ON THE THROUGHPUT EFFICIENCY OF GREEDY MAXIMAL SCHEDULING IN WIRELESS AD HOC NETWORKS

BY

MATHIEU LECONTE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Adviser:

Professor Rayadurgam Srikant

ABSTRACT

Due to its low complexity, Greedy Maximal Scheduling (GMS), also known as Longest Queue First (LQF), has been studied extensively for wireless networks. However, GMS can result in degraded throughput performance in general wireless networks. In this thesis, we derive performance bounds of GMS for wireless networks under the general k-hop interference model. In particular, we prove that GMS achieves 100% throughput in all networks with eight nodes or less, under the two-hop interference model. Further, the obtained performance bounds improve upon previous results for larger networks up to a certain size. We also provide a simple proof to show that GMS can be implemented using only local neighborhood information in networks of any size.

To family and friends

ACKNOWLEDGMENTS

I want to thank my adviser, R. Srikant, whose patience I am testing on a regular basis. Many thanks also go to my friends here at the University of Illinois, and especially Christophe and Nicolas, with whom sharing experiences has always been a great source of enjoyment.

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES vii

LIST OF ABBREVIATIONS viii

LIST OF SYMBOLS ix

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 RELATED WORK 4

CHAPTER 3 NETWORK MODEL 6

CHAPTER 4 PRELIMINARIES 10

CHAPTER 5 THROUGHPUT OPTIMALITY OF GMS IN SMALL NETWORKS 13
- 5.1 Main Result and Outline of the Proof 13
- 5.2 Properties of the Unstable Subsets of Links 16
- 5.3 Throughput Optimality for Node-Exclusive Models 20

CHAPTER 6 PERFORMANCE OF GMS UNDER K-HOP INTERFERENCE MODEL 21
- 6.1 Efficiency in Cycles 21
- 6.2 Performance Bounds of GMS in Larger Networks under 2-Hop Interference 23
- 6.3 Performance Bounds for k-Hop Interference Model 25

CHAPTER 7 DECENTRALIZATION OF GMS 28
- 7.1 On the Equivalence of LGMS and GMS 32

CHAPTER 8 CONCLUSION 38

APPENDIX A PROOF OF LEMMA 2 39

REFERENCES 45

LIST OF TABLES

6.1 Efficiency of GMS in Small Networks (2-Hop Interference Model) 26

LIST OF FIGURES

5.1 Two examples of networks of 9 nodes with $\sigma^* < 1$. 15

A.1 Tree of cliques - the 3 possible cases. 42

LIST OF ABBREVIATIONS

ACK	Acknowledgment
CTS	Clear to send
FH-CDMA	Frequency Hopping - Code Division Multiple Access
GMS	Greedy Maximal Scheduling
LGMS	Local Greedy Maximal Scheduling
LGS	Local Greedy Scheduling
LQF	Longest Queue First
MAC	Media Access Control
MWS	Maximum Weight Scheduling
NP	Non-polynomial
OLoP	Overall local pooling
RTS	Request to send
SLoP	Subset local pooling

LIST OF SYMBOLS

k	Minimum number of hops that separate two non-interfering links
n	Number of nodes of the network
$\mathbf{G}$	Network graph
$\mathbf{V}$	Set of the vertices (or nodes) of $\mathbf{G}$
$\mathbf{E}$	Set of the edges (or links) of $\mathbf{G}$
$\mathbf{U}$	An unstable subgraph of G
$\mathbf{I}(l)$	Set of links interfering with link l
$\mathbf{G_I}$	Interference graph associated with the network graph $\mathbf{G}$
$\mathbf{V_I}$	Set of vertices of $\mathbf{G_I}$
$\mathbf{E_I}$	Set of edges of $\mathbf{G_I}$
$\mathcal{M}_\mathbf{L}$	Set of the rate vectors of all maximal schedules of $\mathbf{L}$
$Co(\mathcal{M}_\mathbf{L})$	Convex hull of the set $\mathcal{M}_\mathbf{L}$
Λ	Capacity region of a network
$w(x)$	Weight of a x, where x can be a single link or a schedule
γ^*	Efficiency ratio of a scheduling policy
σ^*	Local pooling factor of a graph
T_l	Backoff time
GMS($\mathbf{L}$)	Schedule produced by GMS when applied to the set of links $\mathbf{L}$
GMS($\mathbf{L}$)	Schedule produced by LGMS when applied to the set of links $\mathbf{L}$
π	Tie-breaking mechanism
$d(x, y)$	Distance between x and y

CHAPTER 1

INTRODUCTION

In wireless communication networks with limited resources, efficient resource allocation plays an important role in achieving high performance and providing good quality of service. In this thesis, we study *link scheduling* for wireless networks, where all links (node pairs) may not be able to simultaneously transmit due to transceiver constraints and/or radio interference. A *scheduling algorithm* determines which links can transmit at each time instant so that no two active links interfere with each other.

The performance metric of interest in this thesis is *throughput* and we restrict our attention to MAC layer (or link-level) throughput as opposed to end-to-end throughput. It is well known that the queue-length based Maximum Weighted Scheduling (MWS) algorithm is *throughput optimal* [1], in the sense that it can stabilize the queues in the network for all traffic rates in the capacity region of the network. However, MWS has a high computational complexity. In addition, MWS is not amenable to distributed implementation. These drawbacks greatly limit the deployment of MWS in real networks. Even in small networks, MWS can require quite a lot of operations because its complexity is tied to the number of maximal schedules of the network. As an example, we can observe that an 8-cycle and a complete graph of 8 nodes have respectively 48 and 56 distinct maximal schedules if we consider directed links and the 2-hop interference model. So, although those networks are small, decentralization of MWS is still not practical for them.

Therefore, it is of interest to find simple, distributed scheduling algorithms which can achieve optimal or near optimal performance.

While much attention has been focused in the literature on finding near-optimal algorithms for very large networks, many currently used ad hoc or mesh networks are of moderate size ranging from a few nodes to a few tens of nodes. Applications of this type include military and civilian networks. Thus, we focus our attention in this thesis on obtaining performance bounds for small to moderately sized networks although our results on information complexity apply to general networks.

Our contributions in this thesis are as follows:

- We prove that GMS is throughput optimal in small networks. Specifically, we show throughput optimality in networks with 8 or fewer nodes under the 2-hop interference model. This result is tight in the sense that we can find networks with 9 nodes where GMS is not throughput optimal under the 2-hop interference model.

- As a by-product of the above result, we also establish the throughput-optimality of GMS in networks with up to 5 nodes under the 1-hop interference model. This rigorously proves a numerical observation in [2].

- We derive bounds on the efficiency ratio of GMS for arbitrary networks. These bounds improve previous results [3] for networks with up to 26 nodes.

- We show that GMS, which requires global knowledge of link weights, is equivalent to an algorithm called Local GMS (LGMS) [4], which uses only local neighborhood information and thus is amenable to distributed implementation.

This thesis is organized as follows. In Chapter 2 we introduce the related work. We describe the network model in Chapter 3 and provide some useful notions in Chapter 4. In Chapter 5, we prove that GMS is throughput optimal in small networks with up to 8 nodes under the 2-hop interference model. Chapter 6 provides a lower bound on the efficiency ratio of GMS as a function of the network size under the k-hop interference model. We consider decentralization of GMS and prove the equivalence of GMS and LGMS in Chapter 7.

CHAPTER 2

RELATED WORK

The Greedy Maximal Scheduling (GMS) algorithm, also known as the Longest-Queue-First (LQF) algorithm [5], has low complexity and hence can be deployed in practical systems. Its performance has been observed to be close to optimal for a variety of network scenarios in simulations and experiments (e.g., [6]). Hence it is intriguing and important to analyze and understand the performance of GMS for networks with common topologies; this task may include providing sufficient conditions for GMS to be throughput optimal, and calculating/bounding the efficiency ratio of GMS when it is not throughput optimal.

To identify sufficient conditions for GMS to be throughput optimal, the concept of *local pooling* (will be defined formally later) was introduced in [7]. The authors showed that if the network satisfies the local pooling condition then GMS is throughput optimal. In particular, if the interference graph of the network is a tree, then the local pooling condition is satisfied and GMS is throughput optimal. In addition to tree (interference) graphs, [2,8] identified several classes of graphs (e.g., trees of cliques, perfect graphs, chordal graphs) which also satisfy the local pooling condition. Independently [3] and [8] showed that the local pooling condition is satisfied for tree networks (note that the interference graph of a tree network may not be a tree) under the k-hop interference model.

In [2,8], the local pooling condition was tested for small interference graphs

via exhaustive numerical search. They found that local pooling is satisfied (GMS is throughput optimal) for networks with up to 5 links under the 1-hop interference model and for networks with up to 7 links under the k-hop ($k \geq 2$) interference model.

In [3,9], the notion of local pooling was generalized to σ-local pooling and the concept of *local-pooling factor* was introduced. They showed that the efficiency ratio of GMS is equal to the local-pooling factor of the network and proposed a recursive procedure to estimate/bound the local-pooling factor of a network. Our results primarily build upon the results in [7] and [9].

Compared with GMS, the Local GMS (LGMS) algorithm [4] has an even lower complexity (its computational complexity is linear in the number of links in the network), and is amenable to distributed implementation [10]. In this thesis, we show that LGMS and GMS are equivalent in the sense that they produce the same set of schedules for arbitrary networks.

CHAPTER 3

NETWORK MODEL

We assume a time-slotted system, with time slots of unit duration. In addition, here we focus on the MAC layer and thus we only consider 1-hop traffic.

We model a wireless network by a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of nodes/vertices and $\mathbf{E}$ is the set of directed links/edges. Nodes are wireless transmitters/receivers and there exists a link between two nodes if they can directly communicate with each other. For any link $l \in \mathbf{E}$, we can define the set of its interfering links as

$$\mathbf{I}(l) = \big\{ l' \in \mathbf{E} \mid l' \text{ interfers with } l \big\}. \tag{3.1}$$

Associated with a network graph $\mathbf{G}$, we can define an interference graph $\mathbf{G_I} = (\mathbf{V_I}, \mathbf{E_I})$, where $\mathbf{V_I} = \mathbf{E}$ and there is an edge in $\mathbf{G_I}$ from l to l' if $l' \in \mathbf{I}(l)$. This graph contains all the information regarding the interference model used.

A *schedule* of $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is a subset of links $\mathbf{M} \subseteq \mathbf{E}$ that can be activated/scheduled at the same time according to the interference constraint, i.e., no two links in $\mathbf{M}$ interfere with each other. We assume that all links have unit capacity, i.e. a scheduled link can transmit one packet per unit time. Associated with a schedule $\mathbf{M}$ is a rate vector $\overrightarrow{m} \in \{0,1\}^{|\mathbf{E}|}$. The l^{th} element of $\overrightarrow{m}$ is equal to 1 ($m_l = 1$) if link l is scheduled ($l \in \mathbf{M}$); $m_l = 0$

otherwise. Then, a convex combination of schedules will be an element of $[0,1]^{|\mathbf{E}|}$.

A schedule is said to be *maximal* if no link can be added to it without violating the interference constraint. Denote by $\mathcal{M}_{\mathbf{E}}$ the set of the rate vectors of all maximal schedules of $\mathbf{E}$ and by $Co(\mathcal{M}_{\mathbf{E}})$ its convex hull.

A *scheduling algorithm* is a procedure to decide which schedule should be used (i.e., which subset of links should be activated) in every time slot. The *capacity region* of the network is the set of all arrival rates $\overrightarrow{\lambda}$ for which there exists a scheduling algorithm that can stabilize the queues, i.e., the queues are bounded in some appropriate stochastic or deterministic sense depending on the arrival model used. For the purposes of this thesis, we will assume that if the arrival process is stochastic, then it is a stationary, ergodic process with finite first and second moments, and the resulting queue length process admits a Markovian description. Alternatively, one can also assume that the arrival process is deterministically bounded like the well-known (σ, ρ) process. It is known from [1] that the capacity region is given by

$$\Lambda = \{ \overrightarrow{\lambda} \mid \exists \overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{E}}), \overrightarrow{\lambda} < \overrightarrow{u} \}. \tag{3.2}$$

(When dealing with vectors, inequalities are interpreted component-wise.) We say that a scheduling algorithm is *throughput optimal*, or achieves the *maximum throughput*, if it can keep the network stable for all arrival rates in the capacity region Λ.

Suppose each link $l \in \mathbf{E}$ is associated with a non-negative weight $w(l)$. Let $w(\mathbf{M}) = \sum_{l \in \mathbf{M}} w(l)$ be the weight of schedule $\mathbf{M}$. A *maximum weighted schedule* of $\mathbf{G}$ is a schedule which has the maximum weight among all schedules of $\mathbf{G}$. Note that if all link weights are strictly positive, then a maximum

weighted schedule must be a maximal schedule.

It is well known that the *Maximum Weighted Scheduling* (MWS) algorithm, which selects a maximum weighted schedule in every time slot with the link weights being the link queue lengths, is throughput optimal [1]. However, finding a maximum weighted schedule of a network is equivalent to finding a maximum weighted independent set of the associated interference graph, which is known to be NP-hard for general interference graphs [11]. In addition, MWS is centralized in nature and is very difficult to decentralize.

The *Greedy Maximal Scheduling* (GMS) algorithm, which is a natural low-complexity alternative to MWS, proceeds as follows: in each time slot the schedule that will be used is built sequentially; start with an empty schedule; at each step, choose a link l with maximum weight among the non-disabled links and add it to the current schedule, then disable all links which interfere with l; continue until all the remaining links are disabled. Note that any schedule obtained by GMS is maximal.

Greedy Maximal Scheduling (GMS) Algorithm

$\text{GMS}(\mathbf{E}) := \emptyset$

$\mathbf{E}' := \mathbf{E}$

WHILE $(\mathbf{E}' \neq \emptyset)$

Pick a globally heaviest link l:

$$w(l) \;=\; \max_{l' \in \mathbf{E}'} w(l')$$

$\text{GMS}(\mathbf{E}) := \text{GMS}(\mathbf{E}) \cup \{l\}$

$\mathbf{E}' := \mathbf{E}' \setminus \{l\} \setminus \mathbf{I}(l)$

ENDWHILE

Since GMS is a greedy algorithm, in general, it may not achieve the full capacity region of the network. The *efficiency ratio* γ^* of a scheduling algorithm is the largest fraction of the capacity region that is stabilized by the algorithm.

$$\begin{aligned} \gamma^* &= \sup\{\gamma \mid \forall \overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{E}}), \forall \overrightarrow{\lambda} \leq \gamma \overrightarrow{u}, \text{ the system} \\ &\qquad \text{is stable under arrival rate } \overrightarrow{\lambda}\}. \end{aligned} \tag{3.3}$$

Throughout this thesis, we will often focus on the k-hop interference model. This model says that two links interfere with each other if and only if the shortest path in $\mathbf{E}$ between them is of length at most $k-1$. The 1-hop and 2-hop interference models are of special practical interest. Under the 1-hop interference model (also called the *node-exclusive* or *primary* interference model), any two links sharing a common node cannot be active simultaneously. This can be used to describe and analyze wireless networks which use FH-CDMA with only one transceiver per node [12]. The 2-hop interference model is well suited to model networks that use RTS/CTS and link-level ACKs, which many practical communication schemes do (such as IEEE 802.11 [13], [14]).

CHAPTER 4

PRELIMINARIES

In [7], the notion of *local pooling* was introduced. Local pooling is better explained when broken down into two parts [2]: subset local pooling (SLoP) and overall local pooling (OLoP).

Definition 1 (subset local pooling - SLoP) *A set of links* $\mathbf{L}$ *satisfies subset local pooling if* $\exists \overrightarrow{\alpha} \in \mathbb{R}_+^{|\mathbf{L}|}$ *and* $c > 0$*, such that* $\forall \overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{L}})$, $\overrightarrow{\alpha}^T \overrightarrow{u} = c$.

Note that the interference constraints used to compute maximal schedules of $\mathbf{L} \subseteq \mathbf{E}$ are given by the interference graph $\mathbf{G_I}$ of $\mathbf{G}$.

One way to understand the subset local pooling condition is to view $\alpha_l \geq 0$ as the utility obtained when receiving a unit of service on link l. Then $\overrightarrow{\alpha}^T \overrightarrow{u} = c$ is the total utility associated with the service rate vector $\overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{L}})$. If the total utility for any rate vector is a constant, then there is no way a vector $\overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{L}})$ can strictly dominate another vector $\overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}})$, because otherwise $\overrightarrow{u}$ would provide a strictly larger total utility than $\overrightarrow{v}$.

Definition 2 (overall local pooling - OLoP) *A graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *satisfies overall local pooling if all subsets of* $\mathbf{E}$ *satisfy subset local pooling.*

It was proved in [7] that GMS is throughput optimal for a network if the network graph satisfies overall local pooling. Although local pooling is useful when trying to determine whether GMS is throughput optimal for a network,

it is also of interest to get a sense of how well GMS performs when it is not optimal. To answer this question, the *local pooling factor* σ^* of a graph was introduced in [9]. This factor gives a bound on how bad the schedules picked by GMS can be, compared to the optimal ones. As for local pooling, this notion is better explained when broken down into parts.

Definition 3 (σ-local pooling) *A set of links* $\mathbf{L}$ *satisfies* σ*-local pooling if* $\forall \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}}), \sigma \overrightarrow{u} \not\geq \overrightarrow{v}$.

Definition 4 (local pooling factor) *The local pooling factor of a graph* $\mathbf{G}$ *is the supremum of all* σ *such that every subset* $\mathbf{L}$ *of* $\mathbf{E}$ *satisfies* σ*-local pooling.*

$$\begin{aligned} \sigma^* &= \sup\{\sigma | \forall \mathbf{L} \subseteq \mathbf{E},\ \forall \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}}), \sigma \overrightarrow{u} \not\geq \overrightarrow{v}\} \\ &= \inf\{\sigma | \exists \mathbf{L} \subseteq \mathbf{E},\ \exists \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}}), \sigma \overrightarrow{u} \geq \overrightarrow{v}\}. \end{aligned}$$

It was proved in [9] that the efficiency ratio γ^* of GMS in a graph is equal to its local pooling factor σ^*. Note that if a set of links satisfies subset local pooling, then it satisfies σ-local pooling for all $\sigma < 1$.

If GMS is not throughput optimal in a graph, we know that, under some feasible arrival rate, there exist links whose queues will go up to infinity. To analyze those links, [9] introduces the notion of *unstable subset of links*, which is a subset of links whose queues can all go to infinity under GMS under some feasible arrival rate. We use the following slightly different definition:

Definition 5 (unstable subset of links) *Let* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *be a network graph and* $\mathbf{U} \subseteq \mathbf{E}$. *We say that* $\mathbf{U}$ *is an unstable subset of links of* $\mathbf{G}$ *if* $\exists \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{U}})$ *such that* $\overrightarrow{u} > \overrightarrow{v}$.

Note that $\overrightarrow{u} > \overrightarrow{v}$ if and only if $\exists \sigma < 1$ such that $\sigma \overrightarrow{u} \geq \overrightarrow{v}$. Thus, the unstable subsets of links of $\mathbf{G}$ are the subsets of $\mathbf{E}$ that fail σ-local pooling

for some $\sigma < 1$. This implies that the local pooling factor σ^* (and thus the efficiency ratio of GMS γ^*) of a graph $\mathbf{G}$ is strictly less than 1 if and only if there exists at least one unstable subset of links of $\mathbf{G}$. So, to prove that GMS is throughput optimal in a graph $\mathbf{G}$, we only need to show that $\mathbf{G}$ has no unstable subsets of links.

Also, if a subset of links $\mathbf{U}$ is unstable, then there exists a feasible arrival rate $\overrightarrow{\lambda} \in Co(\mathcal{M}_{\mathbf{U}})$ such that, for all $l \in \mathbf{U}$, the average service rate u_l of l under GMS is strictly less than the average arrival rate λ_l of l. In other words, there exists a feasible arrival rate such that the queues of all the links of $\mathbf{U}$ go to infinity. Such a feasible arrival pattern can be constructed as in [9]. Moreover, it is clear that if a set of links $\mathbf{L}$ satisfies subset local pooling then it cannot be an unstable subset of links, because otherwise $\exists \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}})$ such that $\overrightarrow{u} > \overrightarrow{v}$ and then $\overrightarrow{\alpha}^T \overrightarrow{u} > \overrightarrow{\alpha}^T \overrightarrow{v}$ for all nonzero $\overrightarrow{\alpha} \in \mathbb{R}_+^{|\mathbf{L}|}$.

CHAPTER 5

THROUGHPUT OPTIMALITY OF GMS IN SMALL NETWORKS

In this chapter we analyze the performance of GMS in small networks. Indeed, most wireless ad hoc networks that one may encounter in reality are small. More precisely, we want to find the maximum network size under which GMS is guaranteed to achieve full capacity. Although some parts of the reasoning apply to any interference model, we focus on the 2-hop interference model in this chapter. We only consider connected network graphs, because the links in different components of disconnected graphs do not interfere with each other under this interference model.

5.1 Main Result and Outline of the Proof

The main result of this section is the following theorem:

Theorem 1 *Under the* 2*-hop interference model, GMS is throughput optimal (achieves the full capacity region) in all network graphs with* 8 *nodes or less.*

To prove that GMS is throughput optimal in graphs with 8 nodes or less, our approach is to prove that such graphs cannot have any unstable subsets of links. The entire proof of the above result is quite complicated. So we provide an outline first, where we will assume the truth of certain intermediate propositions. The proof of the theorem follows rather easily from these intermediate propositions which will be proved later in Section 5.2.

The propositions essentially show that unstable subsets of links have certain properties which imply that there must be at least some minimum number of nodes in the network graph.

Proposition 1 *Any unstable subset of links must contain at least* 3 *links that can be simultaneously scheduled.*

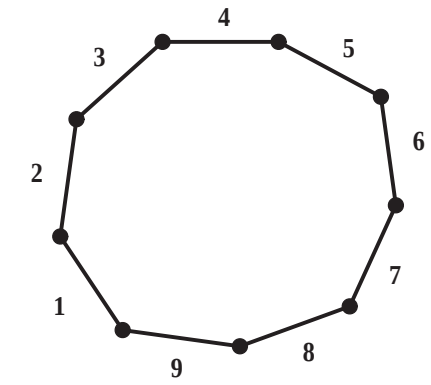
Proposition 2 *Under the* 2*-hop interference model, if* $\mathbf{G}$ *has an unstable subset of links* $\mathbf{U}$*, then a maximal schedule of* $\mathbf{U}$ *of size* 2 *or more disables at least* 3 *nodes of* $\mathbf{G}$*.*

We say that a node is *disabled* if it is exactly 1 hop away from a scheduled link. Since the minimum distance between two scheduled links is 2 hops in the 2-hop interference model, a disabled node cannot be scheduled.

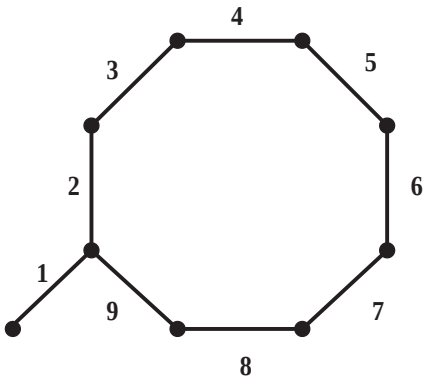
The proof of Theorem 1 is quite straightforward assuming the validity of the above two propositions.

Proof: (Theorem 1) Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a network graph and $\mathbf{U}$ be an unstable subset of links of $\mathbf{G}$. Let $|\mathbf{V}| = n$. Because of Proposition 1, we know there exists a maximal schedule of $\mathbf{U}$ that contains at least 3 links. When using that particular maximal schedule, as each link involves 2 different scheduled nodes, at least 6 nodes are scheduled. Moreover, Proposition 2 tells us that this maximal schedule disables at least 3 other nodes of $\mathbf{G}$. The total number of nodes n must be greater than or equal to the number of scheduled nodes + the number of disabled nodes, hence $n \geq 9$. It implies that any network graph of 8 nodes or less cannot have any unstable subset of links, hence GMS is throughout optimal in those graphs. ■

Before we proceed to the proof of Propositions 1 and 2, we show that the result in Theorem 1 is tight in the sense that there are networks with 9 nodes for which GMS is not throughput optimal.



(a) 9-cycle



(b) 8-cycle with an open link

Figure 5.1: Two examples of networks of 9 nodes with $\sigma^* < 1$.

5.1.1 What about 9 nodes?

We provide here two examples of networks with 9 nodes in which the efficiency of GMS can be bounded away from 1.

The first example is a ring network with 9 links. Call it $\mathbf{C}_9 = (\mathbf{V}, \mathbf{E})$ and label the links as in Figure 5.1(a). We can show that the local pooling factor of $\mathbf{C}_9$ is at most $\frac{2}{3}$. Indeed, let us use the following maximal schedules for equal amounts of time: $\{1, 4, 7\}$, $\{2, 5, 8\}$ and $\{3, 6, 9\}$. The resulting vector $\overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{E}})$ is $\overrightarrow{u} = \frac{1}{3}\overrightarrow{e_{\mathbf{E}}}$. But we may choose to use the following maximal schedules instead: $\{1, 5\}$, $\{2, 6\}$, $\{3, 7\}$, $\{4, 8\}$, $\{5, 9\}$, $\{6, 1\}$, $\{7, 2\}$, $\{8, 3\}$ and $\{9, 4\}$. The resulting vector $\overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{E}})$ is then $\overrightarrow{v} = \frac{2}{9}\overrightarrow{e_{\mathbf{E}}}$. We have

$\overrightarrow{v} = \frac{2}{3}\overrightarrow{u}$, which shows that $\mathbf{E}$ fails σ-local pooling for $\sigma = \frac{2}{3}$, hence $\sigma^* \leq \frac{2}{3}$. In addition, the bound provided by Lemma 4 in Section 6.2 tells us that $\sigma^* \geq \frac{2}{3}$, so finally $\sigma^* = \frac{2}{3}$. Note that this result also implies that the bound of Theorem 2 is tight for $n = 9$ and 10.

The second example we analyze here is a size-8 ring network with one additional open link. Let us call it $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and label the links as in Figure 5.1(b). As previously for $\mathbf{C}_9$, we will construct two vectors $\overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{E}})$ such that $\sigma\overrightarrow{u} \geq \overrightarrow{v}$ componentwise, with $\sigma < 1$. The maximal schedules used for $\overrightarrow{u}$ are: $\{1, 4, 7\}$ twice, $\{3, 6\}$, $\{5, 8\}$ and $\{3, 8\}$. So $\overrightarrow{u} = \frac{1}{5}(2, 0, 2, 2, 1, 1, 2, 2, 0)$. The maximal schedules used for $\overrightarrow{v}$ are: $\{1, 5\}$, $\{1, 6\}$, $\{4, 8\}$ twice and $\{3, 7\}$ twice. So $\overrightarrow{v} = \frac{1}{6}(2, 0, 2, 2, 1, 1, 2, 2, 0)$. We have $\overrightarrow{v} = \frac{5}{6}\overrightarrow{u}$, thus $\sigma^* \leq \frac{5}{6}$. Note that this second example also shows that the bound provided by Lemmas 11 and 12 is tight for $k = 2$.

In Section 6.2 we will provide a lower bound on σ^* for networks with 9 nodes or more.

5.2 Properties of the Unstable Subsets of Links

We will prove here the propositions that we used to derive Theorem 1.

5.2.1 Minimum size of a maximum schedule

We first establish the following lemma:

Lemma 1 *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a network graph and $\mathbf{L}$ be a subset of links of $\mathbf{G}$. If any two maximal schedules of $\mathbf{L}$ sharing a common link have the same size, then $\mathbf{L}$ satisfies subset local pooling and hence cannot be unstable.*

Proof: For all $l \in \mathbf{L}$, let s_l be the size of the maximal schedules containing l, and take $\alpha_l = \frac{1}{s_l}$. Let $\mathcal{M}_{\mathbf{L}} = \{\overrightarrow{m_j}\}_j$ be the set of maximal schedules in $\mathbf{L}$, where $\overrightarrow{m_j}$ denotes a particular maximal schedule. Let $\overrightarrow{u} \in Co(\mathcal{M}_{\mathbf{L}})$, i.e., $\exists\{\mu_j\}_j$ such that $\overrightarrow{u} = \sum_j \mu_j \overrightarrow{m_j}$, $\sum_j \mu_j = 1$ and $\forall j,\ \mu_j \geq 0$. Then

$$\overrightarrow{\alpha}^T \overrightarrow{u} = \sum_j \mu_j \overrightarrow{\alpha}^T \overrightarrow{m_j}.$$

For all j, we have

$$\overrightarrow{\alpha}^T \overrightarrow{m_j} = \sum_{l \in \overrightarrow{m_j}} \frac{1}{s_l} = 1 \tag{5.1}$$

because, if $l \in \overrightarrow{m_j}$, then $\overrightarrow{m_j}$ has size s_l. So

$$\overrightarrow{\alpha}^T \overrightarrow{u} = \sum_j \mu_j = 1, \tag{5.2}$$

which is the definition of subset local pooling. ■

We now present some intuition behind the above proof. As GMS uses only maximal schedules, when a link l is scheduled for an amount of time u_l, some service is also provided to some of the other links in the network. The sum of the service provided to other links while scheduling l is proportional to u_l, because the size of the maximal schedules containing l is a constant. More precisely, let $\mathbf{L}_k$ be the set of links with maximal schedules of size k. Then, if $\mathbf{M}$ is a maximal schedule containing a link in $\mathbf{L}_k$, $\mathbf{M}$ is of size k. Therefore, $\forall l \in \mathbf{M}$, l belongs to $\mathbf{L}_k$ because all maximal schedules containing l must have the same size as $\mathbf{M}$. Thus $\sum_{l \in \mathbf{L}_k} u_l = k\tau_k(\overrightarrow{u})$, where $\tau_k(\overrightarrow{u})$ is the fraction of time we use a maximal schedule of size k. It means that the only way to provide more service to all the links in $\mathbf{L}_k$ is to spend more time $\tau_k(\overrightarrow{u})$ serving links of $\mathbf{L}_k$. But we cannot spend more time serving links in $\mathbf{L}_k$ for all k's at the same time, so one vector in $Co(\mathcal{M}_{\mathbf{L}})$ cannot strictly

dominate another. In other words, to have $\overrightarrow{u} > \overrightarrow{v}$, with $\overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}})$, we must have $u_l > v_l$ for all $l \in \mathbf{L}$. In particular, $\sum_{l \in \mathbf{L}_k} u_l > \sum_{l \in \mathbf{L}_k} v_l$, thus $\tau_k(\overrightarrow{u}) > \tau_k(\overrightarrow{v})$ for all k, which is not possible as $\sum_k \tau_k(\overrightarrow{u}) = \sum_k \tau_k(\overrightarrow{v}) = 1$.

We can now prove Proposition 1:

Proof: (Proposition 1) Let $\mathbf{U}$ be an unstable subset of links of $\mathbf{G}$. Suppose that the maximal schedules of $\mathbf{U}$ have size at most 2. We will simply check that all the maximal schedules of $\mathbf{U}$ that share a common link have the same size. Let $l \in \mathbf{U}$. Consider all the maximal schedules in $\mathcal{M}_{\mathbf{U}}$ that contain l. If one of them has size 2, which means that once l is scheduled, it is still possible to schedule another link, then all those maximal schedules must have size 2; otherwise, all the maximal schedules containing l are of size 1 (and actually $\{l\}$ is the only such maximal schedule). Lemma 1 implies that $\mathbf{U}$ satisfies subset local pooling, so it cannot be unstable, a contradiction. Thus there must exist a schedule of $\mathbf{U}$ of size 3. ■

5.2.2 Minimum number of disabled nodes

To prove Proposition 2, we will use the following lemma and its proof is included in the appendix.

Lemma 2 *We consider the 2-hop interference model. Let $\mathbf{U}$ be an unstable subset of links of $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. For any link l of $\mathbf{U}$, there are at least 2 nodes of $\mathbf{V}$ exactly 1-hop away from l.*

This implies that any maximal schedule of $\mathbf{U}$ disables at least 2 nodes and thus, using the same line of proof as for Theorem 1, it follows that GMS achieves full capacity in any network of 7 nodes or less under 2-hop interference. Note that, using Lemma 11, similar bounds can be obtained for the k-hop interference model with any $k > 2$. However, these bounds may

not be tight without extra work, and deriving them is beyond the scope of this thesis.

Proof: (Proposition 2) We consider the 2-hop interference model. Let $\mathbf{U}$ be an unstable subset of links of $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. We want to prove that any maximal schedule of $\mathbf{U}$ of size 2 or more disables at least 3 nodes of $\mathbf{V}$. It follows from Lemma 2 that at least 2 nodes are disabled. We will show that all the links of a maximal schedule of $\mathbf{U}$ of size 2 or more cannot disable the same 2 nodes.

By contradiction, suppose $\exists\{l_1, ...l_k\} \in \mathcal{M}_{\mathbf{U}}$, with $k \geq 2$, such that $\{l_1, ...l_k\}$ disables exactly 2 nodes. Denote by a and b these two nodes. Let $\mathbf{L_a}$ and $\mathbf{L_b}$ be the set of links of $\mathbf{U}$ containing a and b respectively. We will show $\mathbf{U} = \{l_1, ...l_k\} \cup \mathbf{L_a} \cup \mathbf{L_b}$. Indeed, suppose there exists another link $l' \in U$. One of its extreme nodes has to be either disabled or scheduled, otherwise l' could be scheduled, which would contradict the fact that $\{l_1, ...l_k\}$ is maximal. If one of the extreme nodes of l' is disabled, then it must be a or b and l' is in $\mathbf{L_a} \cup \mathbf{L_b}$. Thus, one of the extreme nodes of l' must be scheduled, so the other one cannot be scheduled because otherwise $\{l_1, ...l_k\}$ is not feasible. If that second node is not scheduled, then, by definition, it is disabled, which cannot be, as seen earlier. Thus there is no other link in $\mathbf{U}$.

Now for any $l \in \mathbf{U}$, we will show that the maximal schedules of $\mathbf{U}$ containing l have the same size. If $l \in \mathbf{L_a}$, then all the l_i's interfere with l and, either all the links in $\mathbf{L_b}$ interfere with l and then $\{l\}$ is the only maximal schedule containing l, or we can schedule exactly one additional link of $\mathbf{L_b}$ and then all maximal schedules containing l have size 2. The case $l \in \mathbf{L_b}$ is completely similar. If $l \in \{l_1, ...l_k\}$, then all the links in $\mathbf{L_a} \cup \mathbf{L_b}$ interfere with l, and thus $\{l_1, ...l_k\}$ is the only maximal schedule containing l. We have considered all the possible cases. Thus, using Lemma 1, $\mathbf{U}$ satisfies subset

local pooling, which cannot be. ■

5.3 Throughput Optimality for Node-Exclusive Models

Note that Proposition 1 does not require any assumption on the interference model. Thus it can be used to obtain performance bounds for interference models other than the 2-hop interference model. One such example is provided in the corollary below.

Corollary 1 *GMS is throughput optimal in all network graphs of* 5 *nodes or less for* any interference model *that allows a node to be part of only one scheduled link at any time (e.g.,* 1*-hop interference model).*

Proof: Proposition 1 directly implies that a graph with at least one unstable subset of links must have at least 6 nodes under such interference models. ■

In particular, this result confirms that GMS is throughput optimal in all graphs of 5 links or less under the 1-hop interference model, which has been obtained by exhaustive numerical search in [2]. Furthermore, we can find network graphs with 6 nodes (e.g., a ring network of size 6) where GMS is not throughput optimal.

CHAPTER 6

PERFORMANCE OF GMS UNDER K-HOP INTERFERENCE MODEL

6.1 Efficiency in Cycles

In this section, we will derive the exact value of the efficiency of GMS in ring networks for arbitrary number of nodes n and k-hop interference for arbitrary k. Let us denote by $\mathcal{C}$ the cycle of size n. We can label the links with the integers from 1 to n, such that consecutive links are labeled with consecutive integers. We also refer to maximal schedules of $\mathcal{C}$ as vectors of size n with 0's and 1's, where a 1 at position i means that the link labeled i is in the schedule and a 0 that it is not. We assume that the service rate on an active link is 1. $\mathcal{M}_{\mathcal{C}}$ is the set of all maximal schedules of $\mathcal{C}$, and $Co(\mathcal{M}_{\mathcal{C}})$ is its convex hull. The size of a schedule refers to the number of links it contains. $\|x\|_1 = \sum_{i=1}^{n} |x_i|$.

Lemma 3 *The maximum size of a maximal schedule of a cycle of size n is $\lfloor \frac{n}{k+1} \rfloor$. The minimal size is $\lceil \frac{n}{2k+1} \rceil$.*

Proof: Let $\overrightarrow{M}$ and $\overrightarrow{m}$ be maximal schedules of $\mathcal{C}$ of maximum and minimum size respectively. Because of the k-hop interference model, two consecutive active links of a maximal schedule must be separated by at least k and at most $2k$ inactive links, which already implies that $\|\overrightarrow{M}\|_1 \leq \frac{n}{k+1}$ and $\|\overrightarrow{m}\|_1 \geq \frac{n}{2k+1}$. But these numbers might not be integers, so in fact $\|\overrightarrow{M}\|_1 \leq \lfloor \frac{n}{k+1} \rfloor$ and $\|\overrightarrow{m}\|_1 \geq \lceil \frac{n}{2k+1} \rceil$. Moreover, we can easily construct

schedules that achieve these values. For example, consider the schedule such that the active links are the links labeled $l_0 = 1$, $l_1 = k+2$,...$l_L = L(k+1)+1$ with $L = \lfloor \frac{n}{k+1} \rfloor - 1$; the resulting schedule is of size $\lfloor \frac{n}{k+1} \rfloor$ and it is maximal because $k \leq n - l_L < 2k+1$. So $\|\overrightarrow{M}\|_1 = \lfloor \frac{n}{k+1} \rfloor$. Likewise, if the active links are the links labeled $s_0 = 1$, $s_1 = 2k+2$,...$s_{S-1} = (S-1)(2k+1)+1$ with $S = \lceil \frac{n}{2k+1} \rceil - 1$, the resulting schedule is of size $\lceil \frac{n}{2k+1} \rceil - 1$ and we have $2k+1 < n - s_{S-1} \leq 4k+2$. Thus we can add exactly one more link to that schedule, so that it will be maximal. So $\|\overrightarrow{m}\|_1 = \lceil \frac{n}{2k+1} \rceil$. ■

Proposition 3 *The efficiency of GMS in a cycle of size n is $\sigma^* = \frac{\lceil \frac{n}{2k+1} \rceil}{\lfloor \frac{n}{k+1} \rfloor}$.*

Proof: We will prove equality by showing that inequalities hold in both directions. As any strict subgraph of $\mathcal{C}$ is a tree, we know that GMS has an efficiency of 1 in those, so we need only worry about the whole cycle. Again, let $\overrightarrow{M}$ and $\overrightarrow{m}$ be maximal schedules of $\mathcal{C}$ of maximum and minimum size respectively. Also, let φ be the operator that "shifts" a schedule of one link, i.e., $\varphi : (i_1, ...i_n) \mapsto (i_n, i_1, ...i_{n-1})$. Then we have $\frac{\|\overrightarrow{M}\|_1}{n} \overrightarrow{1} = \frac{1}{n}(\sum_{i=0}^{n-1} \varphi^i(\overrightarrow{M}))$ and $\frac{\|\overrightarrow{m}\|_1}{n} \overrightarrow{1} = \frac{1}{n}(\sum_{i=0}^{n-1} \varphi^i(\overrightarrow{m}))$, where $\overrightarrow{1}$ is the all-1 vector of size n. Using the lemma, the first of those two vectors dominates the second one by a factor $\frac{\lceil \frac{n}{2k+1} \rceil}{\lfloor \frac{n}{k+1} \rfloor}$, so $\sigma^* \leq \frac{\lceil \frac{n}{2k+1} \rceil}{\lfloor \frac{n}{k+1} \rfloor}$. Moreover, $\forall \epsilon > 0$, there exist two vectors $\overrightarrow{\mu}$, $\overrightarrow{\nu}$ in $Co(\mathcal{M}_\mathcal{C})$ such that $(\sigma^* + \epsilon)\overrightarrow{\mu} \geq \overrightarrow{\nu}$ componentwise. But then $(\sigma^* + \epsilon)\|\overrightarrow{\mu}\|_1 \geq \|\overrightarrow{\nu}\|_1$. As $\overrightarrow{\mu}$ and $\overrightarrow{\nu}$ are convex combinations of maximal schedules of $\mathcal{C}$, we have $\|\overrightarrow{\mu}\|_1 \leq \|\overrightarrow{M}\|_1$ and $\|\overrightarrow{\nu}\|_1 \geq \|\overrightarrow{m}\|_1$. Thus $(\sigma^* + \epsilon) \geq \frac{\|\overrightarrow{m}\|_1}{\|\overrightarrow{M}\|_1} = \frac{\lceil \frac{n}{2k+1} \rceil}{\lfloor \frac{n}{k+1} \rfloor}$. Letting $\epsilon \to 0$ completes the proof. ■

6.2 Performance Bounds of GMS in Larger Networks under 2-Hop Interference

As we have seen in Section 5.1 that GMS may not achieve the full capacity in network graphs with more than 8 nodes, in this section we derive a lower bound on the efficiency ratio of GMS for larger networks.

Let $\|\overrightarrow{u}\|$ denote the l_1-norm of $\overrightarrow{u}$, i.e., $\|\overrightarrow{u}\| = \sum_i |u_i|$. In particular, if $\overrightarrow{M}$ is the rate vector associated with a schedule $\mathbf{M}$ of $\mathbf{E}$, then $\|\overrightarrow{M}\|$ is the number of links in $\mathbf{M}$. We will use $l \in \overrightarrow{u}$ to say that $u_l > 0$. We first prove the following lemma.

Lemma 4 *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a network graph, $\mathbf{L} \subseteq \mathbf{E}$ and*

$$\rho = \min_{l \in \mathbf{L}} \frac{\min\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_{\mathbf{L}} \text{ and } l \in \overrightarrow{u}\}}{\max\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_{\mathbf{L}} \text{ and } l \in \overrightarrow{u}\}}, \tag{6.1}$$

then $\mathbf{L}$ satisfies σ-local pooling for all $\sigma < \rho$.

Proof: Suppose that $\mathbf{L}$ does not satisfy σ-local pooling for some σ. Then, $\exists\, \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}})$ such that $\sigma\, \overrightarrow{u} \geq \overrightarrow{v}$ componentwise. We can write $\overrightarrow{u} = \sum_{l \in \mathbf{L}} u_l \overrightarrow{e_l}$, $\overrightarrow{v} = \sum_{l \in \mathbf{L}} v_l \overrightarrow{e_l}$, where $\overrightarrow{e_l}$ is the vector with a 1 at position l and 0's everywhere else. Writing $\mathcal{M}_{\mathbf{L}} = \{\overrightarrow{m_j}\}_j$, we also have $\overrightarrow{u} = \sum_j \mu_j \overrightarrow{m_j}$ and $\overrightarrow{v} = \sum_j \nu_j \overrightarrow{m_j}$, with $\sum_j \mu_j = \sum_j \nu_j = 1$ and $\forall j,\ \mu_j \geq 0, \nu_j \geq 0$.

For all $l \in \mathbf{L}$, let

$$\alpha_l = \min\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_{\mathbf{L}} \text{ and } l \in \overrightarrow{u}\}$$

and, similarly,

$$\beta_l = \max\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_{\mathbf{L}} \text{ and } l \in \overrightarrow{u}\}.$$

Then, for all l, we have $\frac{\alpha_l}{\beta_l} \geq \rho$.

Expanding $\sum_j \mu_j$ and $\sum_j \nu_j$ and using the notations introduced above, we obtain the following inequalities:

$$\begin{aligned} 1 &= \sum_j \mu_j = \sum_{l\in\mathbf{L}} \sum_{j:l\in\overrightarrow{m_j}} \frac{1}{\|\overrightarrow{m_j}\|}\mu_j \\ &\geq \sum_{l\in\mathbf{L}} \frac{1}{\beta_l} \sum_{j:l\in\overrightarrow{m_j}} \mu_j = \sum_{l\in\mathbf{L}} \frac{u_l}{\beta_l} \geq \rho \sum_{l\in\mathbf{L}} \frac{u_l}{\alpha_l} \end{aligned} \tag{6.2}$$

and

$$\begin{aligned} 1 &= \sum_j \nu_j = \sum_{l\in\mathbf{L}} \sum_{j:l\in\overrightarrow{m_j}} \frac{1}{\|\overrightarrow{m_j}\|}\nu_j \\ &\leq \sum_{l\in\mathbf{L}} \frac{1}{\alpha_l} \sum_{j:l\in\overrightarrow{m_j}} \nu_j = \sum_{l\in\mathbf{L}} \frac{v_l}{\alpha_l} \leq \sigma \sum_{l\in\mathbf{L}} \frac{u_l}{\alpha_l}. \end{aligned} \tag{6.3}$$

Combining Equations (6.2) and (6.3), we get $\sigma \geq \rho$, which means that $\mathbf{L}$ satisfies σ-local pooling for all $\sigma < \rho$. ■

Note that, if p is an upper bound of the size of the maximal schedules of $\mathbf{L}$ and $l \in \mathbf{L}$, then

$$\rho_l = \frac{\min\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_\mathbf{L} \text{ and } l \in \overrightarrow{u}\}}{\max\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_\mathbf{L} \text{ and } l \in \overrightarrow{u}\}} \geq \frac{2}{p}. \tag{6.4}$$

Indeed, if $\min\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_\mathbf{L} \text{ and } l \in \overrightarrow{u}\} = 1$, then we know that $\max\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_\mathbf{L} \text{ and } l \in \overrightarrow{u}\} = 1$ too, thus $\rho_l = 1$. And if $\min\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_\mathbf{L} \text{ and } l \in \overrightarrow{u}\} \geq 2$, then immediately $\rho_l \geq \frac{2}{p}$. As a consequence of Lemma 4 and the observation above, we can derive the following lower bound on the local pooling factor of a graph.

Theorem 2 *Under the 2-hop interference model, let* $\mathbf{G}$ *be a network graph of size* $n = 8 + 2k$ *or* $8 + 2k - 1$ *for some* $k \in \mathbb{N}^*$. *The local pooling factor* σ^* *of* $\mathbf{G}$ *is at least equal to* $\frac{2}{2+k}$.

Proof: We want to prove that any unstable subset of links $\mathbf{U}$ of $\mathbf{G}$ have

$$\rho = \min_{l \in \mathbf{U}} \frac{\min\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_{\mathbf{U}} \text{ and } l \in \overrightarrow{u}\}}{\max\{\|\overrightarrow{u}\| | \overrightarrow{u} \in \mathcal{M}_{\mathbf{U}} \text{ and } l \in \overrightarrow{u}\}} \geq \frac{2}{2+k}. \tag{6.5}$$

We need only consider unstable subsets of links of $\mathbf{G}$ because all other subsets of links satisfy σ-local pooling for all $\sigma < 1$. Then, using

$$\sigma^* = \inf\{\sigma | \exists \mathbf{L} \subseteq \mathbf{E},\ \exists\, \overrightarrow{u}, \overrightarrow{v} \in Co(\mathcal{M}_{\mathbf{L}}), \sigma\, \overrightarrow{u} \geq \overrightarrow{v}\},$$

it will follow that $\sigma^* \geq \frac{2}{2+k}$.

Let $\mathbf{U}$ be an unstable subset of links of $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Suppose that $|\mathbf{V}| = n \leq 8 + 2k$ for some $k \in \mathbb{N}$. Let p be the size of the maximal schedule of $\mathbf{U}$. It is enough to check that $p \leq 2 + k$. As we saw earlier, we must have $2p + 3 \leq n \leq 8 + 2k$, so $p \leq k + \frac{5}{2}$. As p and k are integers, it follows that $p \leq k + 2$. Then $\rho \geq \frac{2}{p} \geq \frac{2}{2+k}$, which completes the proof. ■

As an example, for $n \leq 20$, which is the case in many applications of wireless ad hoc and mesh networks, we get that $\sigma^* \geq \frac{1}{4}$. Recall that the best lower bound so far is $\sigma^* \geq \frac{1}{6}$ (it is derived in [3] and applies to geometric network graphs under the 2-hop interference model). Thus, the lower bound we provide here strictly improves on previous results for networks of up to 26 nodes. Table 6.1 shows the current best lower bound on the efficiency ratio γ^* of GMS, with 2-hop interference, as a function of the number of nodes in the network.

6.3 Performance Bounds for k-Hop Interference Model

This section is an extension of the results obtained for the 2-hop interference model in the previous section to the k-hop interference model for arbitrary k.

Table 6.1: Efficiency of GMS in Small Networks (2-Hop Interference Model)

Number of Nodes	Lower Bound on γ^*
≤ 8	1
9-10	2/3
11-12	1/2
13-14	2/5
15-16	1/3
17-18	2/7
19-20	1/4
21-22	2/9
23-24	1/5
25-26	2/11
27-28	1/6
≥ 29	1/6 in geometric graphs [3]

However, note that the extra work that made the result tight for the 2-hop model has not been done for the k-hop model.

We can observe that, under the k-hop interference model, a node can be within $\lceil \frac{k}{2} \rceil - 1$ hops of only one active link at a time. Indeed, otherwise, the distance between the two active links would be at most $2(\lceil \frac{k}{2} \rceil - 1) \leq k - 1$. We will say that the nodes within $\lceil \frac{k}{2} \rceil - 1$ hops of an active link are disabled by that link. It is then clear that any active link disables at least $\lceil \frac{k}{2} \rceil - 1$ nodes.

Proposition 4 *Under the k-hop interference model, let $\mathbf{G}$ be a network graph of size $n \leq (1 + \lceil \frac{k}{2} \rceil)(3 + m) - 1$ for some $m \in \mathbb{N}$. The local pooling factor σ^* of $\mathbf{G}$ is at least equal to $\frac{2}{2+m}$.*

The results for the efficiency in cycles show that the smallest cycle in which GMS is not throughput optimal under the k-hop interference model is of size $3(k + 1)$. The proposition above shows that at least an order of $3(\frac{k}{2} + 1)$ is required for GMS to fail. Hence, there is at most a size factor of 2 between the smallest graph for which GMS is not throughput optimal under the k-

hop interference model and the smallest cycle for which we know GMS is not throughput optimal for that same interference model.

CHAPTER 7

DECENTRALIZATION OF GMS

If the links have the *perfect carrier sensing* capability (once a link begins to transmit, all its interfering links can sense the transmission with zero delay), then there exists a simple scheme to implement GMS in a distributed manner. We can divide each time slot into a scheduling period followed by a transmitting period. At the beginning of a time slot, every link l with a positive link weight (queue length) w_l will select a (deterministic) backoff time T_l, with T_l being a decreasing function of w_l (e.g., $T_l = \frac{T}{w_l}$), and will begin to transmit data after time T_l if it senses that none of its interfering links are transmitting. Without loss of generality, we can assume that the link weights are distinct (each link can add a random number uniformly selected in $[0, 1]$ to its weight so that, with probability one, the link weights will be distinct). It is easy to see that the generated schedule after the scheduling period is a schedule returned by GMS. If the length of the scheduling period is much less than the length of the transmission period, then the scheduling overhead is negligible.

However, in practice, since we cannot achieve perfect carrier sensing, collisions are possible which will degrade the throughput performance of the above scheme. We want to find a distributed implementation of GMS that does not require such strong assumptions. To this end, we show in this section that GMS is equivalent to the Local Greedy Maximal Scheduling (LGMS) algorithm which uses only local neighborhood information.

LGMS was first proposed by Preis [4] as a low-complexity approximation algorithm for the maximum weighted matching problem (corresponding to the 1-hop interference model). For a general scheduling problem, LGMS computes a (maximal) schedule as follows. In each step, it adds a *locally heaviest link* (a link with the largest weight compared to its non-disabled interfering links) to the schedule; it then disables all the interfering links of the added link. The process is repeated until the schedule is maximal.

Note that LGMS is different from the Local Greedy Scheduling (LGS) algorithm presented in [15] which schedules only those links that are locally heaviest in the beginning and does not proceed iteratively to produce a maximal schedule. However, the throughput analysis involving local greedy scheduling in [15] is incomplete, so it is worthwhile to study LGMS as an alternative.

Local Greedy Maximal Scheduling (LGMS) Algorithm

LGMS($\mathbf{E}$) := $\emptyset$

$\mathbf{E}' := \mathbf{E}$

WHILE ($\mathbf{E}' \neq \emptyset$)

Pick a locally heaviest link l:

$$w(l) \geq w(l'), \forall l' \in \mathbf{I}(l) \cap \mathbf{E}'.$$

LGMS($\mathbf{E}$) := LGMS($\mathbf{E}$) $\cup \{l\}$

$\mathbf{E}' := \mathbf{E}' \setminus \{l\} \setminus \mathbf{I}(l)$

ENDWHILE

The key difference between LGMS and GMS is that, in each step, LGMS picks a locally heaviest link instead of a globally heaviest link. Since LGMS only requires local neighborhood information, it is amenable to distributed implementation. A distributed implementation of LGMS was proposed for the weighted matching problem in [10].

For a general scheduling problem, we design the following distributed implementation of LGMS. We assume that there exists a collision-free mechanism[1] for every link to send control messages to its interfering links. Each time slot is divided into a scheduling period and a transmission period. At the beginning of the scheduling period, each link will send its weight (queue length) to its interfering links. Then every link will determine whether it will be scheduled or not (to transmit data in the transmission period) according to the following protocol (assume distinct link weights).

Distributed Protocol to Implement LGMS (At Link l)

$m_l := 0$

$\mathcal{N} := \mathbf{I}(l)$

WHILE $(m_l = 0)$

If $w(l) > w(l'), \forall l' \in \mathcal{N}$:

$m_l := 1$, send $m_l = 1$ to all links in $\mathcal{N}$

If received $m_{l'} = 1$ from $l' \in \mathcal{N}$:

$m_l := -1$, send $m_l = -1$ to all links in $\mathcal{N}$

If received $m_{l'} = -1$ from $l' \in \mathcal{N}$:

[1]For example, we can color the links so that no two links interfering with each other have the same color, and then the links can send their control messages in a round-robin fashion based on their colors.

$\mathcal{N} := \mathcal{N} \setminus \{l'\}$

ENDWHILE

We can verify that after the execution of the above distributed protocol, every link l will have either $m_l = 1$ (which means that l is included in the schedule) or $m_l = -1$ (which means that l is disabled because one of its interfering links is scheduled). During the transmission period, any link l with $m_l = 1$ can transmit data. While the computational complexity of LGMS is low, the signaling and time overhead can scale with the size of the network in the worst case (depending upon the topology). However, in small to moderately sized networks, the overhead can be expected to be quite small.

In a network with arbitrary link weights, it is possible that two interfering links l_1 and l_2 have same weight. Then, those two links could stay indefinitely with $m_{l_1} = m_{l_2} = 0$ and $w(l_1) \geq w(l'), \forall l' \in \mathcal{N}_{l_1} \cup \mathcal{N}_{l_2}$. To prevent this from happening, we assume that there is a tie-breaking mechanism that will allow one of the links with equal weight to activate in such cases.

At first glance, one may think that LGMS and GMS are different algorithms and could return different schedules because LGMS proceeds even more greedily than GMS. For the weighted matching problem, Preis [4] showed that, like GMS, LGMS returns a matching with a weight of at least $\frac{1}{2}$ of the weight of a maximum weighted matching. However, a stronger result can be obtained: we will show that the sets of schedules produced by GMS and LGMS are identical, and this result holds for networks with arbitrary interference graphs. While this fact may appear intuitive, it does not seem to have been recognized previously.

7.1 On the Equivalence of LGMS and GMS

7.1.1 Networks with distinct link weights

Theorem 3 *For networks with distinct link weights, LGMS and GMS produce the same schedule.*

Let GMS($\mathbf{L}$) and LGMS($\mathbf{L}$) be the schedule produced by GMS and LGMS, respectively, when applied to a set of links $\mathbf{L} \subseteq \mathbf{E}$. We first prove two lemmas. Lemma 5 says that if we know that a certain link l is included in GMS($\mathbf{E}$), then an alterative way to generate GMS($\mathbf{E}$) is to add l first and then apply GMS on the remaining links that do not interfere with l.

Lemma 5 $\forall l \in GMS(\mathbf{E})$, $GMS(\mathbf{E}) = \{l\} \cup GMS(\mathbf{E} \setminus \{l \cup \mathbf{I}(l)\})$.

Proof: GMS is a sequential algorithm. Let $l_1, l_2, ..., l_m$ be the sequence of links selected by GMS when applied on $\mathbf{E}$. GMS($\mathbf{E}$) $= \{l_1, l_2, ..., l_m\}$[2] and $w(l_1) > w(l_2) > \cdots > w(l_m)$.

For any l_i in the schedule produced by GMS, consider a two-step procedure to produce a maximal schedule as follows: first add l_i to the schedule and disable its interfering links $\mathbf{I}(l_i)$; then apply GMS on the remaining links $\mathbf{E} \setminus \{l_i \cup \mathbf{I}(l_i)\}$. Because $l_1, ..., l_{i-1}, l_{i+1}, ..., l_m$ do not interfere with link l_i (otherwise they cannot form a schedule), we know that $l_1, ..., l_{i-1}, l_{i+1}, ..., l_m$ will be in $\mathbf{E} \setminus \{l_i \cup \mathbf{I}(l_i)\}$, i.e., selecting l_i in the first step will not affect the eligibility of other l_j's in the second step. Therefore, since l_1 has the largest weight among the links in $\mathbf{E} \setminus \{l_i \cup \mathbf{I}(l_i)\}$, GMS will select l_1 and disable its interfering links. Similarly, GMS will select l_2 next, and so on. So, this two-step procedure will produce exactly the same schedule as $GMS(\mathbf{E})$, i.e., $\{l_i\} \cup \text{GMS}(\mathbf{E} \setminus \{l_i \cup \mathbf{I}(l_i)\}) = \{l_1, l_2, ..., l_m\} = \text{GMS}(\mathbf{E})$. ■

[2]It is easy to verify that, for networks with distinct link weights, GMS produces a unique schedule.

Lemma 6 *Let l be the first link selected by LGMS. Link l must also be selected by GMS.*

Proof: Link l is a locally heaviest link in $\mathbf{E}$; i.e., its weight is greater than the weight of the links in $\mathbf{I}(l)$. Suppose l is not selected by GMS: then at least one link in $\mathbf{I}(l)$ will be selected by GMS; otherwise, the schedule produced by GMS is not maximal. Let l' be the first link in $\mathbf{I}(l)$ selected by GMS. At the time instant when GMS decides to select l', l is also eligible and has a greater weight than l', a contradiction. Therefore, l must be selected by GMS. ■

From the two lemmas we can prove Theorem 3.

Proof: (Theorem 3) We prove the theorem by induction on the cardinality of GMS($\mathbf{E}$) (the number of links contained in GMS($\mathbf{E}$)).

Step 1. The theorem is clearly true for all $\mathbf{E}_1$ such that $|\text{GMS}(\mathbf{E}_1)| = 1$, because in this case the globally heaviest link coincides with the (unique) locally heaviest link which will be the only link selected by both GMS and LGMS.

Step 2. Assume the theorem is true for all $\mathbf{E}_m$ with $|\text{GMS}(\mathbf{E}_m)| = m$. Now consider any $\mathbf{E}_{m+1}$ with $|\text{GMS}(\mathbf{E}_{m+1})| = m+1$. Let l be the first link selected by LGMS. By Lemma 6, $l \in \text{GMS}(\mathbf{E}_{m+1})$. Then by Lemma 5, $\text{GMS}(\mathbf{E}_{m+1}) = \{l\} \cup \text{GMS}(\mathbf{E}_{m+1} \setminus \{l \cup \mathbf{I}(l)\})$. Since $|\text{GMS}(\mathbf{E}_{m+1} \setminus \{l \cup \mathbf{I}(l)\})| = m$, we have $\text{LGMS}(\mathbf{E}_{m+1} \setminus \{l \cup \mathbf{I}(l)\}) = \text{GMS}(\mathbf{E}_{m+1} \setminus \{l \cup \mathbf{I}(l)\})$ by induction assumption. Therefore,

$$\begin{aligned} \text{LGMS}(\mathbf{E}_{m+1}) &= \{l\} \cup \text{LGMS}(\mathbf{E}_{m+1} \setminus \{l \cup \mathbf{I}(l)\}) \\ &= \{l\} \cup \text{GMS}(\mathbf{E}_{m+1} \setminus \{l \cup \mathbf{I}(l)\}) \\ &= \text{GMS}(\mathbf{E}_{m+1}). \end{aligned}$$

By induction argument, we show that LGMS($\mathbf{E}$) = GMS($\mathbf{E}$) holds for any $\mathbf{E}$. ■

7.1.2 An Alternate Proof

We will use Lemma 6 and give an alternate proof which may be more intuitive. Suppose we start with a network graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and we run separately GMS and LGMS and obtain the sets of links GMS($\mathbf{E}$) and LGMS($\mathbf{E}$) respectively. For ease of notation, in this proof we will refer to GMS($\mathbf{E}$) as $\mathbf{L}$ and to LGMS($\mathbf{E}$) as $\widetilde{\mathbf{L}}$. The proof relies on the following two simple lemmas:

Lemma 7 *If there exists $l \in \mathbf{L} \setminus \widetilde{\mathbf{L}}$, then there also exists $\widetilde{l} \in \widetilde{\mathbf{L}} \setminus \mathbf{L}$ such that $\widetilde{l} \in \mathbf{I}(l)$ and $w(\widetilde{l}) \geq w(l)$.*

Proof: l must be disabled ($m_l = -1$) at the end of the execution of LGMS. However, none of the links in $\widetilde{\mathbf{L}} \cap \mathbf{I}(l)$ with weights strictly smaller than $w(l)$ can be scheduled by LGMS before l is disabled. Hence, l must be disabled because one of its interfering links of greater weight has been scheduled. ■

Lemma 8 *If there exists $\widetilde{l} \in \widetilde{\mathbf{L}} \setminus \mathbf{L}$, then there also exists $l' \in \mathbf{L} \setminus \widetilde{\mathbf{L}}$ such that $l \in \mathbf{I}(\widetilde{l})$ and $w(l') \geq w(\widetilde{l})$.*

Proof: As GMS produces a maximal schedule, there must be a first link scheduled among the interferers of $\widetilde{l}$ during the execution of GMS. Call that link l' and consider the step of the execution of GMS at which it adds l' to the schedule. At that step, $\widetilde{l}$ is still available, but GMS chooses to schedule l' instead, hence $w(l') \geq w(\widetilde{l})$. ■

We can then prove Theorem 3 for networks with distinct link weights using the two lemmas above and Lemma 6.

Proof: *(Theorem 3)* Combining the two lemmas 7 and 8, we get that, if there exists a link $l \in \mathbf{L} \setminus \widetilde{\mathbf{L}}$, there must exist another link $l' \in \mathbf{L} \setminus \widetilde{\mathbf{L}}$ with $w(l') > w(l)$. The inequality is strict here because the inequalities in both lemmas are strict when the link weights are distinct. This is clearly impossible as GMS schedules links in order of decreasing weight and there must be a first link scheduled by GMS that is not scheduled by LGMS. We conclude that $\mathbf{L} \subseteq \widetilde{\mathbf{L}}$, which in turn implies equality because $\widetilde{\mathbf{L}}$ and $\mathbf{L}$ are both maximal schedules. ■

7.1.3 Networks with general link weights

In the previous subsection we assumed that the link weights are distinct. We can generalize the result to networks where two or more links may have the same weight. One approach to handle this is to introduce a *deterministic* tie-breaking mechanism. For example, we can associate each link with a distinct label.[3] For the candidate links (globally or locally heaviest links) with the same weight, preference is given to the link with the smallest (or largest) label. If both GMS and LGMS use the same deterministic tie-breaking mechanism, then we can show that they will again produce the same schedule.

Formally, let π denote a deterministic tie-breaking mechanism (e.g., the labelling mechanism described above). Let $\text{GMS}_\pi(\mathbf{E})$ and $\text{LGMS}_\pi(\mathbf{E})$ denote the schedule produced by GMS and LGMS on $\mathbf{E}$ using the tie-breaking mechanism π, respectively. We can show:

Proposition 5 For networks with general link weights and under the tie-breaking mechanism π, $\text{LGMS}_\pi(\mathbf{E}) = \text{GMS}_\pi(\mathbf{E})$.

The proof is similar to the proof of Proposition 3.

[3]Each link can generate a random real number uniformly in [0,1] as its label. With probability one these labels are distinct.

On the other hand, if GMS/LGMS apply a *randomized* mechanism to break the tie, i.e., they randomly select a link among the candidate links with the same weight, then GMS/LGMS may produce different schedules. Nevertheless, we can show:

Theorem 4 For networks with general link weights and under a randomized tie-breaking mechanism, LGMS and GMS produce the same set of possible schedules.

Let GMS($\mathbf{E}$) and LGMS($\mathbf{E}$) be the set of possible schedules produced by GMS and LGMS on $\mathbf{E}$, respectively. The proof of Theorem 4 consists of the following two lemmas.

Lemma 9 *GMS*($\mathbf{E}$) $\subseteq$ *LGMS*($\mathbf{E}$).

Proof: Let $\mathbf{M} \in$ GMS($\mathbf{E}$). Suppose $l_1, l_2, ..., l_m$ is the sequence of links selected by GMS when generating $\mathbf{M}$. In step 1 of LGMS, l_1 is a globally heaviest link because it can be scheduled by GMS, thus it is also a locally heaviest link and could be scheduled by LGMS. Similarly, in step i of LGMS, given $l_1, l_2, ..., l_{i-1}$ have been scheduled, l_i is a globally heaviest link among the remaining eligible links because it can be scheduled by GMS; thus it is also a locally heaviest link and could be selected by LGMS. Then, by induction, LGMS could also select $l_1, l_2, ..., l_m$ in sequence under a randomized tie-breaking mechanism, so $\mathbf{M} \in$ LGMS($\mathbf{E}$). ■

Lemma 10 *LGMS*($\mathbf{E}$) $\subseteq$ *GMS*($\mathbf{E}$).

Proof: Let $\mathbf{M} \in$ LGMS($\mathbf{E}$). Suppose $l_1, l_2, ..., l_m$ is the sequence of links selected by LGMS when generating $\mathbf{M}$. We label the links in $\mathbf{E}$ as follows. Link l_1 is labelled 1, and the interfering links of l_1 in $\mathbf{I}(l_1)$ are labelled 2, 3, ...,

m_1 (suppose $|l_1 \cup \mathbf{I}(l_1)| = m_1$); l_2 is labelled m_1+1, and the interfering links of l_2 in $\mathbf{I}(l_2) \backslash \mathbf{I}(l_1)$ are labelled $m_1+2, ..., m_2$ (suppose $|l_1 \cup \mathbf{I}(l_1) \cup l_2 \cup \mathbf{I}(l_2)| = m_2$); and so on. Let π denote the deterministic tie-breaking mechanism using those link labels; i.e., when a tie occurs, preference is given to the link with the smallest label. Let $\mathrm{GMS}_\pi(\mathbf{E})$ and $\mathrm{LGMS}_\pi(\mathbf{E})$ denote the schedule produced by GMS and LGMS on $\mathbf{E}$ under the tie-breaking mechanism π, respectively.

Based on Proposition 5 we know $\mathbf{M} = \mathrm{LGMS}_\pi(\mathbf{E}) = \mathrm{GMS}_\pi(\mathbf{E})$. Note that in every step of GMS, the selection made by π is one of the possible choices of the randomized tie-breaking rule; thus $\mathrm{GMS}_\pi(\mathbf{E})$ can be selected by GMS under the randomized tie-breaking mechanism, hence $\mathbf{M} \in \mathrm{GMS}(\mathbf{E})$. ■

CHAPTER 8

CONCLUSION

GMS is a low-complexity scheduling algorithm which has been observed to achieve near-optimal throughput and delay performance in a variety of wireless network simulations. However, theoretical bounds to date on the performance of GMS only show that it can achieve a fraction of the capacity region. In this thesis, we focused on networks of small to moderate size, which is the case in many practical wireless ad hoc and mesh networks. We established the throughput optimality of GMS in networks with up to 8 nodes under the 2-hop interference model. We also provided a lower bound on its throughput efficiency in larger networks which improves previous bounds. Furthermore, we showed that GMS is equivalent to LGMS, which is amenable to distributed implementation. This means a simple, distributed scheduling algorithm like GMS/LGMS is suitable for many applications in wireless networks.

APPENDIX A

PROOF OF LEMMA 2

In order to prove Lemma 2, we need to further characterize unstable subsets of links. Indeed, in a general network graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ a scheduled link may disable only one node or even no node at all. But not any link can be unstable, i.e., belong to an unstable subset of links. For example, [3] and [8] proved that tree networks cannot be unstable under the k-hop interference model, which means that isolated links cannot be unstable and, thus, a scheduled unstable link must disable at least one node. To be able to prove that a scheduled unstable link disables at least 2 nodes, we will need to determine additional properties of unstable subsets of links. More precisely, we will prove the following lemma:

Lemma 11 *Under the k-hop interference model, the maximum depth of an unstable link in an open tree of cliques is $\lfloor \frac{k}{2} \rfloor$.*

We will now define all the notions required. In order to describe what an open tree of cliques is, we will first explain the notions of tree of cliques and open tree. Recall that a clique is a complete graph, which means that it has an edge between any two vertices. Note that we can define a tree as a connected graph without cycles. We use a similar definition for trees of cliques:

Definition 6 *A tree of cliques is a connected graph $\mathbf{T}$ such that, for any cycle in $\mathbf{T}$, there is a link between any two nodes of that cycle.*

A tree of cliques can be viewed as a tree whose nodes can also be replaced by cliques. In particular, a tree is a tree of cliques.

Definition 7 *An open tree of a network graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *is a subgraph* $\mathbf{T} = (\mathbf{V_T}, \mathbf{E_T})$ *of* $\mathbf{G}$ *that is a tree and such that only one node* r *in* $\mathbf{V_T}$ *is linked to nodes in* $\mathbf{V} \setminus \mathbf{V_T}$. *We say that* r *is the root of* $\mathbf{T}$.

In other words, an open tree is a tree attached by its root to a graph. In particular, a tree is an open tree. The definition of an open tree of cliques follows naturally:

Definition 8 *An open tree of cliques of a network graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *is a subgraph* $\mathbf{T} = (\mathbf{V_T}, \mathbf{E_T})$ *of* $\mathbf{G}$ *that is a tree of cliques and such that only one node* r *in* $\mathbf{V_T}$ *is linked to nodes in* $\mathbf{V} \setminus \mathbf{V_T}$. *We say that* r *is the root of* $\mathbf{T}$.

As mentioned earlier, we are interested in the depth of links in such structures. To define depth, we will use the following function: in a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, the "distance" $\mathrm{d}(a, b)$ between a and b is the minimum number of links of $\mathbf{E}$ needed to connect a to b. Note that this can apply as well to two nodes, two links, or a node and a link.

Definition 9 *Let* $\mathbf{T} = (\mathbf{V_T}, \mathbf{E_T})$ *be an open tree of cliques of root* r. *The depth of a link* $l \in \mathbf{E_T}$ *is* $\delta_l = 1 + \mathrm{d}(l, r)$.

In other words, if the shortest path in $\mathbf{E_T}$ between l and r is of length $t = \mathrm{d}(l, r)$, then we say that l is of depth $t + 1$. Note that if l is the link connecting nodes a and b, then $\mathrm{d}(l, r) = \min\{\mathrm{d}(a, r),\ \mathrm{d}(b, r)\} + 1$.

We will first state and prove a weaker version of Lemma 11 because this proof should convey the main intuition. Then, proving Lemma 11 is only technical details.

Lemma 12 *Under the k-hop interference model, the maximum depth of an unstable link in an open tree is* $\lfloor \frac{k}{2} \rfloor$.

Proof: Let $\mathbf{U}$ be an unstable subset of links of $\mathbf{G}$. There exists a feasible arrival rate $\overrightarrow{\lambda}$ in the interior of $Co(\mathcal{M}_{\mathbf{U}})$ such that the queues of all links in $\mathbf{U}$ increase to infinity. Let $\mathbf{T} = (\mathbf{V_T}, \mathbf{E_T})$ be an open tree of $\mathbf{G}$ and let $\mathbf{U_T} = \mathbf{U} \cap \mathbf{E_T}$. Let l^* be a link in $\mathbf{U_T}$ of maximum depth δ with respect to $\mathbf{T}$. Assume, towards a contradiction, that $\delta > \lfloor \frac{k}{2} \rfloor$, so in fact $\delta \geq \lfloor \frac{k}{2} \rfloor + 1$. Consider the set of all interfering links of l^* in $\mathbf{U}$, and call it $\mathbf{I_U}(l^*) = \mathbf{I}(l^*) \cap \mathbf{U}$. We want to prove that the queues of all the links in $\mathbf{I_U}(l^*)$ cannot increase to infinity. By definition of $\mathbf{I_U}(l^*)$, it is clear that, at every time instant, at least one link of $\mathbf{I_U}(l^*)$ is scheduled. Indeed, all the links that are not in $\mathbf{U}$ have no packet to transmit; and since all the links of $\mathbf{U}$ that interfere with l are in $\mathbf{I_U}(l^*)$, if none of them is scheduled, then l^* should be scheduled. Furthermore, we will check that any two links a and b of $\mathbf{I_U}(l^*)$ interfere with each other. If $a, b \notin \mathbf{U_T}$, then we have $\mathrm{d}(a, r) = \mathrm{d}(a, l^*) - \mathrm{d}(l^*, r) \leq (k-1) - (\delta - 1) \leq k - 1 - \lfloor \frac{k}{2} \rfloor = \lceil \frac{k}{2} \rceil - 1$ and, similarly, $\mathrm{d}(b, r) \leq \lceil \frac{k}{2} \rceil - 1$. Thus, $\mathrm{d}(a, b) \leq \mathrm{d}(a, r) + \mathrm{d}(b, r) \leq 2\lceil \frac{k}{2} \rceil - 2 \leq k - 1$ and a and b interfere with each other. If $a \in \mathbf{U_T}$ and $b \notin \mathbf{U_T}$, then $\mathrm{d}(a, b) = \mathrm{d}(a, r) + \mathrm{d}(r, b) \leq \mathrm{d}(l^*, r) + \mathrm{d}(r, b) = \mathrm{d}(l^*, b)$ and again a and b interfere with each other. Finally, if $a, b \in \mathbf{U_T}$, the proof in [3] applies and we get that a and b interfere with each other. So any two links of $\mathbf{I_U}(l^*)$ interfere with each other, and thus it is not possible to schedule more than one link in $\mathbf{I_U}(l^*)$ at every time instant. Since $\overrightarrow{\lambda}$ is in the interior of the capacity region, $\sum_{l \in \mathbf{I_U}(l^*)} \lambda_l < 1$. As one link of $\mathbf{I_U}(l^*)$ is scheduled at every time instant, it is not possible that all the queues of the links of $\mathbf{I_U}(l^*)$ increase to infinity, so $\delta > \lfloor \frac{k}{2} \rfloor$ cannot be true. ■

We will now prove Lemma 11.

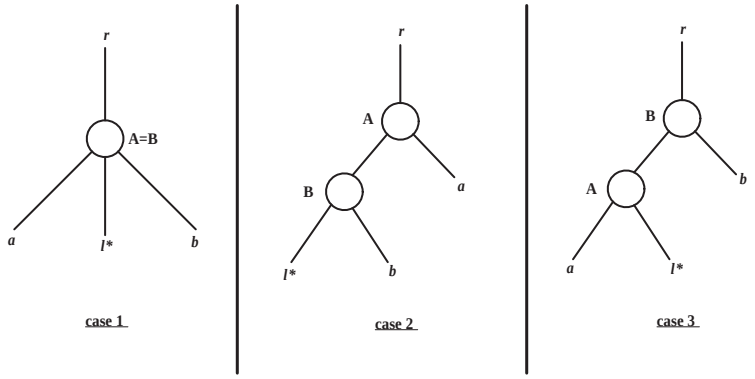


Figure A.1: Tree of cliques - the 3 possible cases.

Proof: (Lemma 11) The proof is identical to that of Lemma 12 except when we want to prove that two links $a, b \in \mathbf{I_U}(l^*) \cap \mathbf{U_T}$ interfere with each other. To that end, define a nearest common ancestor of a and b as a node with smallest depth in $\mathbf{T}$ that belongs to a shortest path in $\mathbf{E_T}$ between a and b. As $\mathbf{T}$ is a tree of cliques, a and b may have many nearest common ancestors, but they will all belong to the same clique C. In that case, we say that C is the nearest common ancestor of a and b in $\mathbf{T}$. Let A be the nearest common ancestor of a and l^* in $\mathbf{T}$ and B be that of b and l^*. We will consider all possible cases for the relative locations of A and B. Those cases are represented in Figure A.1. The case $A = B$ corresponds to case 1. If $A \neq B$, then either A is an ancestor of B, or B is an ancestor of A. These last 2 cases are symmetrical, so we will consider only one of them.

For any $c \in \mathbf{T}$, denote by A_c the node of A that is closest to c, i.e., A_c is such that $\mathrm{d}(c, A_c) = \min_{n \in A} \mathrm{d}(c, n)$. Note that this definition is valid whether c is a node or a link.

We consider case 3 first because it is simpler and case 1 does not bring any new idea. $\mathrm{d}(a, b) = \mathrm{d}(l^*, b) + \mathrm{d}(a, A_b) - \mathrm{d}(l^*, A_b) = \mathrm{d}(l^*, b) + \mathrm{d}(a, r) - \mathrm{d}(l^*, r)$ and $\mathrm{d}(l^*, r) \geq \mathrm{d}(a, r)$ because l^* is of maximum depth in $\mathbf{T}$. So $\mathrm{d}(a, b) \leq \mathrm{d}(l^*, b)$ and thus a and b interfere with each other.

In case 1, $\mathrm{d}(a,b) = \mathrm{d}(l^*,b) + \mathrm{d}(a,A_a) + \mathrm{d}(A_a,A_b) - \mathrm{d}(l^*,A_{l^*}) - \mathrm{d}(A_{l^*},A_b) = \mathrm{d}(l^*,b) + \mathrm{d}(a,r) - \mathrm{d}(r,A_r) - \mathrm{d}(A_r,A_a) + \mathrm{d}(A_a,A_b) - \mathrm{d}(l^*,r) + \mathrm{d}(r,A_r) + \mathrm{d}(A_r,A_{l^*}) - \mathrm{d}(A_{l^*},A_b) \leq \mathrm{d}(l^*,b) + \mathrm{d}(A_a,A_b) + \mathrm{d}(A_r,A_{l^*}) - \mathrm{d}(A_r,A_a) - \mathrm{d}(A_{l^*},A_b)$ because $\mathrm{d}(a,r) \leq \mathrm{d}(l^*,r)$. We can switch the roles of a and b and get another expression. So, we have the following two inequalities:

$$\begin{aligned} \mathrm{d}(a,b) &\leq \mathrm{d}(l^*,b) + \mathrm{d}(A_a,A_b) + \mathrm{d}(A_r,A_{l^*}) \\ &\quad -\mathrm{d}(A_r,A_a) - \mathrm{d}(A_{l^*},A_b) \qquad \text{(A.1)} \\ \mathrm{d}(a,b) &\leq \mathrm{d}(l^*,a) + \mathrm{d}(A_a,A_b) + \mathrm{d}(A_r,A_{l^*}) \\ &\quad -\mathrm{d}(A_r,A_b) - \mathrm{d}(A_{l^*},A_a) \qquad \text{(A.2)} \end{aligned}$$

We now check that in any case we can use one of the previous equations to prove that a and b interfere with each other.

If $A_r = A_a$, then $\mathrm{d}(A_a,A_b) = \mathrm{d}(A_r,A_b)$ and $\mathrm{d}(A_r,A_{l^*}) = \mathrm{d}(A_{l^*},A_a)$ so we can conclude using (A.2).

If $A_{l^*} = A_b$, then $\mathrm{d}(A_a,A_b) = \mathrm{d}(A_{l^*},A_a)$ and $\mathrm{d}(A_r,A_{l^*}) = \mathrm{d}(A_r,A_b)$ so, again, we can conclude using (A.2).

If $A_r \neq A_a$ and $A_{l^*} \neq A_b$, then we use (A.1). ■

We are now ready to prove Lemma 2.

Proof: (Lemma 2) We consider the 2-hop interference model. Let $\mathbf{U}$ be an unstable subset of links of $\mathbf{G}$ and l be a link of $\mathbf{U}$. We want to show that there are at least 2 nodes of $\mathbf{V}$ exactly 1-hop away from l. We will consider all possible cases. If l is an open link, then Lemma 12 implies that one of the extreme nodes of l is in a cycle of $\mathbf{G}$, so its two neighboring nodes in that cycle are disabled. If l is in a cycle $\mathbf{C}$ of size 4 or more, each extreme node of l disables a neighboring node in $\mathbf{C}$. If l is in a cycle $\mathbf{C}$ of size 3, then, firstly, it disables the third node of $\mathbf{C}$. But since a cycle of size 3 is

a clique, Lemma 11 implies that at least one of the extreme nodes of l will disable a node in the rest of $\mathbf{G}$. Finally, if l is not an open link and not in a cycle, then there are two subgraphs of $\mathbf{G}$, both with more than one node, that are connected only through l. Link l will disable one node in both those subgraphs. ■

REFERENCES

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximal throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[2] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks - A partitioning approach," in *Proceedings of ACM MOBICOM'06*, September 2006, pp. 26–37.

[3] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," in *Proceedings of IEEE INFOCOM'08*, April 2008, pp. 1132–1145.

[4] R. Preis, "Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs," in *Proceedings of STACS'99*, 1999, pp. 259–269.

[5] N. McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*. Berkeley, CA: University of California, Berkeley, 1995.

[6] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer control in multihop wireless networks," in *Proceedings of IEEE INFOCOM'05*, 2005, pp. 302–315.

[7] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Advances in Applied Probability*, vol. 38, no. 2, pp. 505–521, 2006.

[8] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," in *Proceedings of IEEE INFOCOM'08*, April 2008, pp. 1139–1147.

[9] C. Joo, X. Lin, and N. B. Shroff, "Performance limits of greedy maximal matching in multi-hop wireless networks," in *Proceedings of IEEE CDC'07*, December 2007, pp. 1128–1133.

[10] J.-H. Hoepman, "Simple distributed weighted matchings," October 2004. [Online]. Available: http://arxiv.org/abs/cs.DC/0410047.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, 1979.

[12] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910–917, 1988.

[13] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in *Proceedings 43rd Annual Allerton Conference on Communication, Control and Computing*, September 2005.

[14] X. Wu and R. Srikant, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," in *Proceedings of IEEE INFOCOM'06*, April 2006, pp. 595–605.

[15] C. Joo, "A local greedy scheduling scheme with provable performance guarantee," in *Proceedings of ACM MOBIHOC'08*, Hong Kong SAR, China, May 2008, pp. 111–120.