifetime vector.

Second, we prove that the conditions are sufficient. The lifetime space, consisting of all feasible lifetime vectors, is convex and compact, which can be seen from the linear (or max) nature of the energy constraint (4–1) and the volume conservation constraint (4–2) or (4–3), as well as the lifetime definition (4–4). Radunovic and Le Boudec showed that, in a convex, compact space, a max-min vector exists, and moreover it is unique and must be lexicographically largest in the space [44]. Hence, we only need to show that a feasible volume schedule that meets the two conditions produces the max-min vector, satisfying the following requirement: The lifetime t_s of one source s cannot be increased

without decreasing lifetime t_w of another source w, for which $t_w \leq t_s$. We show that the above requirement is indeed satisfied based on the following facts. First, due to the first condition, each path from s to the sink has an exhausted node. Second, due to the second condition, the energy of these exhausted nodes is all consumed by feeding sources whose lifetimes are not greater than the lifetime of s. If we increase the lifetime (source volume) of s, at least one of those nodes has to pay, by lowering its lifetime (source volume). □

The theorem gives us some guideline for designing a distributed algorithm that generates a volume schedule to maximize the lifetime vector. Below we give intuitive interpretation.

Based on the first condition, data sources should aggressively set their source volumes to the highest values that their paths to the sink allow.

The lifetime of a source s, which is $v(s)/g_s$, can be interpreted as the average volume assigned to each unit of rate. The second condition requires that each unit of rate received by a node i from an unrestricted feeding source deserves the same amount of volume allocation. In other words, for unrestricted feeding sources, node i should allocate volumes in proportion to their rates (that i receives and forwards). However, each unit of rate from a restricted feeding source (which encounters an exhausted node on its forwarding path) may receive less volume allocation at node i. Moreover, a source should always direct its packets to paths that have highest volume allocation per unit of rate.

4.3 Distributed Progressive Algorithm

After the sink is deployed, before the sources actually generate data packets and deliver them to the sink, a distributed progressive algorithm (DPA) is executed to produce a volume schedule, based on which the data packets will be forwarded.

4.3.1 Rate Schedule, Volume-Bound Distribution, Volume Schedule

DPA iteratively refines a volume schedule, $\{v(i,j),\ \forall (i,j) \in E,\ v(i),\ \forall i \in N\}$. To accomplish this task, we need to introduce a couple of auxiliary concepts. A *rate schedule*, $\{r(i,j),\ \forall (i,j) \in E\}$, is defined by assigning a rate value to each link in the routing graph.

A *volume-bound distribution*, $\{b(i,j),\ \forall (i,j) \in E,\ b(i),\ \forall i \in N\}$, is defined by assigning a volume bound to each link and each node, where *volume bound* $b(i,j)$ specifies the maximum volume that is allowed on link (i,j) and *source volume bound* $b(i)$ specifies the maximum volume that is allowed to be generated from a node i. One of the key operations of DPA is to compute volume bounds.

DPA begins with an initial rate schedule that can be arbitrarily set. From the rate schedule and energy availability at the nodes, it computes a volume-bound distribution based on the second condition in Theorem 3. From the volume-bound distribution, it sets a volume schedule, based on which it will in turn derive a new rate schedule. This completes the first iteration of the algorithm. As shown in Fig. 4-2, in each subsequent iteration, DPA repeats the above computation of a new volume-bound distribution (based on the rate schedule from the previous iteration), then a new volume schedule, and finally a new rate schedule. Each iteration produces a better volume schedule whose lifetime vector is larger than the previous one.

The rate schedule, volume-bound distribution, and volume schedule are stored and computed in a fully-distributed way. Each node only maintains the rates, volume bounds, and volumes of its adjacent links with a space complexity of $O(|D_i| + |U_i|)$. Because each directed link is shared by a pair of upstream-downstream nodes. Some properties of the link will be set by the upstream node and then sent to the downstream node, while other properties will be set by the downstream node and then sent to the upstream node. Details are given below.

Node i will set its *outgoing rates*, $r(i,j), j \in D_i$, by distributing the total incoming rate among the outgoing links. It will learn the *incoming rates*, $r(k,i), k \in U_i$, from upstream neighbors k who set those rates. (We want to stress that the link rates here are auxiliary variables used to facilitate the computation of volumes. They have nothing to do with the actual data-packet rates on the links at the time when DPA is executed. In

fact, DPA can be executed at the beginning of the deployment before any data packets are transmitted.)

Node i will set its *outgoing volumes* $v(i,j)$ by distributing the total incoming volume among the outgoing links. It will learn the *incoming volumes* $v(k,i)$ from upstream neighbors k who set those volumes.

Node i will set its *incoming volume bounds* $b(k,i)$ by distributing its forwarding capacity among the incoming links. It will learn the *outgoing volume bounds* $b(i,j)$ from downstream neighbors j who set those bounds.

In the rest of the section, we will describe the details of DPA, which consists of *Initialization phase* and *iterative phase* with each iteration having two steps. The first step computes volume bounds based on link rates. The second step determines link volumes from volume bounds and then computes new links rates, which sets the stage for the next iteration.

4.3.2 Initialization Phase

This phase arbitrarily sets up a rate schedule. The distributed computation for initializing link rates is described as follows: The sink broadcasts an INIT packet backward in the routing graph. When a node k receives INIT, it forwards the packet to its upstream neighbors. If k has no upstream neighbor (i.e., k is a leaf in the routing graph), it distributes its source rate evenly among its outgoing links, i.e., $r(k,i) \leftarrow \frac{g_k}{|D_k|}, \forall i \in D_k$, where "$\leftarrow$" is the assignment operator. Node k then sends those outgoing rates to its downstream neighbors in a RATE packet. After a node i learns $r(k,i)$ in RATE packets from all upstream neighbors k, it first computes its outgoing rates as $r(i,j) \leftarrow \frac{\sum_{k \in U_i} r(k,i) + g_i}{|D_i|}, \forall j \in D_i$, and then sends those rates to downstream neighbors j in a RATE packet. The initialization phase terminates when the sink receives RATEs from all neighbors. Intuitively, this phase begins with a wave of INITs traveling backward in the routing graph to all nodes, and the INIT packets are turned around at leaf nodes to

form a reverse wave of RATEs that assign the initial rates of all links subject to the flow conservation constraint.

In total, at most $|N|$ INIT packets and $|N|$ RATE packets are transmitted. Each node i sends one INIT of size $O(1)$ and one RATE packet of size $O(|D_i|)$. The initialization phase completes within the maximum round trip time between the sink and any source in the network.

4.3.3 Iterative Phase — Step 1: From Rates to Volume Bounds

The first step of each iteration is to set volume bounds based on link rates. Each node i must appropriately set its incoming volume bounds, $b(k,i), \forall k \in U_i$, and the source volume bound, $b(i)$, such that it does not receive more packets than it is able to forward. The volume bounds are subject to two *volume-capacity constraints*.

First, a node i should not receive and forward more packets than the downstream neighbors can handle. If the application model is characterized by (4–2), then the combined incoming volume bound (set by i) should not exceed the combined outgoing volume bound (set by downstream neighbors).

$$\sum_{k \in U_i} b(k,i) + b(i) \leq \sum_{j \in D_i} b(i,j) \tag{4–6}$$

where $b(i,j)$ is learned by i from j. If the application model is characterized by (4–3), then the constraint becomes

$$\max\{\max_{k \in U_i}\{b(k,i)\}, b(i)\} \leq \sum_{j \in D_i} b(i,j) \tag{4–7}$$

Second, node i should not receive and forward more packets than its energy allows.

$$\sum_{k \in U_i} \alpha \times b(k,i) + \beta_i \times b(i) + \sum_{j \in D_i} \gamma_i \times b'(i,j) \leq e_i \tag{4–8}$$

where

$$b'(i,j) = \begin{cases} (\sum_{k\in U_i} b(k,i) + b(i)) \frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')}, & \text{for application model (4–2);} \\ \max\{\max_{k\in U_i}\{b(k,i)\}, b(i)\} \frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')}, & \text{for application model (4–3).} \end{cases}$$

Because of (4–6)-(4–7), $b'(i,j) \leq b(i,j)$, and therefore the above constraint is more relax than one that replaces $b'(i,j)$ with $b(i,j)$.

As we have explained in the previous section, the second condition of Theorem 3 requires that volume allocation should be made in proportion to the incoming rates (which must be adjusted for restricted feeding sources, as will be discussed shortly in Step 2). Hence, we have the following *rate-bound property*.

$$\begin{gathered} \frac{b(k,i)}{r(k,i)} = \frac{b(k',i)}{r(k',i)} = \frac{b(i)}{g_i}, \\ \forall k,k' \in U_i \cup \{i\}, r(k,i) \neq 0, r(k',i) \neq 0, g_i \neq 0 \end{gathered} \tag{4–9}$$

If $r(k,i) = 0$, then $b(k,i) = 0$. If $g_i = 0$, then $b(i) = 0$.

The distributed computation of Step 1 is described as follows: The sink begins the process of setting volume bounds after the rate initialization phase terminates (at the time when the sink receives RATEs from all upstream neighbors), or after Step 2 completes (at the time when the sink receives VOL_RATE packets from all upstream neighbors — to be described in Section 4.3.4). The sink sets its incoming volume bounds to be infinite and sends a BOUND packets to upstream neighbors, carrying the volume bounds of its incoming links. After a node i receives BOUNDs from all downstream neighbors $j \in D_i$ and learns all outgoing volume bounds $b(i,j)$, it sets the incoming volume bounds, $b(k,i), k \in U_i$, and its source volume bound $v(i)$ as large as possible, based on (4–9) subject to the constraints of (4–6)-(4–7) and (4–8). Node i then sends its incoming volume bounds to the upstream neighbors in a BOUND packet.

In total, $|N|$ BOUND packets are transmitted. Each node i only transmits one packet of size $O(|U_i|)$.

4.3.4 Iterative Phase — Step 2: From Volume Bounds to Volumes and Rates

Next we discuss how to set the volumes of all links based on the volume bounds from Step 1. Each node i should set the outgoing volumes, $v(i,j), \forall j \in D_i$, and its source volume $v(i)$, which are subject to the following *bound constraint*.

$$v(i) \leq b(i), \quad v(i,j) \leq b(i,j), \quad \forall j \in D_i, \forall i \in N \tag{4–10}$$

The first condition of Theorem 3 requires us to set the source volume as high as possible. Hence, we assign

$$v(i) \leftarrow b(i) \tag{4–11}$$

In addition to (4–10), outgoing link volumes are also subject to the volume conservation constraint in (4–2) or (4–3). A node cannot send more packets than it receives. If it does not receive enough incoming volumes, its outgoing volumes may have to be set lower than what the volume bounds allow. If the volume conservation constraint is (4–2), to satisfy this constraint, node i assigns its outgoing volumes as follows.

$$v(i,j) \leftarrow (\sum_{k \in U_i} v(k,i) + v(i)) \frac{b(i,j)}{\sum_{j' \in D_i} b(i,j')}, \quad \forall j \in D_i \tag{4–12}$$

where $v(k,i)$ is set by upstream neighbor k and learned by i from k. If the volume conservation constraint is (4–3), node i assigns the outgoing volumes to be

$$v(i,j) \leftarrow \max\{\max_{k \in U_i}\{v(k,i)\}, v(i)\} \frac{b(i,j)}{\sum_{j' \in D_i} b(i,j')}, \ \forall j \in D_i \tag{4–13}$$

It can be shown by induction that the above assignment satisfies the bound constraint in (4–10).

First, we prove by induction that using (4–12) will satisfy the bound constraint (4–10). Consider the base case with $U_i = \emptyset$. By (4–12), (4–6) and the fact that $U_i = \emptyset$, we

have

$$\begin{aligned} v(i,j) &= v(i)\frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')} \\ &\le \sum_{j'\in D_i} b(i,j')\frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')} \\ &= b(i,j) \end{aligned}$$

Next we make the inductive assumption that $v(k,i) \le b(k,i), \forall k \in U_i$, and prove the case when $U_i \neq \emptyset$. (This is a valid inductive assumption for a DAG routing graph, which has no loop for circular reasoning.) Together with (4–6) and (4–11), we have

$$\begin{aligned} v(i,j) &= (\sum_{k\in U_i} v(k,i) + v(i))\frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')} \\ &\le (\sum_{k\in U_i} b(k,i) + b(i))\frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')} \\ &\le \sum_{j'\in D_i} b(i,j')\frac{b(i,j)}{\sum_{j'\in D_i} b(i,j')} \\ &= b(i,j) \end{aligned}$$

The induction proof for the case of (4–13) is similar.

This result, together with (4–8), ensures that the assigned volumes satisfy the energy constraint required in (4–1) — to see this, one has to use the fact that $v(i,j) \le b'(i,j)$ due to (4–12)-(4–13) and (4–10), where $b'(i,j)$ is defined in (4–8). Consequently, the resulting volume schedule is feasible.

After we set the link volumes, we assign new link rates below based on the rate-volume property in (4–5), setting the stage for the next iteration. For application model (4–2),

$$r(i,j) \leftarrow (\sum_{k\in U_i} r(k,i) + g_i)\frac{v(i,j)}{\sum_{j'\in D_i} v(i,j')}, \quad \forall j \in D_i \tag{4–14}$$

For application model (4–3),

$$r(i,j) \leftarrow \max\{\max_{k\in U_i} r(k,i), g_i\}\frac{v(i,j)}{\sum_{j'\in D_i} v(i,j')}, \ \forall j \in D_i \tag{4–15}$$

We have one additional issue that must be handled. As shown in Fig. 4-3, the volume bound assigned by i on link (w, i) for an unrestricted feeding source s will be fully utilized. However, the *volume bound* assigned by i on link (k, i) for a restricted source u may not be fully utilized due to an upstream bottleneck x that may set a tighter bound on the source volume of u. In this case, the *volume* $v(k, i)$, which is set by k and constrained by the limited upstream energy at x, is smaller than the volume bound $b(k, i)$. When this happens, we shall reduce $b(k, i)$ to match $v(k, i)$, and allow $b(w, i)$ to be larger, which will in turn allow s to have a larger source volume and thus a larger lifetime. Since volume bounds are set at Step 1 based on link rates, we can achieve the reduction of $b(k, i)$ by artificially reducing the rate $r(k, i)$.

More specifically, after the link rates are calculated based on (4–14)-(4–15), they may be reduced by multiplying a *reduction factor* $f(i)$ ($\in (0, 1]$), which has an initial value of 1 and is updated at each iteration as follows. Suppose node i is not a direct neighbor of the sink. If i is exhausted, i.e., $\sum_{k \in U_i} \alpha \times v(k, i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i v(i, j) = e_i$, or it was exhausted in one of the previous iterations, then it updates $f(i)$:

$$\begin{aligned} B(i) &\leftarrow \sum_{j \in D_i} b(i, j)/f(i), \\ V(i) &\leftarrow \sum_{j \in D_i} v(i, j) \times \frac{e_i}{\sum_{k \in U_i} \alpha \times v(k, i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i v(i, j)}, \\ f(i) &= \frac{V(i)}{B(i)} \end{aligned} \tag{4–16}$$

where $B(i)$ and $V(i)$ are the would-be volume bound and volume on all outgoing links, respectively, if the rate reduction had not been preformed to reduce the outgoing volume bound in previous iterations. Clearly, the value of $f(i)$ will stabilize at an exhausted node i only when the volume $\sum_{j \in D_i} v(i, j)$ matches the bound $\sum_{j \in D_i} b(i, j)$. After updating $f(i)$, node i reduces the outgoing rates as follows.

$$r(i, j) \leftarrow r(i, j) \times f(i), \quad \forall j \in D_i \tag{4–17}$$

For the example in Fig. 4-3, both i and x will perform the above operation. When x does so, its rate reduction will propagate downstream, causing the reduction of $r(k,i)$, which in turn causes the reduction of $b(k,i)$ and the increase of $b(w,i)$.

The distributed computation of Step 2 for setting volumes/rates is a natural continuation of Step 1. After a node with no upstream neighbor receives BOUND (defined Step 1) from all downstream neighbors, it is able to assign its source volume by (4–11) and outgoing volumes by (4–12)-(4–13). It then updates the link rates by (4–14)-(4–15), (4–16), and (4–17). After that, it sends the outgoing volumes/rates to the downstream neighbors by a VOL_RATE packet. After a node i receives VOL_RATE packets from all upstream neighbors k and learns $v(k,i)$, it is able to assign its source volume by (4–11), the outgoing volumes by (4–12)-(4–13), and the new outgoing rates by (4–14)-(4–15), (4–16), and (4–17). It sends the outgoing volumes/rates to downstream neighbors in VOL_RATE. When the sink receives VOL_RATE from all upstream neighbors, it knows that Step 2 is completed.

Step 2 transmits $|N|$ packets. Each node i sends only one packet of size $O(|D_i|)$. Each iteration, including Step 1 and Step 2, completes within the maximum round trip time between the sink and any source in the network.

4.3.5 Property

DPA carries out three computations to set volume bounds, volumes, and rates, respectively. We show that all three computations lead to better lifetime vectors.

First, consider the computation of volume bounds. The total forwarding capability of a node, which is determined by (4–6)-(4–7) and (4–8), is distributed as volume bounds based on the rate-bound property in (4–9), which essentially performs volume shift from feeding sources with larger volume per unit of rate (i.e., larger lifetime) to those with smaller volume per unit of rate. Such volume shift increases the lifetime vector. The only problem is that a volume bound may not be fully turned into volume if there is an upstream exhausted node which sets a tighter volume bound. This problem is solved by

rate reduction, which contiguously updates a reduction factor by (4–16) until the volume matches the bound.

Second, the volume assignments in (4–10) and (4–12)-(4–13) are aggressive in the sense that they try to fully utilize all volume bounds, by setting the source volumes as high as possible and by forwarding all incoming volumes at each node.

Third, the rate reduction in (4–14)-(4–15), (4–16) and (4–17) artificially decreases the link rates if the volume bounds are not fully turned into the volumes. In subsequent iterations, due to (4–9), decreased rates lead to decreased volume bounds on those links, allowing other links that can fully utilize their bounds to have higher volume bounds.

In summary, the volume bound computation performs volume shift from large-lifetime sources to small-lifetime sources; the volume computation and the rate reduction technique ensure that the volume bounds are fully utilized. Together, they improve the lifetime vector as DPA executes through its iterations. As the lifetime vector moves increasingly closer to its maximum value, the room for improvement becomes smaller and smaller. Our simulations will show that DPA converges rapidly.

Theorem 4. *When DPA stabilizes the link volumes, the resulting volume schedule produces the maximum lifetime vector.*

Proof: Let G be the subgraph consisting of all paths from sources to the first encountered exhausted nodes or to the sink if no exhausted nodes are encountered. Rate reduction has no impact on the link rates inside G. When link volumes are stabilized in G, link rates and volume bounds must also be stabilized because their linear inter-dependency in (4–5), (4–9), (4–12)-(4–13) and (4–14)-(4–15). We prove by induction that

$$v(i,j) = b(i,j), \quad \forall (i,j) \in G. \tag{4–18}$$

Consider the base case with $U_i = \emptyset$. Node i is not exhausted and hence

$$\beta_i \times v(i) + \sum_{j \in D_i} \gamma_i \times v(i,j) < e_i$$

By (4–11) and (4–12)-(4–13), it can be rewritten as

$$\beta_i \times b(i) + \sum_{j \in D_i} \gamma_i \times b'(i,j) < e_i$$

where $b'(i,j)$ is defined in (4–8). Therefore, the real constraint for the value of $b(i)$ is (4–6)-(4–7). Since we should set $b(i)$'s value as large as possible, we have

$$b(i) = \sum_{j \in D_i} b(i,j)$$

By (4–12)-(4–13), we have $v(i,j) = b(i,j)$. Next we make inductive assumption that $v(k,i) = b(k,i), \forall k \in U_i$, and prove the case when $U_i \neq \emptyset$. (This is a valid inductive assumption for a DAG routing graph, which has no loop for circular reasoning.) The proof is similar to the base case, except that $v(k,i)$ and $b(k,i)$ are included in the formulas.

To prove that the resulting volume schedule achieves the maximum lifetime vector, we have to show that the two conditions in Theorem 3 are satisfied.

First, we prove by contradiction that the first condition holds. If not, the sink will be in G. Consider an arbitrary link $(i, sink)$. We have proved earlier that $v(i, sink) = b(i, sink)$, which is not possible because $b(i, sink)$ is infinity.

Second, we prove that the second condition of Theorem 1 holds. Let s and w be two unrestricted feeding sources of node i. Let P_1 be a path from s to i that has no exhausted node. Let P_2 be a path from w to i that has no exhausted node. Both P_1 and P_2 are in G. Hence, for each link on the paths, its volume is equal to its volume bound. By (4–5), (4–9), (4–14)-(4–15), (4–2)-(4–3) and (4–18), the ratio of volume to rate is kept constant over the links of P_1, and equal to $\frac{v(s)}{g_s}$, the lifetime of s. Similarly, the ratio of volume to rate is kept constant over the links of P_2, and equal to $\frac{v(w)}{g_w}$, the lifetime of w. Moreover, these two ratios have to be the same when P_1 and P_2 intersect at i due to (4–9) and (4–18).

By assigning link rates in proportion to link volumes, Eq. (4–14)-(4–15) attempts to equalize the links' volume-to-rate ratios, which means, after each iteration, the rate

is shifted away from downstream links with smaller volume-to-rate ratios to links with larger volume-to-rate ratios. The construction of the initial rate schedule ensures that every routing path is a forwarding path and there is no potential feeding source at the beginning. An unrestricted feeding source may become a potential feeding source by shifting its rate away from a path. When an unrestricted feeding source of i has a larger lifetime than other unrestricted feeding sources of i (due to routing paths that are not through i), the node's downstream link towards i will have smaller volume-to-rate ratio than its other links. Only in this case, due to (4–14)-(4–15), its rate will be contiguously shifted away from feeding i, and eventually turns itself into a potential source of i.

We have proved earlier the first condition of Theorem 3 that there is an exhausted node on every path from a source to the sink. Let G' be the subgraph consisting of all paths from sources to the last encountered exhausted nodes, and C' be the set of those last encountered exhausted nodes, which forms a *cut* of the network that separates the sink from all sources. When link volumes are stabilized in G', link rates and volume bounds must also be stabilized because their linear inter-dependency in (4–5), (4–9), (4–12)-(4–13) and (4–14)-(4–15). We prove by contradiction that

$$v(i,j) = b(i,j), \quad \forall (i,j) \in G'. \tag{4–19}$$

Suppose, $\exists (k,i) \in G'$, $v(k,i) < b(k,i)$. Based on the definition of G', any path from i to the sink must contain an exhausted node. By (4–8), (4–10), (4–12)-(4–13) and the above assumption, we have

$$\sum_{k \in U_i} \alpha \times v(k,i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i \times v(i,j) < e_i$$

In addition, from (4–6)-(4–7) and (4–2)-(4–3), we have that, $\exists (i,j) \in E$, $v(i,j) < b(i,j)$. Following the same token, we know that j must not be exhausted and it also has a downstream link whose volume is smaller than bound. Repeating the above reasoning, we can construct a path all the way to the sink without passing an exhausted node, which

contradicts with the fact that any path from i to the sink must contain an exhausted node.

By (4–5), (4–9), (4–14)-(4–15), (4–2)-(4–3) and (4–19), the ratio of volume to rate is kept constant on any path segment in G' that does not contain an exhausted node (which performs rate reduction). It is easy to see that the lifetime of a restricted feeding source u of a node i can only be equal to or smaller than that of an unrestricted source because, due to rate reduction, the radio of volume to rate will decrease when we traverse a path backward from i to u and cross an exhausted node. □

4.3.6 Termination Conditions

By Theorem 4, we shall terminate DPA when it has stabilized the link volumes, which can be detected by adding a flag that is transitively carried by the control messages. The flag is initially unset. A node sets the flag if it changes a link volume by an amount that is not negligibly small. It is up to the application requirement to decide on how small is negligible. The sink will stop if it does not receive a flag that is set. Alternatively, DPA may also be terminated artificially after a certain number of iterations, or when the resulting lifetime vector meets the application requirement.

4.3.7 Overhead

DPA has a flooding-based design. Flooding would be considered as inefficient for point-to-point tasks such as routing a packet from a source to a destination. But for a global task such as building a volume schedule that involves every node and every link, flooding is the obvious choice that allows every node to participate in the distributed computation.

While the flooding design itself may appear non-innovative, the novelty of DPA is in the details that establishes the constraints and formulas for nodes to perform *localized operations* — iteratively computing their individual volume bounds from rates, volumes from volumes bounds, and rates from volumes with reduction — yet globally, as a *net outcome*, produce a progressively better lifetime vector, approaching to the optimal result.

During each iteration, node i sends two control packets, one BOUND of size $O(|U_i|)$ and one VOL_RATE of size $O(|D_i|)$. Upstream/downstream neighbors represent a subset of all nodes within the communication range of i. The packet size is limited when we choose a small number of upstream/downstream neighbors for routing purpose. We performed many simulations in Section 4.4, which shows that DPA converges quickly towards the optimal lifetime vector. To achieve no more than 5% deviation from the optimal, for networks of 1,000 nodes, less than 25 iterations are needed. In addition, the overhead (i.e. number of iterations) increases slowly with network size.

If the network is designed to collect tens of thousands of data packets from each source, the small overhead of DPA (in tens of control packets per node) is negligible. If the number of iterations is pre-determined, we can take the small energy consumption of DPA into account by reducing the nodes' energy (e_i) for an appropriate amount.

4.3.8 Network Dynamics

After DPA computes a volume schedule, because of network dynamics, the nodes may not always be able to follow the schedule exactly to forward packets. For example, a wireless link may be temporarily down due to environmental noise interference. The packets to be carried by this link will have to be sent to other downstream neighbors. If all outgoing links with non-zero volumes are temporarily down, then packets have to be forwarded to outgoing links with zero volumes. Consequently, the actual lifetime vector will deviate from the one predicted by the volume schedule.

To keep up with changes, DPA may be re-executed to compute a new volume schedule. There is a tradeoff between overhead and better lifetime vector. The frequency of executing DPA is dependent on the amount of overhead allowed. For example, suppose the sink collects aggregate information from the network periodically based on the application model characterized by (4–3), and data packets are longer than control packets (INIT/RATE/BOUND/VOL_RATE). If DPA is allowed to consume no more than 0.5% of

all energy, then the sink may execute DPA for 5 iterations each time after it receives 2,000 data packets.

The only alternative solution [24, 25] in the literature is centralized. It is much harder for a centralized algorithm to handle network dynamics because that requires the sink to collect the complete network information before each execution.

4.4 Simulation

In this section, we use simulations to evaluate the performance of DPA. We begin with a simple network topology with a few nodes and then move to increasingly larger random networks.

4.4.1 A Simple Illustrative Test Case

The first simulation is performed on the routing graph shown in Fig. 4-4, where a circle represents a source node, a square represents a non-source node, and the two numbers beside a node are the initial energy (in Joules) and the source rate (in packets/min), respectively. DPA itself does not dictate how the routing graph should be constructed. Instead, it can work with any routing graph that contain the potential routing paths it can choose from (see Section 4.1). DPA works at the application level; it is independent of which MAC protocol is used. Suppose $\alpha = \beta = 0.000012$ Joule/packet and $\gamma = 0.0000432$ Joule/packet, which are chosen based on the parameters in [23] and will be used in all our simulations.

Tables 4-1 shows the lifetime vectors after the first, second, 10th, and 20th iterations of DPA, as well as the maximum lifetime vector (MLV) in the last column, which is computed numerically based on Hou's centralized algorithm [25]. The result demonstrates that the sequence of lifetime vectors produced by DPA converges rapidly towards MLV. Table 4-2 shows the source volumes that are assigned by DPA to the source nodes after the first, second, 10th, and 20th iterations, as well as the optimal source volumes that produce MLV. Recall that the source volume is the number of packets that a source can

successfully deliver to the sink over its lifetime. Source q has a larger lifetime than w but a smaller source volume. That is because its source rate is smaller.

4.4.2 Convergence Speed of DPA

The second simulation studies the convergence speed of DPA on large sensor networks, each having 500 nodes that are randomly deployed in a $1,000 \times 1,000$ area. The sink consists of 4 base stations, evenly spaced along one edge of the deployment area. Each sensor has a transmission range of 100 and an initial energy of 5 joules. There are 100 data sources randomly selected from the 500 sensors. Their source rates are all 1 packet per minute. The routing graph is constructed based on hop count. Each sensor picks its downstream neighbors from those neighbors that are one hop closer to the sink. (All other simulations will also use random networks produced in the above setting except that, when the network size is not 500 nodes, we change the deployment area proportionally while keeping the same node density.)

Consider the lifetime vector V_x produced by DPA after the xth iteration. We measure how much V_x deviates from MLV by the following two metrics. Let $t_x(s)$ be the lifetime of source s in V_x. Let $t_*(s)$ be the lifetime of s in MLV. The *max deviation* of V_x is defined as

$$\max_{s \in S} \{ \frac{|t_x(s) - t_*(s)|}{t_*(s)} \},$$

and the *avg deviation* is defined as

$$\frac{1}{|S|} \sum_{s \in S} \frac{|t_x(s) - t_*(s)|}{t_*(s)}.$$

Fig. 4-5 shows the avg/max deviations of lifetime vectors produced by DPA on 500-node sensor networks. The deviations drop quickly to an insignificant level after a small number of iterations. Each of the data points used to produce the figures in this section is the average of 100 simulation runs on different random networks. Table 4-3 presents some data points for Fig. 4-5. For example, the avg/max deviations are merely 0.066 and 0.013 respectively after 20 iterations — that means, in the worse case, the lifetime of any

source deviates from its optimal value by no more than 6.6%, and on the average case, the lifetime of a source deviates from the optimal by 1.3%.

4.4.3 Scalability of DPA

We evaluate the scalability of DPA on random networks of 500 to 3,000 nodes (with 20% being sources). We set a *target* (avg or max) deviation to be 0.025, 0.05, 0.075 or 0.1. We then count the number of iterations that DPA has to perform in order to produce a lifetime vector whose deviation is bounded by the target value. The simulation results are presented in Fig. 4-6. It shows that the *overhead* for DPA to satisfy a target deviation, *which is measured by the number of iterations*, grows slowly with the network size. Recall that a node sends at most 2 small control packets in each iteration. Even for a network of 3,000 nodes, only 12 iterations are needed to achieve an avg deviation of 5%, and 32 iterations are needed for a max deviation of 5%.

4.4.4 Comparison with Hou's Centralized Algorithm

We use *LP* to stand for Hou's centralized algorithm [25] based on iterative linear programming. Our DPA can also be used as a centralized algorithm when the information about the network is available at the sink. The target max deviation for DPA is set to be 0.025. Fig. 4-7 compares the running times of the two algorithms. It shows that DPA are orders of magnitude faster than LP, and the gap widens when the network size increases.

Next we compare the communication overhead of the two algorithms when LP is used as a centralized algorithm while DPA is used as a distributed algorithm. For LP, the sink has to collect network information, including, for each node, source rate (4 bytes), node energy (4 bytes), transmission power γ (4 bytes), node ID, and IDs of its downstream neighbors (2 bytes each). The sink has also to download the resulting volume schedule to the network, which includes, for each node, its source volume and the volumes of its outgoing links (4 bytes each). For DPA, in every iteration, a node sends out the volumes/rates of its outgoing links and the volume bounds of its incoming links (4 bytes each).

The communication overhead of DPA spreads evenly among all nodes. The communication overhead of LP concentrates on nodes surrounding the sink. For 5,000-node random networks, the left plot in Fig. 4-8 shows the nodal communication overhead in ascending order. The overhead is measured by the number of bytes that a node has to transmit. Clearly, some nodes in LP (at the right end of the figure) bear a huge burden of communication overhead.

The right plot in Fig. 4-8 shows the maximum nodal overhead with respect to network size. The maximum nodal overhead of LP increases much faster than that of DPA.

4.4.5 Comparison with Other Centralized and Distributed Solutions

We compare DPA with two additional algorithms: SLP (following the same name used in [24]) that is a linear programming solution for maximizing the minimum lifetime of all sources, and MPR (Minimum-Power Routing [17, 22]) that is a distributed algorithm for energy-efficient routing.

First we run DPA, SLP, and MPR on 100-node random networks (with all nodes being sources). Fig. 4-9 compares the lifetime vectors produced by the algorithms. Each curve represents the lifetime vector in ascending order generated from one of the three algorithms. The smallest lifetime in the vector produced by DPA is more than 100% larger than that by MPR. For SLP, the result shows that maximizing the minimum *lifetime* of sources does not maximize the *lifetime vector* of the network. DPA produces far better source lifetimes in the lower three quarters of the vector. Second, we compare the algorithms on larger networks. Fig. 4-10 shows the avg/max deviations of the lifetime vectors produced by SLP and MPR on networks of 500 to 3,000 nodes (with 20% being sources). The deviations are large when comparing with those of DPA, which can be made arbitrarily small.

4.5 Summary

We have proposed a distributed progressive algorithm for maximizing the lifetime vector in a wireless sensor network, the first algorithm of its kind for this problem. The design of the algorithm was based on the necessary and sufficient conditions that we have proved for producing the maximum lifetime vector. Simulations are performed to demonstrate the performance of the algorithm.

sources	1st iter.	2nd iter.	10th iter.	20th iter.	MLV
k	41.9	41.9	41.3	41.9	41.9
v	41.9	41.9	41.3	41.9	41.9
u	36.9	63.2	131.2	129.4	125.8
x	33.6	60.7	109.0	121.6	125.8
m	129.9	146.2	158.9	157.3	157.3
j	108.8	154.2	160.2	157.3	157.3
w	151.0	140.5	156.7	157.3	157.3
q	335.5	200.5	239.5	251.7	251.6

Table 4-1. Data source lifetimes (in days)

sources	1st iter.	2nd iter.	10th iter.	20th iter.	MLV
k	60.4	60.4	59.4	60.4	60.4
v	120.8	120.8	118.8	120.8	120.8
u	53.1	90.9	188.9	186.3	181.2
x	48.3	87.5	156.9	175.1	181.2
m	187.0	210.6	228.8	226.5	226.4
j	156.6	222.0	230.6	226.5	226.4
w	434.8	404.6	451.2	453.0	452.9
q	483.1	288.7	344.8	362.5	362.3

Table 4-2. Data source volumes (in thousands of packets)

	1st iter.	10th iter.	20th iter.	30th iter.
max dev.	2.28	0.25	0.066	0.031
avg dev.	0.39	0.045	0.013	0.007

Table 4-3. Some data points used to produce Fig. 4-5

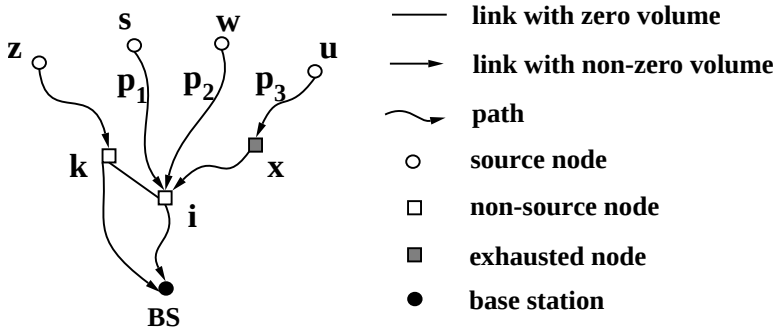


Figure 4-1. There is no exhausted node on P_1 or P_2; nodes s and w are unrestricted feeding sources of i. There is an exhausted node x on P_3; node u is a restricted feeding source of i. There is no forwarding path from z to i; node z is a potential source of i.

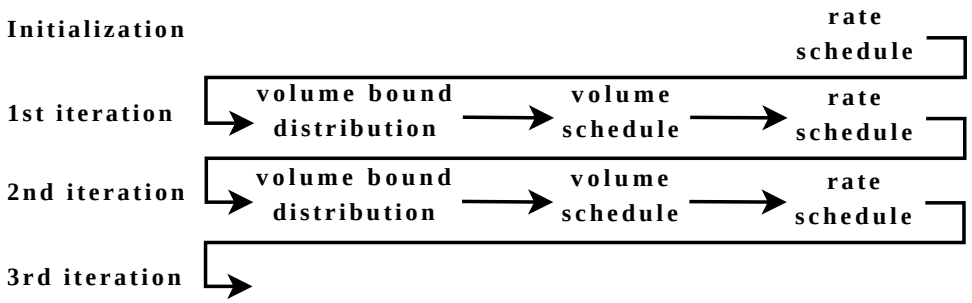


Figure 4-2. Iterations of DPA

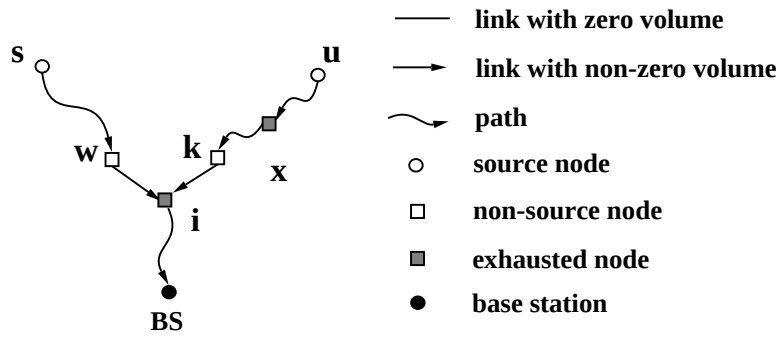


Figure 4-3. There is no exhausted node from s to i; node s is an unrestricted feeding sources of i. There is an exhausted node x from u to i; node u is a restricted feeding source of i. The upstream bottleneck x may prevent source u from fully utilizing the volume bound set by i on link (k, i).

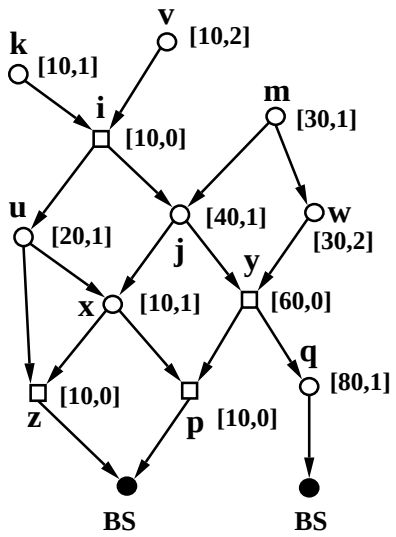


Figure 4-4. A simple illustrative test case.

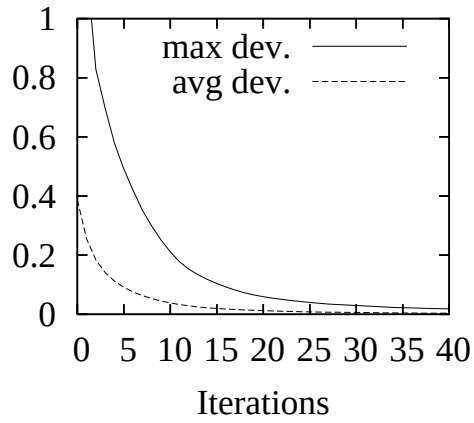


Figure 4-5. max deviation and avg deviation of lifetime vector with respect to the number of iterations that DPA has performed

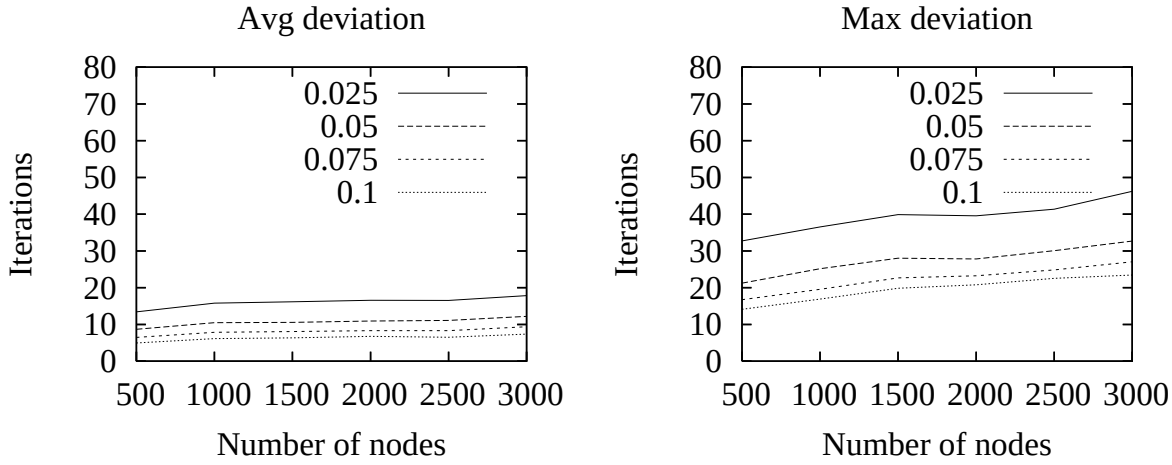


Figure 4-6. DPA scales well. Its overhead grows slowly with the network size.

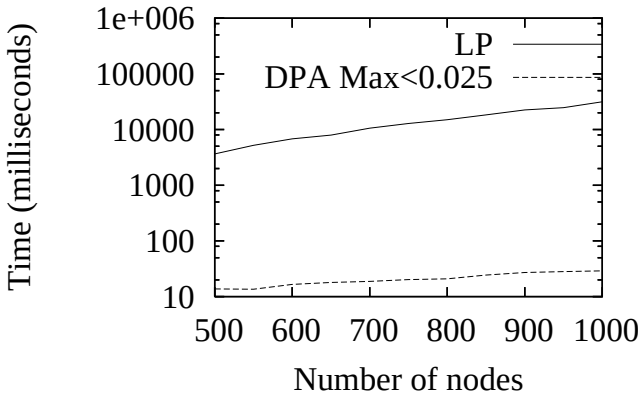


Figure 4-7. Comparison of running time between LP and DPA

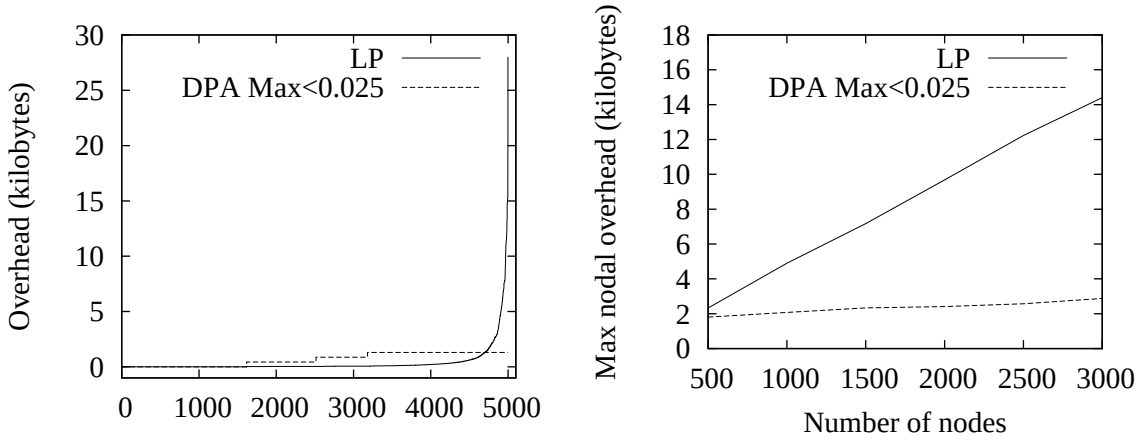


Figure 4-8. *Left plot*: comparison of nodal overhead distribution between LP and DPA. *Right plot*: comparison of maximum nodal overhead between LP and DPA

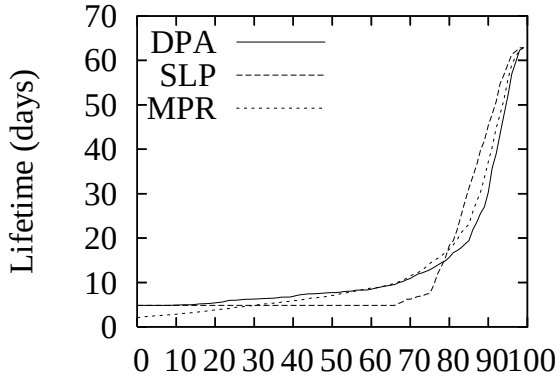


Figure 4-9. Network lifetimes of DPA, SLP and MPR

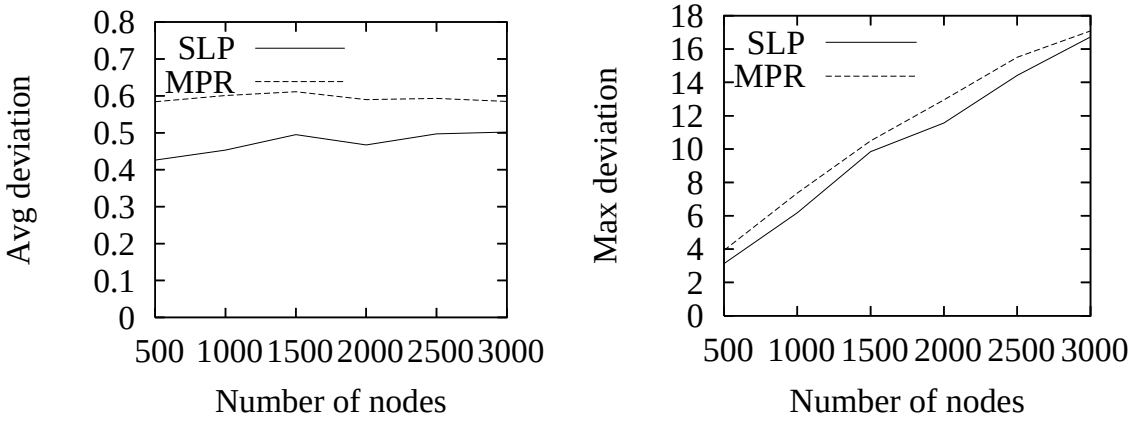


Figure 4-10. Avg and max deviations of SLP and MPR

CHAPTER 5
CONCLUSION

Two important problems in multihop wireless networks are studied. They are end-to-end flow rate fairness and lifetime fairness.

We propose two approaches to achieve global end-to-end flow rate maxmin in multihop wireless networks. The first approach is a cross-layer design. A generalized maxmin model is first proposed for multihop wireless networks. At the network layer, our design allocates network capacity to end-to-end flows for maxmin bandwidth allocation. At the MAC layer, it achieves the allocated bandwidth shares for the flows through a two-level weighted fair queuing algorithm. We demonstrate the effectiveness of the proposed solution in enhancing end-to-end fairness. The second approach proposed is a fully distributed approach. We transform the global maxmin objective to four local conditions and prove that, if the four local conditions are satisfied in the whole network, then the global maxmin objective must be achieved. We then design a distributed rate adaptation protocol based on the four conditions. Our approach does not modify the backoff scheme of IEEE 802.11. It replaces per-flow queueing with per-destination queueing. Most important, it achieves far better fairness (or weighted fairness) among end-to-end flows than existing approaches.

We propose a distributed progressive algorithm for maximizing the lifetime vector in a wireless sensor network, the first algorithm of its kind for this problem. The design of the algorithm is based on the necessary and sufficient conditions that we have proved for producing the maximum lifetime vector. With our progressive algorithm, a result is available at any time and is getting better as more time is spent. We demonstrate that the algorithm is able to converge rapidly towards the maximum lifetime vector with low overhead.

REFERENCES

[1] "IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Quality of Service Enhancements. IEEE Std 802.11e." (Nov 2005).

[2] Akyol, U., Andrews, M., Gupta, P., Hobby, J., Saniee, I., and Stolyar, A. "Joint Scheduling and Congestion Control in Mobile Ad-Hoc Networks." *Proc. of IEEE INFOCOM'08* (2008).

[3] amd S. Sahni, J. Park. "An Online Heuristic for Maximum Lifetime Routing in Wireless Sensor Networks." *IEEE Transactions on Computers* 55 (2006).8: 1048–1056.

[4] Balakrishnan, H., Padmanabhan, V., Seshan, S., and Katz, R. "A comparisonof mechanisms for improving TCP performance over wireless links." *Proc. of ACM SIGCOMM'96* (1996).

[5] Bejerano, Y., Han, S.-J., and Li, L. E. "Fairness and load balancing in wireless lans using association control." *Proc. of ACM MobiCom'04* (2004).

[6] Bertsekas, Dimitri and Gallager, Robert. *Data networks.* Prentice-Hall Inc, 1992, 2nd ed.

[7] Bhardwaj, M. and Chandrakasan, A.P. "Bounding the lifetime of sensor networks via optimal role assignments." *Proc. of IEEE INFOCOM'02* 3 (2002): 1587C1596.

[8] Blough, D. and Santi, P. "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks." *Proc. of ACM MobiCom'02* (2002): 183C192.

[9] Bonomi, F. and Fendick, K. "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service." *IEEE Network* (1995): 9(2):25–39.

[10] Bose, P., Morin, P., Stojmenovic, I., and Urrutia, J. "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks." *Proc. of 3rd Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM'99)* (1999).

[11] Chang, J. and Tassiulas, L. "Energy conserving routing in wireless ad-hoc networks." *Proc. of IEEE INFOCOM'00* (2000).

[12] Chen, L., Low, S. H., Chiang, M., and Doyle, J. C. "Cross-layer Congestion Control, Routing, and Scheduling Design in Ad Hoc Wireless Networks." *Proc. of IEEE INFOCOM'06* (2006).

[13] Chen, S. and Yang, N. "Congestion Avoidance based on Light-Weight Buffer Management in Sensor Networks." *IEEE Transactions on Parallel and Distributed*

Systems, Special Issue on Localized Communication and Topology Protocols for Ad Hoc Networks 17 (2006).9.

[14] Chen, S. and Zhang, Z. "Localized Algorithm for Aggregate Fairness in Wireless Sensor Networks." *Proc. of ACM Mobicom'06* (2006).

[15] Chen, X., Zhai, H., Wang, J., and Fang, Y. "TCP Performance over Mobile Ad Hoc Networks." *Canadian Journal of Electrical and Computer Engineering* 29 (2004): 129–134.

[16] Chiu, D. and Jain, R. "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks." *Journal of Computer Networks and ISDN* 17 (1989).1: 1–14.

[17] Doshi, S., Bhandare, S., and Brown, T.X. "An on-demand minimum energy routing protocol for a wireless ad hoc network." *ACM Mobile Computing and Communications Review* 6 (2002).3.

[18] Faffe, Jeffrey M. "Bottleneck Flow Control." *IEEE Transactions on Communications* COM-29 (1981).7: 954–962.

[19] Fang, Z. and Bensaou, B. "Fair Bandwidth Sharing Algorithms based on Game Theory Frameworks in Wireless Ad-Hoc Networks." *Proc. of IEEE INFOCOM'04* (2004).

[20] Gambiroza, V., Sadeghi, B., and Knightly, E. W. "End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks." *Proc. of Mobicom'04, Philadelphia, PA, USA* (2004).

[21] Gao, Q., Zhang, J., and Hanly, S. "Cross-Layer Rate Control in Wireless Networks with Lossy Links: Leaky-Pipe Flow, Effective Network Utility Maximization and Hop-by-Hop Algorithms." *Proc. of IEEE INFOCOM'08* (2008).

[22] Gomez, J., Campbell, A., Naghshineh, M., and Bisdikian, C. "Conserving transmission power in wireless ad hoc networks." *Proc. of 9th International Conference on Network Protocols (ICNP)* (2001).

[23] Heinzelman, W. "Application-Specific Protocol Architecture for Wireless Networks." *Ph.D. Thesis, MIT* (2000).

[24] Hou, Y. T., Shi, Y., and Sherali, H. "On Lexicographic Max-Min Node Lifetime for Wireless Sensor Networks." *Proc. of IEEE ICC'04* 7 (2004): 3790–3796.

[25] Hou, Y. Thomas, Shi, Yi, and Sherali, Hanif D. "Rate Allocation in Wireless Sensor Networks with Network Lifetime Requirement." *Proc. of ACM MobiHoc'04* (2004): 67–77.

[26] Huang, X. L. and Bensaou, B. "On Max-Min Fairness and Scheduling in Wireless Ad-hoc Networks: Analytical Framework and Implementation." *Proc. of MobiHoc'01, Long Beach, California* (2001).

[27] Kalpakis, K., Dasgupta, K., and Namjoshi, P. "Maximum lifetime data gathering and aggregation in wireless sensor networks." *Proc. of IEEE ICN'02* (2002).

[28] Kar, K., Sarkar, S., and Tassiulas, L. "Achieving Proportionally Fair Rates Using Local Information in Aloha Networks." *IEEE Transactions on Automated Control* 49 (2004).10.

[29] Karp, B. and Kung, H. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks." *Proc. of ACM MobiCom'00* (2000).

[30] Kelly, F., Maulloo, A., and Tan, D. "Rate control in communication networks: shadow prices, proportional fairness and stability." *Journal of the Operational Research* 49 (1998).

[31] Kleinberg, J. M., Rabani, Y., and Tardos, E. "Fairness in Routing and Load Balancing." *IEEE Symposium on Foundations of Computer Science* (1999).

[32] Li, B. "End-to-End Fair Bandwidth Allocation in Multi-hop Wireless Ad Hoc Networks." *Proc. of IEEE ICDCS'05* (2005).

[33] Li, Q., Aslam, J., and Rus, D. "Online power-aware routing in wireless Ad-hoc networks." *Proc. of ACM MobiCom'01* (2001): 97–107.

[34] Lin, X. and Shroff, N. B. "The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks." *IEEE/ACM Trans. on Networking* 14 (2006).2.

[35] Low, S. H. and Lapsley, D. E. "Optimization Flow Control, I: Basic Algorithms and Convergence." *IEEE/ACM Transactions on Networking* (1999).

[36] Luo, H., Cheng, J., and Lu, S. "Self-Coordinating Localized Fair Queueing in Wireless Ad Hoc Networks." *IEEE Transactions on Mobile Computing* 3 (2004).1.

[37] Luo, H., Lu, S., and Bharghavan, V. "A New Model for Packet Scheduling in Multihop Wireless Networks." *Proc. of MobiCom'00* (2000).

[38] Madan, R., Luo, Z. Q., and Lall, S. "A distributed algorithm with linear convergence for maximum lifetime routing in wireless sensor networks." *Proc. of the Allerton Conference on Communication, Control and Computing* (2005).

[39] Massoulie, L. and Roberts, J. "Bandwidth Sharing: Objectives and Algorithms." *EEE Transactions on Networking* 10 (2002).3.

[40] Mo, J. and Walrand, J. "Fair End-to-End Window-Based Congestion Control." *IEEE/ACM Trans. on Networking* 8 (2000).5.

[41] Mosely, Jeannine. *Asynchronous distributed flow control algorithms, Ph.D. thesis.* Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1984.

[42] Perkins, C. E. and Royer, E. M. "Ad Hoc On-demand Distance Vector Routing." *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications* (1999).

[43] Qiu, Y. and Marbach, P. "Bandwidth Allocation in Wireless Ad-Hoc Networks: A Price-based Approach." *Proc. of IEEE INFOCOM'03* (2003).

[44] Radunovic, B. and Boudec, J. Le. "A Unified Framework for Max-Min and Min-Max Fairness with Applications." *Proc. of Allerton?2* (2002).

[45] Rodoplu, V. and Meng, T. "Minimum energy mobile wireless networks." *IEEE Journal of Selected Areas in Communications* 17 (1999).8: 1333–1344.

[46] Sankar, A. and Liu, Z. "Maximum Lifetime Routing in Wireless Ad-hoc Networks." *Proc. of IEEE INFOCOM'04* (2004).

[47] Sarkar, S. and Tassiulas, L. "End-to-end Bandwidth Guarantees Through Fair Local Spectrum Share in Wireless Adhoc Networks." *IEEE Transactions on Automatic Control* 50 (2005).9.

[48] Shi, Y. and Hou, T. "Theoretical Results on Base Station Movement Problem for Sensor Networks." *Proc. of IEEE INFOCOM* (2008).

[49] Singh, S., M.Woo, and Raghavendra, C. "Power-aware routing in mobile ad-hoc networks." *Proc. of MOBICOM (1998)* (1998): 181–190.

[50] Stojmenovic, I. and Lin, X. "Power-aware localized routing in wireless networks." *IEEE Tran. on Parallel and Distributed Systems* 12 (2001).11: 1122–1133.

[51] Tassiulas, L. and Sarkar, S. "Maxmin Fair Scheduling in Wireless Networks." *Proc. of IEEE INFOCOM'02* (2002).

[52] Wang, X. and Kar, K. "Distributed Algorithms for Max-min Fair Rate Allocation in Aloha Networks." *Proc. of the 42nd Annual Allerton Conference, Urbana-Champaign* (2004).

[53] Wattenhofer, R., Li, L., Bahl, P., and Wang, Y. "Distributed topology control for power efficient operation in multihop wireless ad hoc networks." *Proc. of INFOCOM (2001)* (2001).

[54] Wu, Y., Fahmy, S., and Shroff, N. B. "On the Construction of a Maximum-Lifetime Data Gathering Tree in Sensor Networks: NP-Completeness and Approximation Algorithms." *Proc. of IEEE INFOCOM'08* (2008).

[55] Xue, Y., Li, B., and Nahrstedt, K. "Optimal Resource Allocation in Wireless Ad Hoc Networks: A Price-based Approach." *IEEE Transactions on Mobile Computing* 5 (2006).4.

[56] Yi, Y. and Shakkottai, S. "Hop-by-Hop Congestion Control over a Wireless Multi-hop Network." *Proc. of IEEE INFOCOM'04, Hong Kong, China* (2004).

[57] Zhai, H. and Fang, Y. "Distributed Flow Control and Medium Access in Multihop Ad Hoc Networks." *IEEE Transactions on Mobile Computing* 5 (2006).11.

[58] Zhu, Junhua, Chen, Shan, Bensaou, Brahim, and Hung, Ka-Lok. "Tradeoff between Lifetime and Rate Allocation in Wireless Sensor Networks: A Cross Layer Approach." *Proc. of IEEE INFOCOM'07* (2007).

[59] Zussman, G. and Segall, A. "Energy Efficient Routing in Ad Hoc Disaster Recovery Networks." *Proc. of IEEE INFOCOM'03* (2003).

BIOGRAPHICAL SKETCH

Liang Zhang was born in Beijing, China. He received his B.E. and M.E. degrees in computer science and technology from Tsinghua University, China, in 1999 and 2002, respectively. After that, he had worked in Oracle R&D Center in China for one year. In 2003, he joined the Department of Computer and Information Science and Engineering at the University of Florida, to pursue his Ph.D. degree. His advisor is Dr. Shigang Chen. His research focused on fairness in multihop wireless networks.