

Maximizing Throughput in Wireless Networks via Gossiping

Eytan Modiano, Devavrat Shah, and Gil Zussman^{*}

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

{modiano, devavrat, gilz}@mit.edu

ABSTRACT

A major challenge in the design of wireless networks is the need for distributed scheduling algorithms that will efficiently share the common spectrum. Recently, a few distributed algorithms for networks in which a node can converse with at most a single neighbor at a time have been presented. These algorithms guarantee 50% of the maximum possible throughput. We present the *first distributed scheduling framework that guarantees maximum throughput*. It is based on a combination of a distributed matching algorithm and an algorithm that compares and merges successive matching solutions. The comparison can be done by a deterministic algorithm or by randomized gossip algorithms. In the latter case, the comparison may be inaccurate. Yet, we show that if the matching and gossip algorithms satisfy simple conditions related to their performance and to the inaccuracy of the comparison (respectively), the framework attains the desired throughput. It is shown that the complexities of our algorithms, that achieve nearly 100% throughput, are comparable to those of the algorithms that achieve 50% throughput. Finally, we discuss extensions to general interference models. Even for such models, the framework provides a simple distributed throughput optimal algorithm.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks]:Network Architecture and Design — *Wireless communication*; G.3 [Mathematics of Computing]:Probability and Statistics — *Probabilistic algorithms*

General Terms: Algorithms, Performance, Design

Keywords: Scheduling, Stability, Distributed algorithms, Gossip algorithms, Wireless networks, Matching

1. INTRODUCTION

One of the major challenges in the design and operation of wireless networks is to schedule transmissions to efficiently

^{*}The authors' names appear in an alphabetical order.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMetrics/Performance'06, June 26–30, 2006, Saint Malo, France.
Copyright 2006 ACM 1-59593-320-4/06/0006 ...\$5.00.

share the common spectrum among links in the same geographic area. A *centralized* scheduling policy that achieves the maximum attainable throughput region has been presented in the seminal paper by Tassiulas and Ephremides [23]. However, the lack of central control in wireless networks calls for the design of *distributed* scheduling algorithms. Such algorithms should achieve the maximum throughput or at least a guaranteed fraction of the maximum throughput. In this paper we present a distributed scheduling framework that guarantees *maximum throughput*. The framework is based on randomized algorithms that generate feasible schedules and on algorithms that compare and merge possible schedules in a distributed manner. Due to the distributed operation, the merge procedure may sometimes result in an inferior schedule. Despite that fact, we show that the framework achieves the desired throughput.

Different wireless technologies pose different constraints on the set of transmissions that can take place simultaneously. In this paper, we demonstrate our approach by focusing on primary interference constraints. These constraints imply that *each station can converse with at most a single neighbor at a time* (i.e. the set of active links at any point of time constitutes a matching¹). Networks operating according to these constraints are also known as node-exclusive spectrum sharing networks and have been studied in [3, 5, 6, 10, 24, 27].

One of the first attempts to deal with primary interference constraints was by Hajek and Sasaki [10] who proposed centralized algorithms for finding minimum length schedule to satisfy *given* traffic requirements. As mentioned above, the first attempt to deal with stochastic arrivals was in [23]. The results presented in [23] are applicable to general constrained resource allocation problems including scheduling in wireless networks with primary interference constraints. The routing and link activation policy presented there guarantees to stabilize the network (i.e. provide 100% throughput) whenever the arrival rates are within the stability region.

The results of [23] have been extended to various settings of *wireless networks*. For example, Neely et al. [20] recently developed a throughput optimal joint routing, scheduling, and power allocation algorithm for wireless networks with time-varying channel conditions and stochastic traffic. The set of transmission constraints in *input-queue switches* is very similar to primary-interference constraints in wireless networks. The main difference is that in a wireless network

¹A set of links is a matching, if no two links are incident to the same node.

the network graph is not necessarily bipartite and there is no central authority controlling the system. Scheduling in input queue switches has been extensively studied in the past (e.g. [18]).

The optimal algorithms developed in [18, 20, 23] require solving a *global optimization problem*, taking into account the queue backlog information for every link in the network. For example, for primary interference constraints a maximum weight matching problem has to be solved in every slot. Obtaining a centralized solution to this problem in a wireless network does not seem to be feasible, due to the communication overhead associated with collecting the network topology and queue backlog information, and due to the limited processing capability available to the nodes. This motivates us to study *distributed scheduling algorithms*, where by a distributed algorithm we mean the following:

- (i) Each node can communicate only with its neighbors.
- (ii) The computation at any node cannot utilize the global topological information. This disallows nodes to gather information about every network parameter and then perform computation locally.

The design of distributed scheduling algorithms has attracted a lot of attention recently. Lin and Shroff [17] studied the impact of imperfect scheduling on cross-layer rate control. Regarding primary interference constraints, they showed that using a distributed maximal matching algorithm along with a rate control algorithm may achieve 50% throughput. Similar results for different settings were also obtained in [6, 26]. Finally, Chaporkar et al. [5] study the problem from a different point of view. They characterize the stability region of a maximal scheduling policy under arbitrary topologies and interference models². We will elaborate more on [5, 6, 17, 26] in Section 7.

Despite these recent efforts, the effects of using imperfect or approximate distributed scheduling algorithms on the network throughput are still not fully understood. In particular, there are two key questions in this context. The first is what performance in terms of network stability results from using such algorithms. This question can be rephrased as: can a distributed algorithm achieve 100% throughput? The second is what are the tradeoffs between the *decentralization costs* (e.g. communication complexity and local computation complexity) and the network throughput. In this paper we try to answer these questions for a relatively simple network model. Thus, we deliberately *do not* consider in this paper issues of cross-layer design.

We consider a slotted multihop wireless network with a stochastic packet arrival process in which a node can converse with at most a single neighbor at a time. Our approach for designing distributed scheduling algorithms builds upon the ideas presented by Tassiulas [22] and Giaccone et al. [9]. The approach presented in [22] works as follows. In each time slot, a feasible solution to the maximum weighted matching problem is obtained. If the value of the new solution is higher than the value of the current solution, they are replaced. Using this approach guarantees achieving 100% throughput under certain conditions on the way in which the matching is obtained (see more details in Section 3). In

²For the node-exclusive spectrum sharing model, maximal scheduling reduces to maximal matching.

the context of input-queue switches, [9] proposed enhancements to this method which reduce the queueing delay. It was shown in [9] that merging the current and new matchings by selecting heavy edges from both of them still results in 100% throughput.

One of our main contributions is the extension of the frameworks of [9, 22] to a situation where exact comparison between different matchings is impossible. This is especially relevant in wireless networks where distributed algorithms should be used for the comparison. We show that if the difference between the preferable matching and the selected matching is bounded with high probability, the system is stable for any set of rates $(1 - \alpha - \beta)\Lambda^*$, where Λ^* is the stability region under a perfect scheduler. α and β are small constants that depend on the allowed difference between the preferable and selected matchings and on the probability that a maximum weight matching will be selected.

Another key observation is that feasible matchings can be compared and merged in a distributed manner by collecting *only semi-local information*. Namely, nodes may need information from a few hops away but in general no node will need to obtain information originating at all the nodes in the network.

Based on these observations, we propose a distributed framework that iteratively finds feasible solutions to the maximum weight matching problem and merges consecutive solutions. As mentioned above, the merge operation does not always need to select the preferable matching. The first phase of this combined framework is to obtain a solution to the *maximal matching* problem (i.e. a feasible solution to the *maximum weight matching* problem) by a randomized distributed algorithm. We consider a number of options for such an algorithm and discuss their performance.

The second phase is to *compare* and *merge* successive solutions. We propose three alternatives for collecting the information required for the comparison. The first approach is based on deterministically collecting information within local components of the network. This approach provides 100% throughput at time and communication complexities that are comparable to those of the algorithms proposed recently. The second approach is based on a gossip algorithm³ that disseminates information in a randomized manner in order to compute the difference between the values of the new solution and the old solution. One of the advantages of the gossip algorithm is that it does not require any infrastructure (such as unique addresses). We show that despite the randomized and perhaps inaccurate operation of the gossip algorithm, the stability region can be as close as desired to Λ^* . The third approach is based on a novel gossip algorithm. It computes estimates of the difference between the new and old solutions by distributedly computing minimum values of exponential random variables. The complexity of this approach is *significantly lower* than of the second approach and it can achieve the same stability region.

Finally, we note that in [5] it has been shown that using maximal-scheduling in networks with certain *secondary interference constraints* may reduce the stability region to $\Lambda^*/8$. Moreover, under general interference constraints, achieving 100% throughput (even in a centralized manner) requires to solve an NP-Complete problem (e.g. maximum weight independent set). Hence, generalizing our approach

³for more information on gossip algorithms see [4, 15] and references therein.

to other interference constraints may provide a significant improvement. We conclude by briefly discussing the required modifications to the scheduling and gossip algorithms that will allow them to distributedly achieve maximum throughput under general interference constraints.

The main contribution of this paper is the design of randomized distributed algorithms that attain 100% throughput with decentralization costs that are comparable to those of the maximal matching algorithms proposed recently. Recall that those algorithms may attain as low as 50% throughput. To the best of our knowledge, this paper is the first attempt to design distributed algorithms that achieve 100% throughput in a wireless network and to systematically quantify the complexities of such algorithms.

This paper is organized as follows. In Section 2 we present the network model and formulate the problem. In Section 3 we present the framework for obtaining maximum throughput in a distributed manner. Sections 4 and 5 present algorithms for obtaining the matching and for merging consecutive solutions. Implementation considerations and possible enhancements are discussed in Section 6. In Section 7 we evaluate and compare the performance of the different algorithms. Extensions to other interference models are briefly discussed in Section 8. We summarize the results and discuss future research directions in Section 9.

2. MODEL AND PROBLEM FORMULATION

We consider a wireless network modeled by a directed graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is the set of nodes and $E = \{(i, j) : i, j \in V\}$ is the set of *directed* links. We assume that the time is slotted, denoted by m , and that the packet length is normalized so as to be transmittable in a unit time slot.

At this stage we consider single-hop traffic⁴. Let $A_{ij}(m)$ denote the number of packets arriving at node i that wish to be transmitted to a neighboring node j at the end of time-slot m . In this paper, we consider $A_{ij}(\cdot)$ to be Bernoulli i.i.d. process⁵ with arrival rate λ_{ij} , that is $\Pr(A_{ij}(0) = 1) = \lambda_{ij}$. The vector of arrivals at time m is denoted by $A(m) = [A_{ij}(m), (i, j) \in E]^T$ and the arrival rate vector is denoted by $\Lambda = [\lambda_{ij}, (i, j) \in E]^T$.

Let $Q_{ij}(m)$ denote the number of packets destined to node j and queued at node i at the beginning of time-slot m . Let $Q(m) = [Q_{ij}(m), (i, j) \in E]^T$ denote the queue-size vector. For transmissions, a node needs to satisfy certain wireless-interference constraints. We consider primary interference constraints, also known as *node-exclusive spectrum sharing* (see [3, 5, 6, 10, 24, 27] for a similar setup). Under these constraints, each node can communicate with at most a single neighbor at a time (a node cannot transmit and receive simultaneously). Thus, the set of simultaneously active links form a matching in the graph G . Let $\Pi(G)$ denote the set of all feasible matchings (not necessarily maximal) in the graph G . In particular, let $\pi = [\pi_{ij}, (i, j) \in E]^T \in \Pi(G)$ be a 0 – 1 vector representing a matching.

⁴The results of the paper should be extendible to multi-hop case using the “back-pressure” mechanism [23] for obtaining edge weights and applying our distributed framework.

⁵Using fluid model techniques, it is not hard to extend the results for any arrival process satisfying the strong law of large numbers.

The scheduling decision must be made such that the schedule is a matching at each time. To this end, let $S_{ij}(m) \in \{0, 1\}$ be indicator variable of whether link (i, j) is active or not at time m . We assume that transmissions happen in the middle of a time slot (possibly utilizing information about $Q(m)$). Let vector $S(m) = [S_{ij}(m), (i, j) \in E]^T$ denote the scheduling decision vector. Then, $S(m) \in \Pi(G)$. A scheduling algorithm, in this paper, selects $S(m)$ based on the queue-sizes, $Q(k), k \leq m$, and arrivals $A(k), k \leq m$. Thus, the dynamics of the system can be described as

$$\begin{aligned} Q(m+1) &= A(m) + [Q(m) - S(m)]^+ \\ &= A(m) + Q(m) - D(m), \end{aligned} \quad (1)$$

where $D_{ij}(m) = \mathbf{1}_{Q_{ij}(m) > 0} S_{ij}(m)$ and $D(m) = [D_{ij}(m)]^T$.

DEFINITION 1 (ADMISSIBLE RATE-VECTOR). *An arrival rate vector Λ is called admissible, if there exists a collection of matchings, $\pi_j, 1 \leq j \leq K$ such that*

$$\Lambda \leq \sum_{j=1}^K \alpha_j \pi_j, \quad \alpha_j \geq 0, \quad \sum_{j=1}^K \alpha_j < 1.$$

DEFINITION 2 (STABILITY). *A scheduling algorithm is called stable if for any admissible Λ , the average queue-size is bounded, that is, $\limsup_m \mathbb{E}[Q_{ij}(m)] < \infty$, $(i, j) \in E$.*

DEFINITION 3 (STABILITY REGION). *The set of all admissible rate vectors Λ is called the stability region and is denoted by Λ^* .*

Tassiulas and Ephremides [23] established the existence of a stable scheduling algorithm. In particular, the algorithm that schedules according to $S^*(m)$ where

$$S^*(m) = \arg \max_{\pi \in \Pi(G)} Q^T(m) \pi, \quad (2)$$

is a stable algorithm. In the context of switch scheduling and node-exclusive spectrum sharing networks, this algorithm has to schedule the edges of the *Maximum Weight Matching* at each time slot, where the edge weights are the queue sizes. The maximum weight matching in any graph can be found in $O(n^3)$ computation time, using a centralized algorithm [16]. However, in wireless networks (unlike a router) implementing a centralized algorithm is not feasible. For bipartite graphs, distributed scheduling algorithms for finding a maximum weight matching are known [1, 2]. To the best of our knowledge, for arbitrary networks, no distributed algorithm for finding maximum weight matching is known. This motivates our goal: *Design a stable distributed scheduling algorithm for the wireless scheduling problem.*

Since such an algorithm has to be implemented in a distributed manner, we define the following standard performance measures [21]. We define the *Communication Complexity* as the number of messages sent over all links during the execution of the algorithm. The *Time Complexity* is defined as the number of time slots required, if the local computation time is negligible (significantly shorter than a slot) and if during each time slot a node can exchange errorless control messages with all its neighbors. Note that although this is a standard definition of the time complexity in a distributed algorithm, it does not adhere to the specific characteristics of networks with primary interference constraints. Yet, we use this standard model as a framework for comparing the performance of the different algorithms.

We will discuss this definition in more detail in Section 7. In some specific cases, we will also be interested in the *Local Computation Complexity* (it will become significant only for a distributed version of the centralized algorithm).

3. DISTRIBUTED SCHEDULING ALGORITHM

In this section, we describe a general framework for obtaining maximum throughput in a distributed manner. We present a generic algorithm, referred to as GEN-ALGO, and prove its throughput properties. In the following sections, we will elaborate on the distributed implementation of this algorithm. As mentioned in Section 1, this algorithm is motivated by the results of [9, 22].

GEN-ALGO

- (1) Let $S(m)$ denote the schedule at time m .
 - (2) Let $R(m+1)$ be a new schedule obtained by a distributed procedure NEW-SCH.
 - (3) The schedule at time $m+1$, $S(m+1)$, is obtained by a distributed procedure MIX using inputs $S(m)$ and $R(m+1)$.
-

We state and prove the following property for GEN-ALGO. The proof methodology is similar to that of [9, 22].

THEOREM 1. *Let the procedures NEW-SCH and MIX satisfy the following properties.*

- P1. *For any time m , define $X(m) = \mathbf{1}_{R(m)=S^*(m)}$, where $S^*(m)$ is a Maximum Weight Matching at time m . Let $X(m)$ be independent random variables with $\Pr(X(m) = 1) \geq \delta$, for some $\delta > 0$, for all m . Here the probability is with respect to randomness of NEW-SCH.*
- P2. *For any time m , with respect to the randomness of the MIX, $Q(m)^T S(m) \geq \max\{Q(m)^T S(m-1), (1-\gamma)Q(m)^T R(m)\}$, with probability at least $1-\delta_1$, such that $\delta_1 \ll \delta$.*

Then, the algorithm is stable for any arrival rate vector Λ such that

$$\Lambda \leq \sum_{j=1}^K \alpha_j \pi_j, \quad \alpha_j \geq 0, \quad \sum_j \alpha_j < 1 - \gamma - 2\sqrt{\frac{\delta_1}{\delta}}. \quad (3)$$

Before we prove the Theorem 1, we note that a system satisfying P1 and P2 is stable for any set of rates $(1-\gamma-2\sqrt{\delta_1/\delta})\Lambda^*$, where Λ^* is the stability region under a perfect (centralized) scheduler. Notice that the main difference from [9, 22] is the addition of property P2 and the identification of the tradeoff between γ, δ_1, δ , and the obtained throughput. Property P2 allows changing the schedule according to inaccurate estimations of the difference between the weights of $S(m)$ and $R(m+1)$. This opens the possibility for implementing the comparison mechanism in a distributed and possibly approximate manner.

Thus, if we manage to develop distributed algorithms NEW-SCH and MIX that satisfy properties P1 and P2 with small enough γ and δ_1 , then the corresponding GEN-ALGO algorithm will provide throughput which is almost the maximum

possible. In the following sections, we will discuss the trade-off between the loss in throughput in terms of γ and δ_1 and the time and communication complexity. We will first exhibit distributed algorithms NEW-SCH and MIX such that $\gamma = 0$ and $\delta_1 = 0$, thereby obtaining 100% throughput. In that case the MIX algorithm requires some coordination between the nodes. Therefore, we will present algorithms that require minimal coordination in which the corresponding γ and δ_1 can be made as small as required (by running the algorithms long enough).

PROOF OF THEOREM 1. We will use the standard Foster's criteria for proving stability of an algorithm. For this purpose, we will use standard quadratic Lyapunov function.

$$L(Q(m)) = Q(m)^T Q(m) = \sum_{(i,j) \in E} Q_{ij}^2(m).$$

Initially, we assume that $Q(0) = [0]$. We will study the drift in $L(\cdot)$ at every T^{th} time-slot for some large enough finite T . To this end, define $m_k = kT$, $k \geq 0$. From Foster's criteria it will suffice to establish that for some constant $\Phi > 0$,

$$\mathbb{E}[L(Q(m_{k+1})) - L(Q(m_k)) | Q(m_k)] \leq -\epsilon \|Q(m_k)\|_1 + \Phi, \quad (4)$$

to prove

$$\limsup_k \mathbb{E}[Q(m_k)] < \infty. \quad (5)$$

We note that the above expectation is with respect to the randomness of the whole Markovian system, conditioned on state $Q(m_k)$. This includes the randomness in future arrivals and algorithms.

Since, each queue-size can increase by at most T packets between any m_k and m_{k+1} , (5) will imply stability of the system. Next, we proceed to prove (4).

Given $Q(m_k) = Q(kT)$, let's consider the change in $L(\cdot)$ over $m \in [kT, (k+1)T]$. Let $A(m)$ denote the vector of arrivals and $S(m)$ denote the vector of schedules at time $m \in [kT, (k+1)T]$. Now, using (1)

$$\begin{aligned} & L(Q(m+1)) - L(Q(m)) \\ &= Q(m+1)^T Q(m+1) - Q(m)^T Q(m) \\ &= \sum_{(i,j) \in E} (Q_{ij}(m+1) - Q_{ij}(m)) (Q_{ij}(m+1) + Q_{ij}(m)) \\ &= \sum_{(i,j) \in E} (A_{ij}(m) - D_{ij}(m)) (2Q_{ij}(m) + A_{ij}(m) - D_{ij}(m)) \\ &= 2 \sum_{(i,j) \in E} Q_{ij}(m) (A_{ij}(m) - S_{ij}(m)) + (A_{ij}(m) - D_{ij}(m))^2. \end{aligned} \quad (6)$$

The last equality uses the fact that

$$Q_{ij}(m) D_{ij}(m) = Q_{ij}(m) S_{ij}(m).$$

Note that, for any m , $(A_{ij}(m) - D_{ij}(m))^2 \leq 1$. Using this bound along with summation of (6) for $m+1, \dots, m+T$, we obtain

$$\begin{aligned} & L(Q(m+T)) - L(Q(m)) \\ &\leq 2 \sum_{k=1}^{T-1} Q(m+k)^T (A(m+k) - S(m+k)) + 2(T-1)\mathcal{E}, \end{aligned} \quad (7)$$

where $\mathcal{E} = |E|$. Now, the arrival process is Bernoulli i.i.d. and hence taking conditional expectation with respect to

$Q(m)$ of (7) yields,

$$\begin{aligned} & \mathbb{E}[L(Q(m+T)) - L(Q(m))|Q(m)] \\ & \leq 2 \sum_{k=1}^{T-1} \mathbb{E}[Q(m+k)^T(\Lambda - S(m+k))|Q(m)] + 2(T-1)\mathcal{E}, \end{aligned} \quad (8)$$

Now, Λ is an admissible rate vector, that is for some $\pi_j \in \Pi(G)$, $1 \leq j \leq K$,

$$\Lambda = \sum_{j=1}^K \alpha_j \pi_j, \quad \alpha_j \geq 0, \quad \sum_j \alpha_j < 1. \quad (9)$$

Let $\rho = \sum_j \alpha_j < 1$. Also, let $S^*(m+k)$ denote the maximum weight schedule at time $m+k$, that is

$$S^*(m+k) = \arg \max_{\pi \in \Pi(G)} Q^T(m+k)\pi. \quad (10)$$

For simplicity, denote $W^*(m) = Q^T(m)S^*(m)$. For algorithm GEN-ALGO, denote

$$\Delta(m) = Q^T(m)S^*(m) - Q^T(m)S(m). \quad (11)$$

Putting (7)-(11) together, we obtain

$$\begin{aligned} & \mathbb{E}[L(Q(m+T)) - L(Q(m))|Q(m)] \leq \\ & -(1-\rho) \sum_{k=1}^T \mathbb{E}[W^*(m+k) + \Delta(m+k)|Q(m)] + \\ & 2(T-1)\mathcal{E} \leq -T(1-\rho)(W^*(m) - nT) + \\ & \sum_{k=1}^T \mathbb{E}[\Delta(m+k)|Q(m)] + 2T\mathcal{E}. \end{aligned} \quad (12)$$

To obtain (12), we have used the fact that the weight of the maximum weight matching can be reduced by at most n per time-slot in the worst case. That is, $W^*(m+k) \geq W^*(m) - nk$, unconditionally.

Next, we will use properties P1 and P2 of algorithms NEW-SCH and MIX to obtain the desired result. Motivated by P1, define $Z = \inf_{k \geq 1} \{R(m+k) = S^*(m+k)\}$, and

$$Z_1 = \inf_{k \geq 0} \{P2 \text{ fails at time } m+Z+k\}. \quad (13)$$

Now property P1 immediately imply that

$$\mathbb{E}[Z|Q(m)] = \mathbb{E}[Z] \leq 1/\delta.$$

Hence,

$$\mathbb{E}[\min\{Z, T\}] \leq \mathbb{E}[Z] \leq 1/\delta. \quad (14)$$

By property P2, it is easy to see that for $Z \leq k \leq \min\{T, Z_1\}$,

$$\begin{aligned} \Delta(m+k) & \leq 2(k-Z)n + \gamma W^*(m+Z) \\ & \leq \gamma W^*(m) + 2Tn. \end{aligned} \quad (15)$$

Further, it trivially follows that for $k \leq Z$,

$$\Delta(m+k) \leq W^*(m) + 2Tn. \quad (16)$$

Using (14)-(16), we obtain

$$\begin{aligned} & \sum_{k=1}^T \Delta(m+k) \\ & \leq W^*(m)(\min\{Z, T\} + \hat{T}) + \gamma W^*(m)T + 2T^2n, \end{aligned} \quad (17)$$

where $\hat{T} = T - \min\{T, Z_1\}$. Now, by P2 and simple union-bound it is easy to see that $Z_1 \geq T$ with probability at least $1 - \delta_1 T$. Hence,

$$\mathbb{E}[\hat{T}] \leq \delta_1 T^2. \quad (18)$$

From (17)-(18) and (12), we obtain

$$\begin{aligned} & \mathbb{E}[L(Q(m_{k+1})) - L(Q(m_k))|Q(m_k)] \\ & \leq -T(1-\rho-\gamma-(\delta T)^{-1}-\delta_1 T)W^*(m_k) + 2T(\mathcal{E} + nT). \end{aligned} \quad (19)$$

Now, if $\delta_1 = 0$, then for any $\rho < 1 - \gamma$, we can find large enough finite T the term $1 - \gamma - (\delta T)^{-1} > 0$. For $0 < \delta_1 \ll \delta$, choose $T = \sqrt{1/(\delta\delta_1)}$. Then, (19) becomes

$$\begin{aligned} & \mathbb{E}[L(Q(m_{k+1})) - L(Q(m_k))|Q(m_k)] \\ & \leq -T \left(1 - \rho - \gamma - 2\sqrt{\frac{\delta_1}{\delta}} \right) W^*(m_k) + 2T(\mathcal{E} + nT), \end{aligned} \quad (20)$$

with $T = \sqrt{1/(\delta\delta_1)}$. Since the $W^*(m_k)$ is the weight of maximum weight matching, for any G it is easy to see that

$$W^*(m_k) \geq \frac{\sum_{(i,j) \in E} Q_{ij}(m_k)}{\mathcal{E}}. \quad (21)$$

From (20) and (21), we obtain that for $\rho < 1 - \gamma - 2\sqrt{\frac{\delta_1}{\delta}}$, the desired Foster's criteria is satisfied, i.e.

$$\mathbb{E}[L(Q(m_{k+1})) - L(Q(m_k))|Q(m_k)] \leq -\epsilon \|Q(m_k)\|_1 + \Phi,$$

for some $\epsilon > 0$ and constant Φ . \square

4. DISTRIBUTED NEW-SCH

This section describes a NEW-SCH algorithm that has property P1, described in Theorem 1. We will show that it is enough to run this algorithm for a *constant number of iterations* in order to satisfy this property. In Section 6.2 we will discuss other possibilities for designing NEW-SCH algorithms that satisfy this property.

Note that GEN-ALGO as defined in Section 3 may change the schedule every time slot. However, the running time of the NEW-SCH and the MIX algorithms may be a few time slots. For the ease of presentation, in this section and in Section 5, we assume that the running time of the algorithms is shorter than the slot used for data transmission (perhaps by using mini-slots for control information). In Section 6 we will show that this assumption can be easily relaxed and that Theorem 1 holds even if the decision regarding a new schedule is made every several time slots.

Consider the following distributed randomized matching algorithm, that is described in [14].

RAND-MATCH

- (1) Initially, all n nodes of G are *unmatched*.
 - (2) While there are unmatched nodes with unmatched neighbors, do the following:
 - (i) Each unmatched node having at least one unmatched neighbor decides to be *left* or *right* with probability $1/2$ independently.
 - (ii) If a node, say v_l , becomes *left*, it sends a request to one of its unmatched neighbor uniformly at random.
 - (iii) If a node, say v_r , becomes *right*, on receiving requests from one or more *left* neighbors, it chooses one of them uniformly at random, say u . The nodes v_r and u are set as matched and they inform all of their unmatched neighbors about it.
-

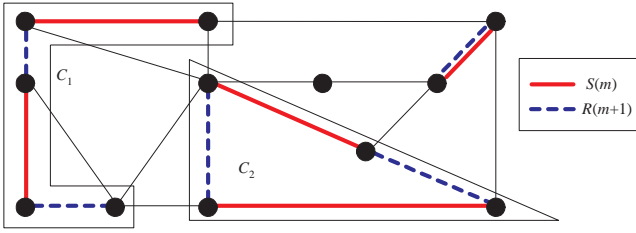


Figure 1: An example of the connected components composing the graph G_1 .

THEOREM 2 (JUNG AND SHAH, 2006 [14]). *For any graph G , the RAND-MATCH algorithm finds a maximal matching in $O(\log^2 n)$ iterations with probability at least $1 - 1/n^4$.*

Before discussing other properties of the algorithm, we note that its applicability to directed graphs is almost immediate. Namely, once an edge is selected, we define its direction from the *left* node to the *right* node. Although the time complexity is $O(\log^2 n)$, the following lemma shows that when RAND-MATCH is used as part of GEN-ALGO, there is no need to wait until it finds a maximal matching.

LEMMA 1. *For any maximal matching, $\pi \in \Pi(G)$, the algorithm RAND-MATCH finds it with probability at least $(2n)^{-n}$ at the end of the first iteration.*

PROOF. Consider $\pi \in \Pi(G)$. Consider a particular pair of nodes u and v that are matched in π by a directed edge (u, v) . Then, the probability that they will be matched by (u, v) at the end of the first iteration is $2^{-2} (d_u)^{-1} d_v^{-1}$, where d_u and d_v are the degrees of nodes u and v . These are less than n and hence the probability is at least $2^{-2} n^{-2}$. Now, using the same argument for the remaining (at most) $n/2$ node pairs and using independence of each node's decision, we obtain that the probability that all the node pairs in π are matched at the end of the first iteration is at least $2^{-n} n^{-n}$. \square

From Theorem 1 and Lemma 1, we see that it is sufficient to use RAND-MATCH for a finite number of iterations, which is not a function of the network size. However, since the corresponding $\delta = (2n)^{-n}$ is extremely small, we will need a MIX algorithm such that δ_1 is very small or preferably 0.

5. DISTRIBUTED MIX

We will describe three mechanisms to obtain a distributed MIX algorithm. Given two matchings $S(m)$ and $R(m+1)$, consider graph G_1 that contains edges only from $S(m)$ and $R(m+1)$. The graph G_1 is composed of connected components denoted by C_1, \dots, C_k . The connected components of G_1 are either: (i) even length cycles with alternate edges belonging to $S(m)$ and $R(m+1)$ or (ii) odd or even length paths again with alternate edges belonging to $S(m)$ and $R(m+1)$. Figure 1 illustrates a network graph and the graph G_1 composed of edges in $S(m)$ and $R(m+1)$.

The three proposed mechanisms merge matchings $S(m)$ and $R(m+1)$ to obtain $S(m+1)$ by using the MERGE procedure, described below. In the context of wireless networks, focusing on connected components is advantageous, since generally there is no need for global information regarding the total weight of $S(m)$ and $R(m+1)$.

MERGE($S(m), R(m+1)$)

- (1) For each component $C_i, 1 \leq i \leq k$, do the following:
 - (i) Consider a node $v \in C_i$. Let edge $e_{S(m)}^v$ and $e_{R(m+1)}^v$ be the edges containing this node v . Assign weights $W_v(S(m)) = Q_{e_{S(m)}^v}(m)$, and $W_v(R(m+1)) = Q_{e_{R(m+1)}^v}(m)$.
 - (ii) Obtain the sums of node weights of C_i ,
 $W(S(m)) = \sum_{v \in C_i} W_v(S(m))$,
 $W(R(m+1)) = \sum_{v \in C_i} W_v(R(m+1))$
 - (iii) If $W(S(m)) - W(R(m+1)) > 0$, then retain edges of $S(m)$ in C_i to obtain edges in $S(m+1)$. Else, replace edges of $S(m)$ in C_i by edges of $R(m+1)$ to obtain edges of $S(m+1)$.
-

In order to simplify the notation, in the rest of this section we will refer to $S(m)$ and $R(m+1)$ as S and R , respectively.

We note that if a connected component of G_1 is a path, a node v at the end of the path will be connected to either e_S^v or e_R^v . Therefore, in that case in step 1.i, it will be assigned either $W_v(S)$ or $W_v(R)$. Notice also that the sum of node weights is exactly twice the sum of edge weights.

From above, it is clear that to implement MERGE in a distributed manner, all we need is a distributed algorithm that obtains the sum over a connected component. The description of MERGE implies that if we can find the sums of node-weights in each connected component precisely, then the schedule $S(m+1)$ will satisfy property P2 of Theorem 1 with $\gamma = \delta_1 = 0$. We first describe a straightforward *summation* mechanism that finds the sums precisely. This requires coordination as well as some form of unique node identities. To make the algorithm totally free of any infrastructure, we will describe two randomized distributed (i.e. *gossip*) algorithms that find estimates of the sums within certain precision with certain confidence.

Similarly to Section 4, we assume that the running time of the algorithms is shorter than the slot used for data transmission. In addition, we assume that all nodes make the decision in Step 1.iii of MERGE simultaneously. In Section 6 we will show that these assumptions can be relaxed.

5.1 Summation Mechanism

Each node has its weight as assigned in the algorithm MERGE. In order to make the decision in Step 1.iii, the MERGE algorithm can find the sums $W(S)$ and $W(R)$, and then compute the difference or compute the difference directly. To this end, in the *summation mechanism* the difference between the sum of the edge weights is computed by passing messages along one direction in the path or cycle.

Specifically, a node, say v , sends a message with its unique signature to one of its neighbors in C_i . The message contains a variable D_v which is initially set to 0. If the message is sent along an edge from S , D_v is increased by $W_v(S)$. If the message is sent along an edge from R , D_v is decreased by $W_v(R)$. The receiving neighbor, say u , sends the updated message further along. Before sending the message, it increases D_v by $W_u(S)$ (if it is sent on an edge from S) or decreases it by $W_u(R)$ (otherwise). If the component is a cycle, then the message comes back to node v with the difference of between the sums of the edge weights on the whole

cycle. We note that a similar process should be initialized by every node along the cycle.

If the component is a path, then the node which is at the end of the path, say v , generates a special message indicating the end of the path. This message also includes the variable $D_v = W_v(S)$ (if it sent along an edge of S) or $D_v = -W_v(R)$ (otherwise). Every node along the path adds or subtracts the weight of the edge on which it forwards the message. Once such a message is received at the other end of the path, the difference between the sum of weights is sent back to the nodes along the path which use it to make their decision. All other messages received at the end of the path (i.e. messages generated by nodes within the path that are not aware of this fact) are discarded.

Such a mechanism requires unique identities and some coordination. For a cycle of length L ($L = |C_i|$), the time complexity (number of iterations) is L and for a path of length L , the time complexity is $2L$. For both cases, the communication complexity is $O(L^2)$, since at each iteration L messages are sent simultaneously. Since $L \leq n$, the time complexity is $O(n)$ and communication complexity is $O(n^2)$. The average message size is proportional to LW_v , since it carries the sum of weights. In such setup, the result of the MERGE algorithm is exact with probability 1 (i.e. in each component it always selects the heaviest matching). Hence, $\delta_1 = 0$ and $\gamma = 0$. This yields a distributed stable algorithm.

5.2 Gossip Mechanism

We first describe a gossip algorithm that finds, using only local information, an estimate of the sums required at step 1.ii of MERGE. It operates at each connected component, it does not require any form of global information, does not require any infrastructure (e.g. node identities), and it can find an average to any precision. This algorithm is required mostly due to existence of cycles and in case one wishes to avoid coordination within these cycles. We will show that the time complexity of the algorithm is relatively high. Yet, it provides the basis and the intuition for the *enhanced gossip algorithm* that will be presented in Section 5.3.

For each connected component, instead of computing the sums $W(S)$ and $W(R)$, the algorithm computes the average of the weights at the nodes (e.g. $\sum_{v \in C_i} W_v(S)/|C_i|$ for S). As can be seen from the description of MERGE, the average is sufficient for the comparison operation. We now describe the algorithm for S (the algorithm for R is exactly the same). For the ease of presentation, in the description of the algorithm we focus on cycles. Very minor changes are required in order to tailor the description to paths.

GOSP-ALGO I

- (1) Initially, each node v has weight $x_v(0) = W_v(S)$. Let $x_v(i)$ denote the value of node v in iteration i .
- (2) For iteration $i = 1, \dots, I$ do the following
 - (i) Each node, say v , decides to contact one of its two neighbors in G_1 with probability 1/2 and with probability 1/2 decides not to contact anyone. If it decides to contact a neighbor, it will contact either of the two neighbors with equal probability.
 - (ii) If a node, say u , is contacted by node v then following happens:
 - (a) if u has decided not to contact any neighbor, then it accepts v with probability 1/2 and they average their current values, that is,
$$x_u(i) = x_v(i) = \frac{x_v(i-1) + x_u(i-1)}{2}.$$
 - (b) Else, if node u has decided to contact node v , then they average.
 - (c) Otherwise, node u ignores node v and nothing happens for v .

- (3) Stop at the end of large enough iteration, say I , and use $x_v(I)$ as an estimate of node average.

The above algorithm is an adaptation of gossip algorithm of [4]. It is essentially averaging over a path or a cycle, say of length L using natural random walk whose spectral gap is $\Theta(L^{-2})$. Given initial values $x(0) = [x_v(0)]_{1 \leq v \leq L}$ for a cycle of length L , we define $x_{ave} = \sum_v x_v(0)/L$. We now restate the following result.

THEOREM 3 (BOYD ET AL., 2005 [4]). *The value at any node v at the end of iteration $I(L)$ of GOSP-ALGO I is such that*

$$\Pr \left(\frac{\sqrt{\sum_v [x_v(I(L)) - x_{ave}]^2}}{\sqrt{\sum_v (x_v(0))^2}} \leq \epsilon \right) \geq 1 - \delta_2,$$

where

$$I(L) = \Theta(L^2 [-\log \delta_2 - \log \epsilon]). \quad (22)$$

We define $\hat{\epsilon} = n\epsilon$. We also define $\hat{W}(S) = x_v(I(L))$ and $\bar{W}(S) = x_{ave}$, when averaging over the weights of S . Similarly, $\hat{W}(R) = x_v(I(L))$ and $\bar{W}(R) = x_{ave}$, when averaging over the weights of R . We use Theorem 3 to prove the following lemma.

LEMMA 2. *Under GOSP-ALGO I at the end of iteration $I(L)$ (defined in (22)),*

$$\Pr(x_{ave}(1 - \hat{\epsilon}) \leq x_v(I(L)) \leq x_{ave}(1 + \hat{\epsilon}); \text{ for all } v) \geq 1 - \delta_2.$$

PROOF. According to Theorem 1

$$\Pr \left(\sqrt{\sum_v [x_v(I(L)) - x_{ave}]^2} \leq \epsilon \sqrt{\sum_v (x_v(0))^2} \right) \geq 1 - \delta_2.$$

Now, for all v

$$|x_v(I(L)) - x_{ave}| \leq \sqrt{\sum_v (x_v(I(L)) - x_{ave})^2}$$

Also, $\sum_v (x_v(0))^2 \leq (\sum_v x_v(0))^2$. Hence,

$$\Pr \left(|x_v(I(L)) - x_{ave}| \leq \epsilon \sum_v x_v(0); \text{ for all } v \right) \geq 1 - \delta_2.$$

Therefore, $\Pr(x_v(I(L)) \in (x_{ave}(1 \pm \hat{\epsilon})); \text{ for all } v) \geq 1 - \delta_2$. \square

Lemma 2 implies that after $I(L)$ iterations, GOSP-ALGO I (at each node in the component) finds a good estimate of the average of the weights of S ($\hat{W}(S)$) with probability at least $1 - \delta_2$. Similarly, it finds a good estimate of the weights of R . For property P2 to hold, the decision will be to switch

to new schedule R only if $\hat{W}(R) > \left(\frac{1+\epsilon}{1-\epsilon}\right) \hat{W}(S)$. Now, the values at different nodes in a component may differ. At the extreme case, some of the nodes will decide that to keep S while others will decide to keep R . In that case, in step 1.iii of the MERGE algorithm, nodes in the same component may make contradicting decisions. We describe the following simple algorithm that should be initiated following GOSP-ALGO I in every connected component in order to take care of this case. The main idea is that if one node in a component C_i makes a different decision than the other nodes, then at least one node estimates that $\left(\frac{1+\epsilon}{1-\epsilon}\right) \hat{W}(S) - \hat{W}(R) > 0$, and therefore, all the nodes in C_i retain the previous matching (S).

AGREE

- (1) Initially, each node v has a *positive* or *negative* difference between the estimates of weights $\left(\frac{1+\epsilon}{1-\epsilon}\right) \hat{W}(S) - \hat{W}(R)$.
 - (2) Each node, say v , contacts its two neighbors in C_i , u and w . If the signs of their estimates differ, v and u send a *cancel change* message to their other neighbors.
 - (3) Upon receipt of the first *cancel change* message on an edge of C_i
 - (a) The node forwards it on the other edge of C_i .
 - (b) The node retains the edge of S in C_i to obtain the edge in $S(m+1)$.
 - (c) A node will discard any *cancel change* message received after the first one.
 - (4) If after L time slots a node does not send or receive a *cancel change* message, it will decide between S and R according to the value of $\left(\frac{1+\epsilon}{1-\epsilon}\right) \hat{W}(S) - \hat{W}(R)$.
-

It is clear that the time complexity of the AGREE algorithm is $O(L)$. The communication complexity is also $O(L)$, since at most one *cancel change* message will be forwarded by each node.

Once the AGREE algorithm halts in all the components, all the nodes completed step 1.iii of MERGE. We now show that following the execution of AGREE, the nodes select a preferable schedule with high probability. We define the event that no cancel message is sent during the AGREE algorithm as AGR . We state the following result.

LEMMA 3. If $\bar{W}(S) < \left(\frac{1-\epsilon}{1+\epsilon}\right)^2 \bar{W}(R)$, then

$$\Pr(AGR) \geq 1 - 2\delta_2.$$

PROOF. Assume that

$$\bar{W}(S) < \left(\frac{1-\epsilon}{1+\epsilon}\right)^2 \bar{W}(R). \quad (23)$$

Assume that a cancel message is sent during the AGREE algorithm. This means that there exists a node such that $\hat{W}(R) < \left(\frac{1+\epsilon}{1-\epsilon}\right) \hat{W}(S)$. From Lemma 2,

$$\Pr\left(\hat{W}(R) \geq (1-\epsilon)\bar{W}(R)\right) \geq 1 - \delta_2,$$

$$\Pr\left(\hat{W}(S) \leq (1+\epsilon)\bar{W}(S)\right) \geq 1 - \delta_2.$$

Putting these together, we obtain that if (23) is true then with probability at least $1 - 2\delta_2$ for all nodes,

$$\hat{W}(R) > \frac{1+\epsilon}{1-\epsilon} \hat{W}(S). \quad (24)$$

This is a contradiction to our assumption of disagreement. the hypothesis of the Lemma. That is, with probability at least $1 - 2\delta_2$, no cancel message is sent in algorithm AGREE when (23) holds. \square

GOSP-ALGO I runs for $I(L)$ iterations (recall that $L \leq n$) and the AGREE algorithm runs for L iterations, which are negligible in comparison to $I(L)$. Recall that

$$I(L) = \Theta(L^2[-\log \epsilon - \log \delta_2]).$$

The stability region is affected by ϵ and δ_2 ($\delta_1 = 2\delta_2$ and $\gamma \approx 4n\epsilon$). Using GOSP-ALGO I along with RAND-MATCH requires that δ_1 will be significantly smaller than $(2n)^{-n}$. We can, for example, set $\delta_2 = \beta^2(2n)^{-n}/8$ and $\epsilon = \alpha/(2n)$, for some small constants $\alpha > 0$ and $\beta > 0$. In that case the system will be stable for $(1 - \alpha - \beta)\Lambda^*$, where Λ^* is the stability region under a perfect scheduler. The time complexity of GOSP ALGO I with these parameters is $I(L) = O(L^2 n \log n)$. Notice that although the constants α and β may affect the number of iterations, they *do not* affect the worst case time complexity. Since in every iteration $O(L)$ messages are sent, the communication complexity is $O(L^3 n \log n)$. The message size during GOSP ALGO I is proportional to W_v .

At first glance the time complexity seems relatively high. However, we note that in every round, the nodes perform simple local computation and transmit small messages. As will be shown in Section 7, this is considerably better than collecting all the topology and backlog information and performing an $O(n^3)$ local computation. In the next section, we describe an improved gossip algorithm with significantly lower time complexity. In that algorithm reaching an agreement within a connected component is an inherent part.

5.3 Enhanced Gossip Mechanism

In this section we describe an alternative gossip algorithm (we refer to this algorithm as GOSP-ALGO II). This algorithm is randomized and is based on the following well-known fact.

LEMMA 4. Let X_1, \dots, X_k be independent random variables distributed according to exponential distribution of rates r_1, \dots, r_k respectively. Then, $X_* = \min_i X_i$ is distributed as exponential variable with rate $\sum_i r_i$.

We will also use the following fact that follows from a simple large deviation estimation, such as Cramer's Theorem [8].

LEMMA 5. Let X_1, \dots, X_k be i.i.d. exponential variables of rate r . Let $S_k = \sum_{i=1}^k X_i/k$. Then, for small enough $\gamma_1 > 0$,

$$\Pr(S_k \notin ((1-\gamma_1)r^{-1}, (1+\gamma_1)r^{-1})) \leq 2 \exp(-\gamma_1^2 k/2).$$

The above facts, Lemmas 4 and 5, can be used to compute the weights corresponding to R and S along a cycle (or a path) of length L as follows. Compute weights of R and S separately. Consider R . Each node v , as described in MERGE, assigns itself weight W_v equal to the weight of edge incident on v under R . Then, each node v draws an

exponential random variable of rate W_v . Denote this by X_v . Then, all the nodes along the cycle (or path) compute the minimum of these random variables. We denote this minimum by X_* . Repeat this process k times. Let the minimum values computed in these k times be $X_*(1), \dots, X_*(k)$.

We define, $S_k = \sum_{i=1}^k X_*(i)/k$. Then, $1/S_k$ is a good estimate of the sum of weights along the cycle, which we wanted to compute. Precisely, we state the following result which is a direct implication of Lemmas 4 and 5.

THEOREM 4. *Let $Z_k = 1/S_k$ be the estimate of the weights $W = \sum_v W_v$. Then, for small $\gamma_1 > 0$,*

$$\Pr(|Z_k - W| \notin (W(1 + \gamma_1)^{-1}, W(1 - \gamma_1)^{-1})) \leq 2e^{-\frac{\gamma_1^2 k}{2}}.$$

The above result, discussion in previous sub-section and Theorem 1 imply that by choosing $\gamma = 4\gamma_1 = \alpha$ and $k = \Theta((n \log n - \log \beta)/\alpha^2)$, for very small $\alpha > 0$ and $\beta > 0$ we obtain stability for any arrival rate in $(1 - \alpha - \beta)\Lambda^*$.

In order to compute S_k , each node has to compute the minimum of X_v ($v = 1, \dots, L$) and this should be done k times. Practically, each of the L nodes can send a vector with k values of X_v to its neighbors. Computing the minimum of these k values takes $O(L)$ iterations. During each iteration $O(L)$ messages are sent, and therefore, the overall communication complexity is $O(L^2)$. Each message contains a vector with k values of W_v . Thus, in the example above the message size is $O(n \log n W_v)$. Although α and β affect the exact message size, they do not affect the worst case.

We note that GOSP-ALGO II is *extremely efficient* as the time complexity is $O(L)$, whereas in GOSP-ALGO I it is $O(L^2 n \log n)$. In addition, even the product of the communication complexity and the message size is $O(L)$ less than in GOSP-ALGO I. Further, we do not need to run any AGREE mechanism, since all the nodes have exactly the same estimate. The GOSP-ALGO II algorithm does not require any form of centralized coordination or infrastructure. Finally, GOSP-ALGO II enables the application of our framework to networks with general interference constraints (see Section 8).

6. IMPLEMENTATION ISSUES AND ENHANCEMENTS

For the simplicity of presentation, in sections 4 and 5 we assumed that (i) the running time of the NEW-SCH and the MIX algorithms is shorter than the slot used for data transmission and (ii) all nodes make the decision in Step 1.iii of the MERGE algorithm simultaneously. In this section, we show that these simplifying assumptions can be easily relaxed and obtain a framework which is more amenable to distributed implementation. In addition, we discuss the applicability of a few distributed matching algorithms to the NEW-SCH procedure. Finally, we present enhancements to the algorithms used in the MIX phase. These enhancements may improve the average case performance.

6.1 Frames

We now assume that transmitting a control message of each of the various algorithms, discussed above, requires a time slot (identical to the time slot required for data transmission). In such a case, GEN-ALGO, using distributed algorithms for NEW-SCH and MIX, will not be able to find a

new schedule every time slot and to mix subsequent schedules immediately. Thus, we define a *frame* as the period required for running NEW-SCH and MIX in a distributed manner. The frame is divided into two time periods with predetermined lengths (t_{new} and t_{mix}). A version of GEN-ALGO implemented using frames is described below. GEN-ALGO-IMP.

- (1) Let $S(m)$ denote the schedule at time m (beginning of the frame).
- (2) At time m initiate the distributed procedure NEW-SCH at all nodes simultaneously. Run it for a *constant* number of slots t_{new} using the queue sizes at time m ($Q(m)$) as input. Let $R(m+t_{new})$ be the new schedule.
- (3) At time $m+t_{new}$ initiate procedure MIX at all nodes simultaneously. Run it for a given number of slots t_{mix} using $S(m)$, $R(m+t_{new})$, and the queue sizes at time m ($Q(m)$) as inputs to obtain $S(m+t_{new}+t_{mix})$.

The nodes should be aware in advance of t_{new} and t_{mix} in order to simultaneously start NEW-SCH and MIX. t_{new} can be any positive constant. For the summation algorithm t_{mix} should be set as $t_{mix} = O(n^2)$. Thus, the nodes need an estimate of n . At the network initialization (or whenever required if topology changes), a distributed algorithm such as GOSP ALGO II (with weights $W_v = 1$ for all v) can be used to obtain n . For GOSP-ALGO I, $I(L)$ iterations are required ($I(L)$ is defined in (22) as a function of ϵ and δ_2 that are set according to the required throughput). Then, for AGREE, L additional iterations are required. Since $L \leq n$, the nodes should set $t_{mix} = O(I(n) + n)$. For GOSP-ALGO II, $t_{mix} = O(n)$.

At this stage, we still assume that all the nodes make the decision in Step 1.iii of the MERGE algorithm simultaneously. Namely, all the nodes move to the next schedule at time $m+t_{new}+t_{mix}$. In Theorem 1, we assumed that the computation of a new schedule happens in the same time slot. However, the stability of GEN-ALGO-IMP will still be implied by Theorem 1 as long as we have bounded $t_{new} + t_{mix}$. This is due to the fact that the weight of any schedule changes by at most finite amount in the finite time. Hence, the drift in the appropriate Lyapunov function only gets less negative by an additive constant (i.e. in eq. (4) in the proof of Theorem 1). Hence, we obtain the following Lemma.

LEMMA 6. *Let the procedures NEW-SCH and MIX, performed during GEN-ALGO-IMP, satisfy properties P1 and P2, described in Theorem 1. Then, GEN-ALGO-IMP is stable for any arrival rate vector Λ satisfying (3).*

We now relax the assumption that all the nodes change their schedule at the same time. Alternatively, we assume that during the MERGE procedure, once a decision is made within a connected component C_i (probably before $m+t_{new}+t_{mix}$), the nodes in C_i change their schedule. This change improves the performance, since some of the components may be able to move to an improved schedule long before the summation or gossip mechanisms converge throughout the network⁶. For exactly the same reasons as for Lemma 6, the following holds.

⁶This is especially applicable to paths (see more comments in Section 6.3) and to summation in cycles.

LEMMA 7. Let the procedures NEW-SCH and MIX, performed during GEN-ALGO-IMP, satisfy properties P1 and P2, described in Theorem 1. The MIX procedure is performed such that once a decision is made within a connected component C_i , the nodes in C_i change their schedule. Then, GEN-ALGO-IMP is stable for any arrival rate vector Λ satisfying (3).

6.2 Other Alternatives for NEW-SCH

In Section 4, a NEW-SCH procedure that satisfies property P1, indicated in Theorem 1, has been presented. Actually, any matching algorithm that satisfies P1 can be used for NEW-SCH. Preferably, such an algorithm will yield a relatively large δ . A few distributed algorithms for finding a maximal matching have been proposed in the past (e.g. [12, 14]). In addition, two distributed approximation algorithms for the maximum weight matching problem have been recently proposed [11, 25].

This section describes two additional distributed NEW-SCH algorithms that have property P1, described in Theorem 1. We will show that it is enough to run these algorithms for a *constant number of iterations* in order to satisfy this property. The following algorithm is an adaptation of the SERENA algorithm of [9].

ARR-MATCH

- (1) At time m , create a graph $G_m = (V, E_m)$ as follows: edge $(i, j) \in E_m$ if and only if $A_{ij}(m) = 1$.
 - (2) Use RAND-MATCH for constant number of iterations to obtain a matching on the graph G_m .
-

We state the following straightforward result about the ARR-MATCH algorithm.

LEMMA 8. ARR-MATCH finds a matching $\pi \in \Pi(G)$ with probability at least $2^{-n}\Gamma(\pi)$ at the end of first iteration of RAND-MATCH on G_m , where

$$\Gamma(\pi) = \prod_{(i,j) \in \pi} \lambda_{ij} \prod_{i \notin \pi} (1 - \sum_j \lambda_{ij}),$$

with notation $i \notin \pi$ means that node i is not matched to any other node under matching π .

PROOF. $\Gamma(\pi)$ is the probability with which the arrivals in time m are only to edges of π and there are no arrivals to other edge in E . The algorithm RAND-MATCH matches all of the nodes in the first iteration with probability at least 2^{-n} . \square

We now present a similar result regarding the randomized algorithm presented by Israeli and Itai [12] whose time complexity is $O(\log n)$ (due to space constraints, we do not restate their algorithm here).

LEMMA 9. For any maximal matching, $\pi \in \Pi(G)$, the matching algorithm of Israeli and Itai [12] finds it with probability at least n^{-n} at the end of the first iteration.

PROOF. Consider $\pi \in \Pi(G)$. Consider a particular pair of nodes that are matched in π , say u and v . Then, the probability that they will be connected at the end of stage 1.1 is $d_u^{-1}d_v^{-1}$, where d_u and d_v are the degrees of nodes u and v . These are less than n and hence the probability is at

least n^{-2} . We can use the same argument for the remaining (at most) $n/2$ node pairs. We obtain that the probability that only edges from π remain in the graph at the end of stage 1.1 is at least n^{-n} . In that case these edges will be matched in stage 2.1. Since the algorithm has a few other stages in which edges can be matched, n^{-n} is a lower bound on the probability that all the nodes that are matched in π are matched at the end of the first iteration. \square

The main disadvantage of any randomized *maximal matching* algorithm is that $\delta \approx O(1/n!) \approx O(n^{-n})$. This requires a MIX algorithm with a very small δ_1 or $\delta_1 = 0$. Such a requirement substantially increases the time and communication complexity of the MIX algorithms. Hence, it is desirable to develop a distributed approximation algorithm for the *maximum weight matching* problem that satisfies property P1 with relatively large δ . However, since the current available approximation algorithms [11, 25] do not satisfy P1, this remains an open problem.

From a practical point of view, it might be desirable to run the NEW-SCH algorithm for a relatively long period (e.g. set $t_{new} = C \log n$, where C is a constant). Such a period is still negligible related to the time required for the MIX algorithm and it would allow selecting between better schedules. This will not change the stability region but it might provide better local solutions.

6.3 Enhancements for Paths

In the summation algorithm presented in Section 5.1, if a connected component of G_1 is a path, the two nodes at both ends of the path will indicate it in the messages they send. Thus, there are two types of messages: (i) messages originating from the end of the path and (ii) messages originating within the path. Essentially, there is no need for the second type. However, since the nodes along a path are not aware of the fact that they are not part of a cycle, they will still send their messages. We propose the following simple enhancement. After receiving a message originating from the end of the path, a node within the path will discard all messages from the second type. This will reduce the number of transmitted messages. Yet, it will not reduce the worst case time and communication complexities.

The gossip algorithms presented in Sections 5.2 and 5.3 can operate well without special messages initiated by nodes at the ends of a path. However, the performance will be improved if all the nodes start using gossip and stop, if they find out that they are not part of a cycle. In such a case, the nodes will always make the correct decision (i.e. for paths $\delta_1 = 0$ and $\gamma = 0$). Moreover, the time complexity along paths will be $O(L)$ and the amount of control information will be reduced. Finally, the paths will change to a better schedule faster.

7. PERFORMANCE EVALUATION

In this section we evaluate and compare the performance of algorithms that can be used in the framework of GEN-ALGO as well as other algorithms. First, we discuss the performance of a method that collects information from all the nodes and computes a solution to the maximum weight matching problem locally in each of the nodes. Then, we discuss the performance of our algorithms. Finally, we discuss the performance of the distributed algorithms, recently pro-

Table 1: Time complexity, communication complexity, local computation complexity, message size, addressing requirement, and stability region of the various algorithms (α and β are small constants)

Algorithm	Time Complexity	Communication Complexity	Local Comp.	Message Size	Addressing Requirement	Stability Region
“Distributed” Centralized Solution	$O(n)$	$O(n E)$	$O(n^3)$	$O(nW_v)$	Yes	Λ^*
RAND-MATCH+SUMMATION	$O(n)$	$O(n^2)$	$O(1)$	$O(nW_v)$	Yes	Λ^*
RAND-MATCH+GOSP-ALGO I+AGREE	$O(n^3 \log n)$	$O(n^4 \log n)$	$O(1)$	$O(W_v)$	No	$(1 - \alpha - \beta)\Lambda^*$
RAND-MATCH+GOSP-ALGO II	$O(n)$	$O(n^2)$	$O(1)$	$O(n \log n W_v)$	No	$(1 - \alpha - \beta)\Lambda^*$
[6],[17] (using [11])	$O(n)$	$O(E)$	$O(n)$	$O(1)$	No	$0.5\Lambda^*$
[5],[17],[26] (using [12])	$O(\log n)$	$O(E)$	$O(1)$	$O(1)$	No	$0.5\Lambda^*$

posed in [5, 6, 17, 26]. The performance measures are time complexity, communication complexity, message size, local computation complexity, the need for unique addresses, and the stability region. The results are summarized in Table 1.

A *distributed version of the centralized solution* requires to repeatedly collect queue backlog information from all the nodes. After each collection, there is a need to solve a maximum weight matching problem. Since in a wireless network there is usually no central authority responsible for network optimization, we assume that this is done by all the nodes. In Table 1 we refer to this algorithm as the “*Distributed*” *Centralized Solution*. Using such an algorithm, the network will be stable for all rates within the stability region Λ^* . The time to collect the required information is bounded by the network diameter, thereby, in the worst case it is $O(n)$. The communication complexity is $O(n|E|)$. The message size is bounded by $O(nW_v)$ (since a node may need to send the weight of n links). Finally, the most problematic part is the local computation that has to be performed after each collection phase. The complexity of the computation is $O(n^3)$, which may not be feasible in wireless nodes with limited computation capabilities.

In Table 1 we present the different performance measures for the summation algorithm. The combination of the RAND-MATCH and the summation algorithms will result in a network which is stable for all rates within the stability region Λ^* . The only disadvantage is that the summation algorithm requires unique addresses and some coordination between the nodes. The time and communication complexity of GOSP-ALGO I and of GOSP-ALGO II depend on the selected constants (γ and δ_1). However, some selected constants enable to approach Λ^* without increasing the complexity. In Table 1 we present the performance measures for both algorithms, where α and β are very small constants. We note that when any of the MIX algorithms is implemented along with RAND-MATCH, the complexities of RAND-MATCH are negligible.

As mentioned in Section 1, the design of distributed algorithms has attracted a lot of attention recently. The impact of imperfect scheduling on cross-layer rate control has been studied in [6, 17]. The stability region of a maximal scheduling policy under arbitrary topologies and interference models has been studied in [5]. In [6, 17] it has been shown that for the node-exclusive spectrum sharing model, using a maximal matching algorithm to schedule the transmissions allows to obtain at least 0.5 of the stability region. In [17] such an algorithm (termed Greedy Maximal Matching) selects the edges according to the queue lengths. In [6] the

distributed algorithm of Hoepman [11] is applied in order to obtain a maximal matching which is a 2-approximation to the maximum weighted matching. The communication complexity is $O(|E|)$, and the message size is $O(1)$. Under our model, the time complexity of this algorithm is $O(n)$. Assuming that each node sorts the edges in a preprocessing phase, the local computation complexity of this phase is $O(n)$. Maximal-matching-based algorithms that do not need to take into account the exact queue lengths are discussed in [5, 17, 26] and it is shown that they also attain 0.5 of the stability region. The complexities are not explicitly discussed in [5, 17, 26]. Yet, the algorithms can be implemented using the maximal matching algorithm of [12]. Therefore, their time complexity is $O(\log n)$, the communication complexity is $O(|E|)$, and the message size is $O(1)$.

Finally, notice that for all the algorithms discussed above, the time complexities were obtained assuming that at each time slot, every node can send a single control message to *all of its neighbors*. Although this is a standard definition used for comparing the performance of different distributed algorithms, it does not adhere to the specific characteristics of networks with primary interference constraints. Taking the transmission constraints into account will increase all the time complexities by $O(n)$.

8. GENERAL INTERFERENCE CONSTRAINTS

In the previous sections, we demonstrated our approach by focusing on primary interference constraints. In this section, we briefly discuss the immediate extension to general interference constraints. In many cases, an interference graph (also known as a conflict graph) G_I can be defined based on the network graph G [13]. Every link in G is represented by a node in the interference graph G_I . Nodes in G_I are connected, if the links that they represent interfere. In such a case, $\Pi(G)$ represents the set of all independent sets on the interference graph G_I and the optimization problem in (2) reduces to finding the *maximum weight independent set* in the interference graph G_I . This problem is NP-Complete, hard to approximate, and clearly cannot be solved distributedly every time slot or every frame.

Yet, the framework presented in Section 3 (GEN-ALGO) *can be applied to general interference constraints*. Specifically, the first required component is a distributed NEW-SCH algorithm that selects an *independent set* in the interference graph, such that the probability of selecting the maximum weight independent set is positive. Such an algorithm can be designed in a similar manner to the RAND-MATCH or

ARR-MATCH algorithms described above. The second component is an algorithm for comparing the obtained solution ($R(m+1)$) to the current solution ($S(m)$). A simple structure that divides the comparison into components does not seem to exist under general interference constraints. Therefore, the overall solutions have to be compared. Accordingly, all the nodes should obtain an estimate of the new solution. This can be done efficiently by a gossip algorithm such as GOSP-ALGO II. Using such an algorithm, all the nodes will have the same estimate, thereby enabling them to make the same decision.

9. CONCLUSIONS

This paper presents the first distributed scheduling algorithms for wireless networks that obtain 100% throughput. We have used a framework that generates random matchings and merges consecutive matchings by selecting heavy edges. We have shown that even if the merge procedure sometimes results in an inferior matching, under certain conditions the throughput will still be nearly 100%. Based on this framework and on the observation that the required information can be collected within local components, we have developed a few distributed algorithms including a very efficient gossip algorithm. We have discussed the performance of the algorithms and shown that they provide a significant improvement over the simple maximal matching algorithms. The presented framework can be extended to general interference constraints.

This work is the first approach towards *distributed* maximum throughput algorithms for wireless networks. Although we have made a theoretical contribution and have taken implementation constraints into account, much work is required in order to develop *truly* implementable algorithms. Some of the key issues in that context are: (i) how should the control messages be transmitted in a node-exclusive spectrum sharing model, (ii) what are the tradeoffs between throughput, delay, and decentralization costs, and (iii) how can the algorithms deal with an asynchronous network.

Separately, there are interesting algorithmic questions that remain unresolved such as improving the quality of the matching chosen by a random sampler. We note that arrival information based schemes (e.g. ARR-MATCH and the schemes in [9]) seem to perform extremely well (in terms of simulation), and therefore, are a subject for further research. Alternatively, we would like to develop a matching algorithm that will select the next matching in a clever way, based on some information regarding the current matching (e.g. [7]). Finally, since some of the results regarding max-min fairness in wireless networks (e.g. [24]) were obtained using the stability of a related system, we intend to explore the possibility to obtain max-min fair rates in a distributed manner.

Acknowledgments

This work was primarily supported by NSF CAREER of D. Shah. The other support came from NSF ITR grant CCR-0325401, by ONR grant number N000140610064, and by DARPA/ AFOSR through the University of Illinois grant no. F49620-02-1-0325. The research of Gil Zussman was supported by a Marie Curie International Fellowship within the 6th European Community Framework Programme.

10. REFERENCES

- [1] M. Bayati, D. Shah, and M. Sharma. Maximum weight matching via maxproduct algorithm. In *Proc. IEEE ISIT'05*, Sep. 2005.

- [2] D. Bertsekas and J. Tsitsiklis. Parallel and distributed computation: numerical methods. *MIT/LIDS Tech. Report*, 2003.
- [3] R. Bhatia, A. Segall, and G. Zussman. Analysis of bandwidth allocation algorithms for wireless personal area networks. *To appear in ACM/Springer Wireless Networks (WINET)*, 12, 2006.
- [4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: design, analysis and applications. In *Proc. IEEE INFOCOM'05*, March 2005.
- [5] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. In *Proc. 43rd Allerton Conf. on Commun., Control, and Comp.*, September 2005.
- [6] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle. Optimal cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *Proc. IEEE INFOCOM'06*, April 2006.
- [7] S. Deb, D. Shah, and S. Shakkottai. Fast incremental algorithm for repetitive optimization: an application to switch scheduling. In *Proc. CISS*, March 2006.
- [8] A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. 2nd edition, Springer, 1998.
- [9] P. Giaccone, B. Prabhakar, and D. Shah. Randomized scheduling algorithms for high-aggregate bandwidth switches. *IEEE J. Sel. Areas Commun.*, 21(4):546-559, May 2003.
- [10] B. Hajek and G. Sasaki. Link scheduling in polynomial time. *IEEE Trans. Inf. Theory*, 34(5):910-917, September 1988.
- [11] J.-H. Hoepman. Simple distributed weighted matchings. *eprint cs.DC/0410047*, October 2004.
- [12] A. Israeli and A. Itai. A fast randomized parallel algorithm for maximal matching. *Inform. Process. Lett.*, 22(2):77-80, January 1986.
- [13] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. *ACM/Springer Wireless Networks*, 11(4):471-487, July 2005.
- [14] K. M. Jung and D. Shah. Fast gossip via non-reversible random walk. In *Proc. IEEE ITW'06*, March 2006.
- [15] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proc. IEEE FOCS'00*, November 2000.
- [16] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, 1976.
- [17] X. Lin and N.B. Shroff. The impact of imperfect scheduling on cross-layer rate control in wireless networks. In *Proc. IEEE INFOCOM'05*, March 2005.
- [18] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proc. IEEE INFOCOM'96*, March 1996.
- [19] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. Available at www.arxiv.org:cs.NI/0504029, Feb. 2006.
- [20] M. Neely, E. Modiano, and C. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE J. Sel. Areas Commun.*, 23(1):89-103, January 2005.
- [21] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.
- [22] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *Proc. IEEE INFOCOM'98*, April 1998.
- [23] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Autom. Control*, 37(12):1936-1948, December 1992.
- [24] L. Tassiulas and S. Sarkar. Maxmin fair scheduling in wireless ad hoc networks. *IEEE J. Sel. Areas Commun.*, 23(1):163-173, January 2005.
- [25] M. Wattenhofer and R. Wattenhofer. Distributed weighted matching. In *Proc. DISC'04*, LNCS 3221:335-348, Springer, October 2004.
- [26] X. Wu and R. Srikant. Regulated maximal matching: a distributed scheduling algorithm for multi-hop wireless networks with node-exclusive spectrum sharing. In *Proc. IEEE CDC-ECC'05*, December 2005.
- [27] G. Zussman and A. Segall. Capacity assignment in bluetooth scatternets - optimal and heuristic algorithms. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 9(1):49-61, February 2004.