# Distributed Link Scheduling in Wireless Networks

Jean-Claude Bermond, Dorian Mazauric, Vishal Misra, Philippe Nain

# Distributed Link Scheduling in Wireless Networks*

Jean-Claude Bermond
Université Côte d'Azur-CNRS-Inria-I3S,
2004 Route des Lucioles, B.P. 93, F-06902, Sophia Antipolis, France
jean-claude.bermond@inria.fr

Dorian Mazauric
Université Côte d'Azur-Inria,
2004 Route des Lucioles, B.P. 93, F-06902, Sophia Antipolis, France
dorian.mazauric@inria.fr

Vishal Misra
Dept. Computer Science, Columbia University, New York, NY, USA
misra@cs.columbia.edu

Philippe Nain
Inria, ENS de Lyon/LIP,
46 avenue d'Italie, 69364, Lyon, France
philippe.nain@inria.fr

April 6, 2020

**Abstract**

This work investigates distributed transmission scheduling in wireless networks. Due to interference constraints, "neighboring links" cannot be simultaneously activated, otherwise transmissions will fail. Here, we consider any binary model of interference. We use the model described by Bui, Sanghavi, and Srikant in [7, 27]. We assume that time is slotted and during each slot there are two phases: one control phase in which a link scheduling algorithm determines a set of non interfering links to be activated, and a data phase in which data is sent through these links. We assume random arrivals on each link during each slot, so that a queue is associated to each link. Since nodes do not have a global knowledge of the queues sizes, our aim (like in [7, 27]) is to design a distributed link scheduling algorithm. To be efficient the control phase should be as short as possible; this is done by exchanging control messages during a constant number of mini-slots (constant overhead).

In this paper, we design the first fully distributed local algorithm with the following properties: it works for any arbitrary binary interference model; it has a constant overhead (independent of the size of the network and the values of the queues), and it does not require any knowledge of the queue-lengths. We prove that this algorithm gives a maximal set of active links, where for

*This work has been supported by ANR program ?Investments for the Future? under reference ANR-11-LABX-0031-01.

any non-active link there exists at least one active link in its interference set. We also establish sufficient conditions for stability under general Markovian assumptions. Finally, the performance of our algorithm (throughput, stability) is investigated and compared via simulations to that of previously proposed schemes.

Keywords: Wireless network ; Link scheduling algorithm ; Binary interference ; Distributed algorithm ; Stability ; Graph.

# 1 Introduction

Link scheduling is a key issue in communication networks as it drastically impacts the overall performance of the system (throughput, delay, etc.). It has received a lot of attention for both wired and wireless networks (radio, ad hoc network, sensor network, etc.) - see Section 3 for prior influential work in this area. In a wireless network, an additional difficulty arises from the existence of physical interference, as only transmissions which do not interfere with each one another over a given period of time should be scheduled. There are three main classes of interference models: protocols and geometric models [1, 15], in which a geometric notion of interference is used, SINR-based models [10], in which correct message reception at a receiver is driven by the experienced SINR (Signal-to-Interference-plus-Noise-Ratio) value, and graph-based models.
In this paper, built on a preliminary work by the authors in [3], we use the graph-based binary interference model (see e.g. [4, 7, 14, 38]). The transmission network is modeled by a graph (undirected or directed), where transmissions may only occur between two neighboring nodes. The interference is modeled by a binary symmetric relation between the links of the network (edges of the undirected graph or arcs of the directed graph) which indicates which pairs of links interfere. This model is only an approximation of the reality and is sometimes considered as simplistic; in practice, the SINR model is often used instead. But due to its mathematical tractability, the binary interference model has been extensively used for the analysis of protocols and networks. We refer the reader to [32, 33] for an evaluation of the accuracy of variuous interference models; for instance, it is shown that the binary interference model is accurate in highly directional antenna settings with obstructions.

We assume that time is slotted. Each time-slot (or simply slot) is composed of two phases: one control phase, in which a link scheduling algorithm determines a set of non-interfering links to be activated, and a data phase, in which data is sent through the links. We also assume that the message arrival process to each link of the network is stochastic, whose characteristics are not necessarily known to the network designers. As a result, a scheduling algorithm has to determine the set(s) of active links as a function of current and past arrivals. A queue is associated to each link. An efficient link scheduling algorithm should activate a set of non-interfering links that will yield the better performance in terms of throughput and delay; a well-known algorithm consists in activating (non-interfering) links such that the sum of their queue sizes (backlogs) is the largest possible.

A well-known example of binary interference is the primary node interference model (see [22]), where only links that do not share a common node can be active at the same time. A matching is a set of links in a graph without common nodes. So, in the primary node interference model, links can be simultaneously activated in the same slot only if they form a matching of the corresponding graph. Consider, for example, the four-node square grid network represented in Figure 1. Here, in each slot, we can simultaneously activate either the two vertical links of the matching 1 (Figure 1(a)) or the

two horizontal links of the matching 2 (Figure 1(b)). Note that it is also possible to only activate a single link in this network; however, it is more efficient to design maximal sets of active links which correspond in that case to a maximal matching. Figure 2(a) shows a matching in a larger grid. In the primary node interference model, activating a set of non-interfering links such that the sum of the sizes of their queues is the largest possible, corresponds to finding a maximum matching. Centralized algorithms have been proposed to solve this problem both for random [37, 36, 31, 28] and deterministic arrivals [19].

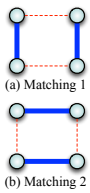



Figure 1: Scheduling algorithm for grid for $d = 0$.

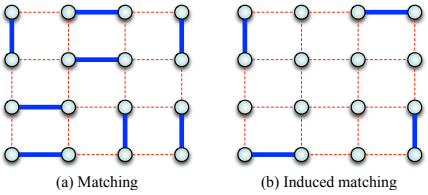



Figure 2: Two admissible schedules (selected active links for transmission represented by solid bold blue lines).

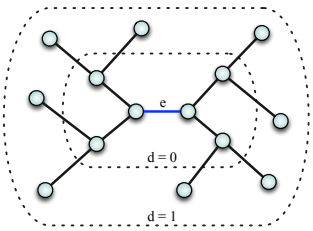



Figure 3: Interference sets of a link $e$ for $d = 0$ and $d = 1$.

Another binary interference model is the 802.11 interference model or distance-2 matching problem (see [38, 4, 2, 20]): here two links interfere if the first one contains a node adjacent to one of the second link. An induced matching $M$ in a graph $G$ is a matching, where no two edges of $M$ are joined by an edge of $G$. Therefore, a set of non-interfering link form an induced matching (see examples in Figure 2(b)).
The problem of finding a maximum induced matching in a graph is NP-Complete [35, 8] and it remains NP-Complete even for very special graphs (for 3-regular planar graphs for instance). So, in that case the problem of finding a set of non-interfering links maximizing the sum of backlogs is in general hard to solve, even in a centralized way.

Centralized algorithms make their decisions based on a total knowledge of the network, in particular, on the current link queue-sizes. While it is reasonable to assume some knowledge of the network, like its topology and the interference set (which remains stable during the use of the network), it is not realistic to assume a global knowledge of the link backlogs. Our aim (like in [7, 27]) is to design a distributed link scheduling algorithm which does not require any knowledge of the link backlogs. In passing, note that devising a distributed algorithm based on a full knowledge of the system state is very difficult to achieve due to the interference. Actually, acquiring this information is at least as much complicated as devising a decentralized link scheduling algorithm, which is the objective of this paper. In addition, exchanging messages with a central unit in a distributed setting will be both energy consuming and will generate high communication overheads, which will in turn affect the performance.
Finally, as noted by many authors, to be efficient the control phase should be as short as possible, thereby implying that exchange of control messages should be done during a constant number of mini-slots (constant overhead).

There are two different classes of scheduling algorithms for wireless networks: contention or random algorithms and collision-free (deterministic) algorithms. Algorithms in the former class may achieve

throughput optimality/near throughput optimality in terms of optimizing the wireless network resources (a precise mathematical is given by Tassiulas et al. [37] in the context of wireline networks) both without any local information [18, 30, 29] or with partial local information [26, 17, 16] but at the expense of some requirements (e.g. perfect carrier sense information) not available in practice. Throughput optimality is typically obtained at the expense of high latencies due, primarily, to the back-off mechanisms that are used to establish the optimality [34]. Algorithms in the second class will in general induce lower latencies than algorithms in the former class, in particular because they are collision-free, but it is extremely challenging to come up with any throughput optimality result for them. In this paper, we will focus on the class of deterministic collision-free algorithms.

Many (collision-free) distributed algorithms which have been proposed yield communication overheads which increase with the size of the network (see e.g. [6, 12, 25]). In particular, [14] presents a distributed algorithm valid for any binary interference model but at the expense of a non-constant overhead (which increases with the size of the network). The need for distributed algorithms with a small constant overhead has been emphasized in [7, 27], where a distributed algorithm with a constant overhead depending on the quality of the desired approximation is described; however, it is only valid for the primary node interference model. An overview of the algorithms in [7, 14] is given in Section 3.

Under these considerations, we make the following contributions to the general problem:

- we propose, to the best of our knowledge, the first distributed transmission algorithm – called `AlgoLog`– (Section 4) which:
  - holds for any binary interference model;
  - has a communication overhead independent of the size of the network;
  - gives a maximal set of active links, where for any non-active link there exists at least one active link in its interference set;
  - requires no explicit knowledge of the queue sizes.
- we formally prove some salient properties of AlgoLog and compute design tradeoffs. In particular, we show that the overhead grows logarithmically with the maximum node degree[1] (Section 4).
- under Markovian assumptions for the arrivals, we identify (Section 5) a set of admissible arrival rates at the links for which the system is stable, a result directly related to the achievable throughput.
- we investigate the performance of AlgoLog via simulations (Section 6).

Our algorithm uses a coloring of the links (such that two links with the same color do not interfere), which implies a knowledge of the binary interference. This knowledge can be acquired when the network is built, and remains unchanged during the use of the network.
Also observe that the "link model" above, which is commonly used in the literature (see e.g. [7, 9]), is idealized since in practice only nodes, and not links, can run a link scheduling algorithm. Indeed, a wireless link is immaterial and does not have any computing or sensing capability. The link model

[1] From a practical standpoint, this presents a more attractive design tradeoff since node degree is bounded by physical layer characteristics whereas the network size can be unbounded.

is often preferred to the node model as it allows the obtention of generic results, and as binary interference is only defined between links. In addition to the contributions in Section 4-6, we show in Section 7 how `AlgoLog` can be emulated if buffers are located in the nodes.

First, notation and model are introduced in Section 2 and algorithms in [7, 14] are discussed in Section 3.

## 2 Definitions, notation, network and interference models

Throughout the paper $\mathbb{N} := \{0, 1, \ldots\}$ is the set of nonnegative integers and $\mathbb{N}^* := \mathbb{N} - \{0\}$.
The network is modeled as a graph $G = (V, E)$, where a link exists between two nodes (or vertices) $u \in V$ and $v \in V$ if both nodes are within transmission range of one another. If the graph is undirected the link will correspond to the edge $\{u, v\}$. If the graph is directed the link will correspond to the arc $(u, v)$ if $u$ is the sender and $v$ is the receiver. In practice the networks are symmetric digraphs, such that if there exists an arc $(u, v)$, then there exists also the arc $(v, u)$.
Due to interference, not all links can successfully transmit messages in the same slot. Here, we use the so-called Binary Symmetric Interference model. In such a model, to each link $e$ is associated a set of interfering links, denoted by $I(e)$. Note that due to symmetry, if $e'$ belongs to $I(e)$, then $e$ belongs to $I(e')$. We assume that $e$ does not beling to $I(e)$.
The (distributed) link scheduling algorithm proposed in Section 4 applies to any kind of transmission graph (directed or undirected) and interference sets $I(e)$, $e \in E$. However, for the sake of both simplicity and concreteness in the examples, we will mainly consider undirected graphs with the $d$-interference model (with $d$ fixed in $\mathbb{N}$) defined below. Here $d(u, v)$ is the distance in $G$ between $u$ and $v$, that is the length of a shortest path between these two nodes.

$$\forall e = \{u_1, u_2\} \in E, I(e) = \{\{v_1, v_2\} \in E \setminus \{e\}, \exists i, j \in \{1, 2\}, d(u_i, v_j) \leq d\}. \quad (1)$$

In other words, two links interfere if one node of the first link is located at distance at most $d$ in $G$ from a node of the second link. As said in the introduction, two particular cases have been mainly studied in the literature. The case $d = 0$ is known as the primary node (or node exclusive) interference model. In this model, two links interfere if they are incident, so that an admissible schedule forms a matching of the transmission graph $G$, namely, a set of links without common nodes (Figure 2(a)). The more realistic model with $d = 1$ is known as the 802.11 interference model. In this model, two nodes may communicate only if their neighbors are not involved in a communication. Said otherwise, in graph theory an admissible schedule is called an induced matching of $G$, namely a matching where no two edges of $M$ are joined by an edge of $G$ (Figure 2(b)). Two instances of binary interference sets are represented in Figure 3 for these two models: when $d = 0$ (respectively $d = 1$) messages are successfully transmitted on link $e$ during a slot if all links other than link $e$ within the first circle (respectively second circle) remain silent in that slot.
We assume that time is slotted and denote by slot $t$ the time interval $[t, t+1)$, $t \in \mathbb{N}^*$. Transmissions on different links are synchronized and the unit of information to be transmitted between two nodes is called a message. We denote by $c(e) \in \mathbb{N}^*$ the capacity of link $e \in E$, defined as the maximum number of messages that can be transmitted on link $e$ in a slot.
Link $e \in E$ is equipped with a buffer of size $B(e) \in \mathbb{N}^*$ messages called buffer $e$. We denote by $q_t(e)$ the weight of link $e$, defined as the number of messages in buffer $e$, at the beginning of slot $t$, waiting to be transmitted on link $e$. If link $e$ is allowed to transmit messages in slot $t$, it will (successfully)

| | |
|---|---|
| $I(e)$ | Links interfering with link $e$, excluding $e$ |
| $d$ | $d$- interference model defined by (1) |
| $d = 0$ | Primary node interference model |
| $d = 1$ | 802.11 node interference model |
| $c(e)$ | Capacity of link $e$ (in number of messages/slot) |
| $B(e)$ | Size of buffer $e$ (in number of messages; $B(e) \in [1, \infty]$) |
| $q_t(e)$ | Weight of link $e$ (= number of messages in buffer $e$) at beginning of slot $t$ |
| $C$ | Number of colors |
| $w_t(e)$ | Virtual weight of link $e$ at beginning of slot $t$ function of $q_t(e)$ and $C$ (8) |
| $\nu_t(e) = (\nu_t(e,1), \ldots, \nu_t(e,T))$ | Control vector of link $e$ in slot $t$ |
| $S$ | Number of control sub-phases in a slot |
| $T + 1$ | Number of mini-slots in a sub-phase |
| $K \geq 2$ | Number of possible intervals for the virtual weights |
| $L$ | Initial value of the last interval |

Table 1: Glossary of main notation.

transmit $\min(c(e), q_t(e))$ messages in that slot. A link $e \in E$ is said to be busy in slot $t$ if buffer $e$ is non-empty at the beginning of slot $t$ or, equivalently, if $q_t(e) > 0$.
The purpose of a link scheduling algorithm is to identify an admissible schedule in each slot, namely, a set of non-interfering busy links. Furthermore we want to insure the stability of the system. At this time, we do not need to specify the queue length of the weights $\{q_t(e)\}_t$, $e \in E$. Such a model will be introduced in Section 5 when we study the stability of the distributed link scheduling proposed in Section 4.

# 3 Related works

In this section, we briefly review two algorithms proposed in [14] and [7], respectively. They are both distributed but the algorithm in [14] does not admit a constant overhead, whereas the one described in [7] is valid only for the primary node interference model ($d = 0$).
In the algorithm described in [14], for each time-slot $t \geq 1$ there are a control phase and a data phase, like in our modeling (Section 2). Before each control phase, composed of $T$ mini-slots, each link $e \in E$ is undetermined, and chooses a backoff value $t(e)$, $1 \leq t(e) \leq T$. In this context undetermined means that the link does not know if it will be active or inactive during the data phase of the current slot. If link $e$ receives a message (from a link in $I(e)$) during a mini-slot $t$, $1 \leq t \leq t(e) - 1$, then $e$ becomes inactive. Otherwise $e$ sends a control message during mini-slot $t(e)$ and if it does not receive a message, then $e$ becomes active ($e$ is inactive otherwise). At the end, a valid set of active links is computed, allowed to send messages during the data phase. This simple algorithm is valid for any interference set, but the choice of the backoff value $t(e)$, for any link $e \in E$, is a function of the weights of links located in its interference set $I(e)$ and function of the weights of links located in interference set of each $e' \in I(e)$. More precisely $t(e)$ depends on the weights of links belonging to

the set $I(e) \cup \{e'' : e'' \in I(e') \text{ for some } e' \in I(e)\}$. Thus, at each time slot $t \geq 1$, each link has to update the weights of links located in its $2d - neighborhood$ (links at distance at most $2d$ from $e$), if we have an interference model based on distance $d$. Therefore the overhead is not constant and furthermore one has to obtain this information, which, due to interference, is a problem as difficult as the transmission scheduling problem.
The algorithm described in [7], `Augmenting Paths Algorithm`, has a constant overhead but it is specific to the primary node interference model ($d = 0$). It uses the "augmenting path tool" developed for matching theory [21] and which is used to find polynomial centralized algorithms to determine maximum matchings. In that case, at each slot $t \geq 1$, a valid set of active links is a matching of the transmission graph $G = (V, E)$. The main idea of `Augmenting Paths Algorithm`, is to compute, at a slot $t + 1 > 1$, a matching of $G$ from the matching of $G$ found at slot $t$. In [7], it is proved that, if $2k + 1$ is the maximum length of the augmenting paths, the algorithm needs a constant overhead of order $4k + 2$ and achieves $\frac{k}{k+2}$ of the capacity region (a refined analysis of their algorithm can be shown to give a tighter bound of $\frac{k}{k+1}$).
Moreover at the beginning of each slot $t \geq 1$, each node $v \in V$ becomes seed with a constant probability $p$. A seed is a node allowed to start an alternating path. In [7], it is not described how to compute $p$ analytically. Finally it is necessary to precise that the augmenting path technique works only for matchings in graphs and so this algorithm is specific to the primary node interference model ($d = 0$), not the more realistic one. Recall that the problem of determining a maximum matching (if the interference model is defined by $d = 0$) can be done in polynomial time with a centralized algorithm, but for interference models defined by $d \geq 1$, determining a maximum valid set of links is an NP-complete problem. In Section 6.2 we compare via simulations the performance of `Augmenting Paths Algorithm` to the performance of our distributed algorithm, proposed in Section 4, for a square grid composed of 121 nodes.

# 4 Presentation of `AlgoLog`

## 4.1 Aim of the control phase

In this section, we propose a distributed link scheduling algorithm - called `AlgoLog` - with a constant communication overhead which is valid for any binary interference model. (Recall that for the sake of simplicity, we will only consider binary d-interference sets as defined Equation (1).) This is in contrast with the algorithm proposed in [14] whose communication overhead increases with the size of the network, and with the algorithm proposed in [7, 27] which works only for the primary node interference model ($d = 0$, see Section 2).
Link scheduling algorithms proposed in the literature like those in [7, 14] require exchanges of information between nodes (typically, any node needs to know the number of pending messages of its neighbors) which is in itself a difficult task due to interference. Our algorithm does not require any exchange of information between nodes and is therefore fully distributed.
In `AlgoLog` the time is slotted, with slot $t$ referring to the time interval $[t, t + 1)$. A slot is composed of two different phases, a control phase followed by a data phase. The aim of the control phase is to determine all links – called **active links** – which will be allowed to send messages in the data phase of slot $t$. At slot $t$, a link $e \in E$ is said busy if $q_t(e) > 0$. Links with $q_t(e) = 0$ will not participate in the control phase. Our first goal is to ensure that all active links form a maximal admissible schedule, i.e. to determine a maximal set $E'$of non-interfering busy links (a set is maximal if a busy link in $E - E'$ cannot be activated without interfering with a link in $E'$). To do that, `AlgoLog` will

use interference as information. Indeed, when a signal is sent on a link $e \in E$, every link $e' \in I(e)$ hears it.
In addition to finding a maximal admissible schedule, `AlgoLog` activates all links whose virtual weights are local maximum (see just after the definition of a virtual weight which is related to the values of $q_t(e)/c(e)$).

## 4.2 Virtual weights and their binary representation

Throughout the paper $W$ is a fixed nonnegative integer whose role and value will be discussed later on. To each link $e \in E$, we associate an integer $w_t(e)$ at the beginning of slot $t \geq 1$ – called the virtual weight of link $e$ at time $t$ – defined precisely in Section 4.5 (see (9)).
The mapping $e \to w_t(e)$ will satisfy the following properties: for all $t \geq 1$,

$$w_t(e) \in \{0, 1, \ldots, W\},$$

$$w_t(e) = 0 \text{ if and only if } q_t(e) = 0, \tag{2}$$

and two interfering busy links at time $t$ will have different virtual weights namely,

$$\text{if } q_t(e) > 0 \text{ and } q_t(e') > 0 \text{ then } w_t(e') \neq w_t(e) \text{ for all } e' \in I(e), e \in E. \tag{3}$$

There are two main reasons for working with virtual weights, and not with the actual weights: one is to impose that two interfering busy links at time $t$ will have different (virtual) weights, a feature that `AlgoLog` will use, and the other one is to parametrize (through $W$) the duration of the control phase, typically to limit its duration - see Section 4.5.

Since the virtual weights $\{w_t(e), e \in E\}$ always lie in the set $\{0, 1, \ldots, W\}$, we may (and will) identify $w_t(e)$ with its binary representation the (so-called) control vector $\nu_t(e)$ defined by

$$\nu_t(e) = (\nu_t(e, 1), \ldots, \nu_t(e, T)) \in \{0, 1\}^T, \tag{4}$$

where binary numbers $\nu_t(e, 1), \ldots, \nu_t(e, T)$ enter the decomposition of $w_t(e)$ in base 2, namely,

$$w_t(e) = \sum_{i=1}^{T} 2^{T-i} \nu_t(e, i), \tag{5}$$

with $T := \min\{q \in N^* : W \leq 2^q - 1\}$. Alternatively,

$$T = \lceil \log_2(W + 1) \rceil, \tag{6}$$

with $\lceil x \rceil$ the smallest integer greater than or equal to $x$. Table 3 shows the control vectors associated to values of $w_t(e)$ between 1 and 15 (here $T = 4$).

**Require:** $w_t(e)$.
**Ensure:** return active ($s(e) = A$) or inactive ($s(e) = I$).
1: $e$ computes $\nu_t(e, i)$, $i = 1, \ldots, T$
2: $s(e) = I$ if $w_t(e) = 0$ otherwise $s(e) = U$ ($w_t(e) > 0$)
3: **for** $j = 1, \ldots, S$ **do**
4: **for** $i = 1, \ldots, T$ **do**
5: **if** $s(e) = U$ and $\nu_{t,e}(i) = 1$ **then**
6: $e$ sends a signal (heard by links in $I(e)$)
7: **if** $e$ does not hear any signal **then**
8: $s(e) = A$
9: **if** $s(e) = U$, $\nu_t(e, i) = 0$, and $e$ hears a signal **then**
10: $s(e) = PI$
11: **if** $s(e) = A$ **then**
12: $e$ sends a signal (of synchronization)
13: **if** $s(e) = PI$ and $e$ does not receive a signal **then**
14: $s(e) = U$
15: **if** $s(e) = PI$ and $e$ receives a signal **then**
16: $s(e) = I$
17: return $s(e)$

Table 2: Control phase of `AlgoLog` at link $e \in E$ in slot $t$.

### 4.3 The algoritm `AlgoLog`

The control phase is composed of $S \geq 1$ successive sub-phases labeled $j = 1, \ldots, S$ with a mini-slot of synchronization between two consecutive sub-phases. We will see later that we can choose $S = T$, so that the parameter $T$, or equivalently $W$ from (6), controls the duration of the control phase.
At the end of the control phase a link will be in one of the two final states: **active** (state A) or **inactive** (state I). However, during the running of the algorithm a link can also be in one of the two other temporary states: **undetermined** (state U) or **potentially inactive** (state PI). During a sub-phase an undetermined link may become either definitely active or potentially inactive. The objective of the mini-slot of synchronization between two sub-phases is to determine which potentially inactive links will become definitely inactive; the others will become undetermined and will participate in the next sub-phase.
We can now describe precisely the algorithm which is also represented in pseudocode in Table 2, where $s(e) \in \{A, I, U, PI\}$ gives the state of link $e$. The reader can also follow the running of the algorithm in the examples in Figures 6-8. In Figure 6 the graph consists of a cycle with 9 links (with $d = 0$); to make the figure more readable we have repeated the first node $u$ at the end. In Figure 7 the graph is a grid with 16 nodes and 24 links (with $d = 0$), and in Figure 8 the graph is a random graph with 48 nodes and 91 links (with $d = 1$). The random graph has been generated using the method described in Section 6.3. Active links are indicated with thick bold blue lines, inactive links with thin dashed red lines, undetermined links with thin black lines, and potentially inactive links with dashed black lines.
At the beginning of the algorithm (**line 2**) link $e$ marks itself as inactive if $w_t(e) = 0$ and as undetermined otherwise ($w_t(e) > 0$). Each sub-phase is divided itself into $T$ (so-called) mini-slots,

labeled $i = 1, \ldots, T$. Each link $e$ undetermined at the beginning of the mini-slot $i$ does the following in the mini-slot $i$:

- (**lines 5-8**) if $\nu_t(e, i) = 1$, link $e$ sends a signal (which will be heard by all links in $I(e)$). If link $e$ does not hear anything from other links during mini-slot $i$, it marks itself as active, otherwise it stays undetermined;
- (**lines 9-10**) if $\nu_t(e, i) = 0$, link $e$ does not transmit in mini-slot $i$. If link $e$ does not hear anything from other links during mini-slot $i$, then it stays undetermined; otherwise it marks itself as potentially inactive.

Figure 4 describes the four transitions previously detailed.

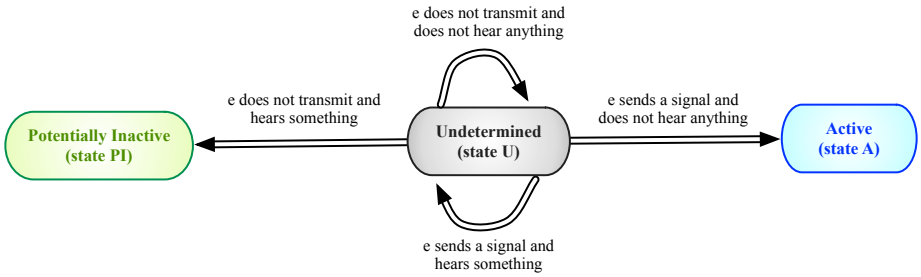



Figure 4: The four transitions and the three states in one sub-phase of `AlgoLog` for a link $e$.

In Figure 6 at the beginning of the algorithm the last link is inactive (thin dashed red line) as $q_t(e) = 0$ and all the other links are undetermined (solid black lines). Here $T = 4$ and the 4 mini-slots of sub-phase 1 are indicated in lines (d)-(g). At the end of the 4th mini-slot only link 5 is active (represented by a solid bold blue line), the others being potentially inactive or inactive (for the last link).

`AlgoLog` could stop after the end of the first sub-phase (**line 10** with $j = 1$ and $i = T$) since, by construction, all active links form an admissible schedule. It is however possible that this admissible schedule is not maximal. Such a situation is depicted in Figure 6 (for $d = 0$), where at the end of mini-slot $T$ (in this example $T = 4$) only link 5 is active and the others are potentially inactive (or inactive). But we will see that in this example, at the end of the $S$ subphases of Algolog, links $1, 3, 5, 7$ are active yielding a maximal admissible schedule (Figure 6 (n)).

To enforce an admissible schedule to be maximal, the following is done: at the end of all the sub-phases, but the last one, each active link transmits a signal, called a synchronization signal (**lines 11-12**). All potentially inactive links that do not hear anything during this additional mini-slot mark themselves as undetermined (**lines 13-14**); the other potentially inactive links (i.e. those that do hear something from neighbor(s)) are marked inactive (**lines 15-16**). `AlgoLog` then enters the second sub-phase (**line 4** with $j = 2$, $i = 1$) and the previous process is repeated until the end of the $S$-th sub-phase (**line 17** with $j = S$ and $i = T$).
Figure 5 describes the transitions previously detailed.

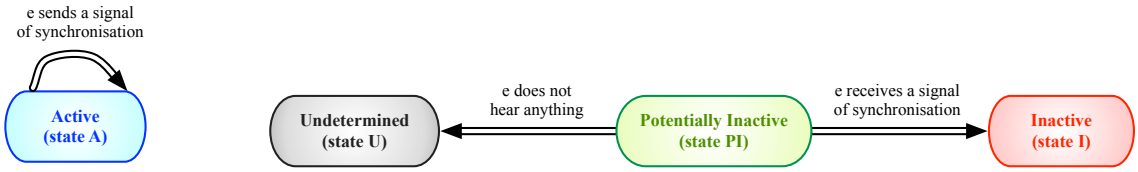



Figure 5: Synchronisation.

We will show in Proposition 2 that property (3) together with the choice for $S = T$ implies that `AlgoLog` always generates a maximal admissible schedule.

In Figure 6 (h), during the mini-slot of synchronization links 4 and 6 become inactive the other links $1, 2, 3, 7, 8$ becoming undetermined. After the first four mini-slots of sub-phase 2 (Figure 6 (i)-(l)), links 3 and 7 become active links and $1, 2, 8$ are potentially inactive. During the mini-slot of synchronization (Figure 6 (m)) links $2, 8$ become inactive and link 1 undetermined. At the end of the next sub-phase (Figure 6 (n)) link 1 becomes active.

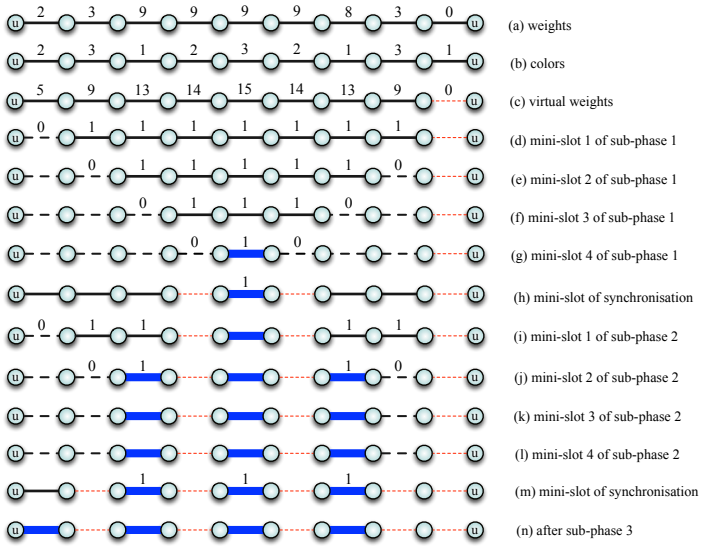



Figure 6: Output of `AlgoLog` for a cycle composed of 9 links for $d = 0$, $C = 3$, $K = 5$, $L = 4$, $W = 15$, and $T = 4$. Active links are indicated with thick bold blue lines, inactive links with thin dashed red lines, undetermined links with thin black lines, and potentially inactive links with dashed black lines.

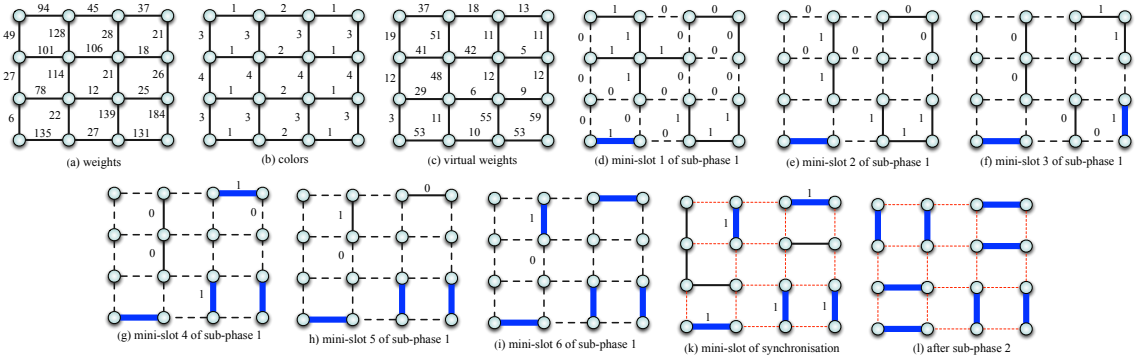



Figure 7: Output of `AlgoLog` for a grid composed of 24 links for $d = 0$, $C = 4$, $K = 15$, $L = 140$, $W = 60$, and $T = 6$.

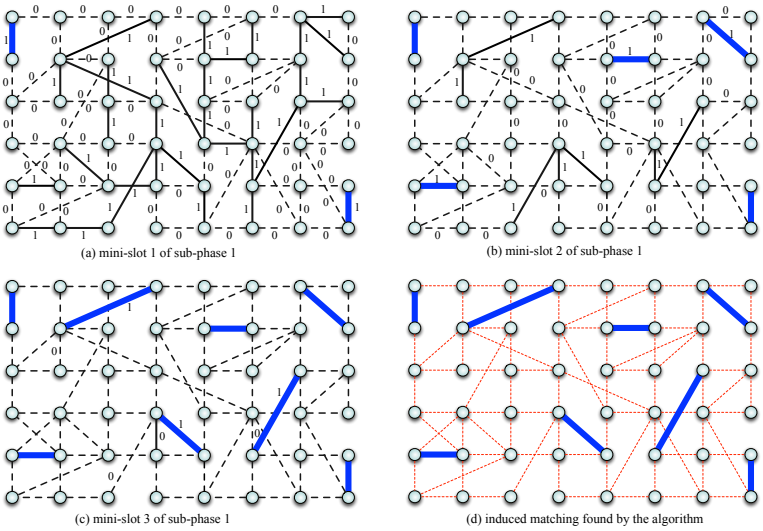



Figure 8: Output of `AlgoLog` for a graph composed of 91 links for $d = 1$.

## 4.4 Data phase

All active links at the end of the control phase of a slot transmit during the data phase of that slot. More specifically, if link $e$ is active in slot $t$ as the result of the control phase of `AlgoLog`, then it will transmit $\min(q_t(e), c(e))$ messages during the data phase of that slot.

## 4.5 Definition of the virtual weights

Recall that we want that two interfering links have different virtual weights. Furthermore, to enhance the performance of `AlgoLog` (stability, delay, ...), we want as much as possible to schedule links with the highest backlogs or, by fairness to links with small bandwidth, links with the highest ratios $q_t(e)/c(e)$ in slot $t$ ($t \geq 1$).
To overcome the problem that interfering links may have the same virtual weights, we will use an admissible coloring of the links, which consists in associating to each link $e$ a color $\chi(e)$ so that any two interfering links have different colors. Let $C$ be the number of colors of such an admissible coloring. We have $\chi(e) \in \{1, 2, \ldots, C\}$. We will see in Section 4.7 how to compute or approximate the minimum value for $C$ in the $d$-interference model. Figure 6 (b) shows an admissible coloring of a cycle graph with 9 links with $C = 3$ and Figure 7 (b) an admissible coloring of the grid with $C = 4$ colors, both within the primary node interference model ($d = 0$).
In addition, in order to ensure that links with the same virtual weights are selected in an equitable manner, we will use a time-dependent coloring, which associates to each link $e$ in slot $t \geq 1$ a color $\chi_t(e) \in \{1, 2, \ldots, C\}$ defined by:

$$\chi_t(e) = (\chi(e) + t - 1) \bmod(C) \quad e \in E,\, t \geq 1, \tag{7}$$

where $p \bmod(C) = p - C\lfloor (p-1)/C \rfloor$ for any integer $p \geq 1$, so that $p \bmod(C) \in \{1, 2, \ldots, C\}$.

As the initial coloring $\chi_1 := (\chi(e), e \in E)$ is admissible, $\chi_t := (\chi_t(e), e \in E)$ is also an admissible coloring for all $t \geq 2$. We state this fact as a lemma:

**Lemma 1** *If $e$ and $e'$ are interfering links, then $\chi_t(e) \neq \chi_t(e')$ for all $t \geq 1$.*

From now on we will work under the following assumption.

**Assumption 1 (Initial coloring)** *The initial coloring $\chi(e)$, $e \in E$, is known and fixed throughout.*

To insure that links with the highest ratios $q_t(e)/c(e)$ will have more chance to be scheduled in slot $t$, to each link $e$ with a positive weight (i.e. $q_t(e) > 0$) we will assign the virtual weight $w_t(e)$ defined as

$$w_t(e) = \begin{cases} Cg(q_t(e)/c(e)) + \chi_t(e) & \text{if } q_t(e) > 0 \\ 0 & \text{if } q_t(e) = 0. \end{cases} \tag{8}$$

The mapping $g$ is defined as follows: $[0, \infty)$ is partitioned into $K \geq 2$ (necessarily) disjoint sets $I_0, \ldots, I_{K-1}$, where $I_0 \cup \cdots \cup I_{K-2} = \{x, x \leq L\}$ and $I_{K-1} = \{x, x > L\}$ with $0 \leq L < \infty$. Then,

$$g(x) = k \quad \text{if } x \in I_k \quad \text{for } k = 0, 1, \ldots, K-1.$$

In summary, either $q_t(e) = 0$ and then $w_t(e) = 0$, or $q_t(e) > 0$ and

$$w_t(e) = C \sum_{k=0}^{K-1} k \, \mathbf{1}\{q_t(e)/c(e) \in I_k\} + \chi_t(e) \in \{1, \ldots, CK\}. \tag{9}$$

Note that there is no need to select $L \geq \max_{e \in E} B(e)/c(e)$ since $\max_{e \in E} q_t(e)/c(e) \leq \max_{e \in E} B(e)/c(e)$ for all $t$. Also notice that, as $k \leq K-1$ then $w_t(e) \leq C(K-1) + C = CK$ for all $t$, so that the maximum virtual weight $W$ satisfies

$$W = CK. \tag{10}$$

From (6) and (10), we obtain

$$T = \log_2(\lceil CK + 1 \rceil). \tag{11}$$

For the first example of the cycle (cf. Figure 6 and Table 3), we choose $L = 4$, $K = 5$, intervals $I_k = (k, k+1]$ for $k = 0, 1, 2, 3$ and $I_4 = (4, \infty)$. For the example of the grid (Figure 7) we choose $L = 140$, $K = 15$ and $I_k = (10k, 10k + 10]$ for $0 \leq k \leq 13$ and $I_{14} = (140, \infty)$.

The following lemma shows that property in Eq. (3) holds.

**Lemma 2** *Let $e, e' \in E$ be two busy interfering links in slot $t \geq 1$. Then, $w_t(e) \neq w_t(e')$.*

**Proof.** Fix $t \geq 1$. There exist $j$ and $k$ such that $\lfloor q_t(e)/c(e) \rfloor \in I_j$ and $\lfloor q_t(e')/c(e') \rfloor \in I_k$. Assume that $j \neq k$. Then (cf. (8))

$$\begin{aligned} |w_t(e) - w_t(e')| &= |C(j-k) - (\chi_t(e') - \chi_t(e))| \\ &\geq C|j-k| - |\chi_t(e') - \chi_t(e)| \qquad \text{(Hint:} |a-b| \geq |a| - |b|) \\ &\geq C|j-k| - (C-1) \qquad (12) \\ &\geq 1, \end{aligned}$$

where (12) follows from the fact that $1 \leq \chi_t(e), \chi_t(e') \leq C$. Hence, $w_t(e) \neq w_t(e')$. Now, assume that $j = k$. We have (cf. (8)) $w_t(e) - w_t(e') = \chi_t(e) - \chi_t(e')$. Therefore, as $\chi_t(e) \neq \chi_t(e')$ by Lemma 1, we get that $w_t(e) \neq w_t(e')$. ∎

| $q_t(e)/c(e)$ | $\chi_t(e)$ | $w_t(e)$ | $\nu_t(e,1)$ | $\nu_t(e,2)$ | $\nu_t(e,3)$ | $\nu_t(e,4)$ |
|---|---|---|---|---|---|---|
| (0,1] | 1 | 1 | 0 | 0 | 0 | 1 |
| (0,1] | 2 | 2 | 0 | 0 | 1 | 0 |
| (0,1] | 3 | 3 | 0 | 0 | 1 | 1 |
| (1,2] | 1 | 4 | 0 | 1 | 0 | 0 |
| (1,2] | 2 | 5 | 0 | 1 | 0 | 1 |
| (1,2] | 3 | 6 | 0 | 1 | 1 | 0 |
| (2,3] | 1 | 7 | 0 | 1 | 1 | 1 |
| (2,3] | 2 | 8 | 1 | 0 | 0 | 0 |
| (2,3] | 3 | 9 | 1 | 0 | 0 | 1 |
| (3,4] | 1 | 10 | 1 | 0 | 1 | 0 |
| (3,4] | 2 | 11 | 1 | 0 | 1 | 1 |
| (3,4] | 3 | 12 | 1 | 1 | 0 | 0 |
| $> 4$ | 1 | 13 | 1 | 1 | 0 | 1 |
| $> 4$ | 2 | 14 | 1 | 1 | 1 | 0 |
| $> 4$ | 3 | 15 | 1 | 1 | 1 | 1 |

Table 3: $(w_t(e), \nu_t(e))$ for all entries $(q_t(e)/c(e), \chi_t(e))$ for $C = 3$, $K = 5$, $L = 4$, and $W = 15$.

### 4.6 `AlgoLog` generates a maximal schedule

We will first show that `AlgoLog` activates all links whose virtual weights are local maximum, and then `AlgoLog` generates a maximal admissible schedule. We say that a busy link $e$ is a local maximum in slot $t \geq 1$, if $w_t(e) > w_t(e')$ for all $e' \in I(e)$. Note that, by Lemma 2, $w_t(e) \neq w_t(e')$ for $e' \in I(e)$, which justifies the strict inequality in the definition of a local maximum.

**Proposition 1** *A local maximum link is always active at the end of a control phase of* `AlgoLog`*. More precisely, it is already active at the end of sub-phase* 1*.*

**Proof.**
Assume that link $e$ is local maximum and that it is not active at the end of mini-slot $T$ of sub-phase 1. So it is either PI (potentially inactive) or U (undertermined) at the end of mini-slot $T$ of sub-phase 1. If link $e$ has become PI in mini-slot $i$, $1 \leq i \leq T$, that is due to the existence of a link $e' \in I(e)$ such that link $e'$ is U at the beginning of mini-slot $i$ and coordinate $i$ of their control vectors satisfy $\nu_t(e', i) = 1$ and $\nu_t(e, i) = 0$. If link $e$ is U, let $i$ be the last mini-slot in which $e$ has sent a message; as it stayed U there exists a link $e' \in I(e)$ such that link $e'$ is U at the beginning of mini-slot $i$ and coordinate $i$ of their control vectors satisfy $\nu_t(e', i) = 1$ and $\nu_t(e, i) = 1$. But, as $w_t(e) > w_t(e')$, in both cases there exists an $\ell < i$ such that $\nu_t(e, \ell) = 1$ and $\nu_t(e', \ell) = 0$ and so at mini-slot $\ell$, $e'$ should have become PI, yielding a contradiction. ∎

In essence, Proposition 1 says that links with 'large' weights (from a local point of view) will be more likely to be scheduled for transmission since the virtual weight of a link is a non-decreasing function of its weight (see (8) and (9)).
The statement in Proposition 1 should however be interpreted with care. In particular, it does not imply that, in any slot $t$, `AlgoLog` always finds a maximum schedule that is one for which the sum

of the virtual weights of the activated links is maximized. To illustrate this point, consider a path composed of three links: $e_1$, $e_2$, and $e_3$. Let us suppose we have the primary node interference model ($d = 0$); so $C = 2$. Assume that the virtual weights satisfy: $w_t(e_2) > w_t(e_1) = w_t(e_3)$ but $w_t(e_1) + w_t(e_3) > w_t(e_2)$. As a result, `AlgoLog` will only activate link $e_2$ (which is a local maximum) in slot $t$ (note that this schedule is maximal). Note, however, that a maximum schedule consists of link $e_1$ and $e_3$ as $w_t(e_1) + w_t(e_3) > w_t(e_2)$, thereby showing that `AlgoLog` does not always find a maximum schedule. Note also that the situation might be worse with the real weights. Indeed we can have, by equation 8, $w_t(e) > w_t(e')$, if $\chi_t(e) > \chi_t(e')$ and $q_t(e)/c(e)$ and $q_t(e')/c(e')$ belong to the same interval although $q_t(e')/c(e')$ might be considerably larger than $q_t(e)/c(e)$.
The above example shows that `AlgoLog` will not allow the network to operate at its maximum throughput (defined as the average number of transmissions per slot). Indeed, in the example above, it would be more efficient to activate links $e_1$ and $e_3$ instead of link $e_2$ (the throughput can be the double if $q_t(e_2) = q_t(e_1) = q_t(e_3)$ and $\chi_t(e_2) = 2$ and $\chi_t(e_1) = \chi_t(e_3) = 1$). This is of course not surprising as this is the price to pay for using a decentralized scheduling algorithm like `AlgoLog`.

We now prove that `AlgoLog` finds a maximal schedule when $S \geq T$, namely, a schedule with the property that each inactive busy link in $E$ is such that there is (at least) one active link in its interference set.

**Proposition 2** *Let $S \geq T = \log_2(\lceil CK + 1 \rceil)$. For each slot $t \geq 1$ and for every $e \in E$ such that $q_t(e) > 0$, there exists at least one link $e' \in I(e) \cup \{e\}$ such that $e'$ is active at the end of the control phase of slot $t$.*

**Proof.** Consider an arbitrary slot $t$ and an arbitrary link $e$ such that $w_t(e) > 0$ (i.e. $e$ is busy).

- If $e$ is active at the end of the control phase of $t$, then the result is proved by taking $e' = e$.
- If $e$ is inactive at the end of the control phase of $t$, that is due to the fact that, when $e$ became inactive (during a mini-slot of synchronization), there was some link $e' \in I(e)$ active, and the proposition is proved with $e'$.
- If $e$ is neither active nor inactive at the end of the control phase of slot $t$, then it is U (undetermined) at the end of all sub-phases (after the mini-slot of synchronization). We will see that if the number $S$ of sub-phases satisfies $S \geq T$ this case cannot happen. More precisely, we will prove that in any sub-phase $j$, $j = 1, \ldots, S$, there exists a link which became PI no sooner than in mini-slot $2 + S - j$. In particular, at the end of sub-phase 1, there exists a link which became PI no sooner than in mini-slot $S + 1$, which yields a contradiction when $S \geq T$.

First, let us see how a link $f$ can be PI or U at the end of mini-slot $T$ of any sub-phase. The reader can follow the proof on the example of Figure 9, where `AlgoLog` is applied to a path with 5 links, labeled $e_1, e_2, e_3, e_4, e_5$, $d = 0, W = 8$, and $T = 3$.

Case 1: Let $f$ be PI at the end of mini-slot $T$ of a sub-phase and let $i(f)$ be the mini-slot of this sub-phase in which $f$ became PI. The latter is due to the existence of a link $f' \in I(f)$ which, at mini-slot $i(f)$, was U with the $i(f)$ coordinate of its control vector given by $v_t(f', i(f)) = 1$ (while $v_t(f, i(f)) = 0$). Hence, $f'$ will be PI, U or A at the end of the sub-phase. Note that, if $f = e$, then $f'$ cannot be A, as otherwise $e$ would become inactive during the mini-slot of synchronization,

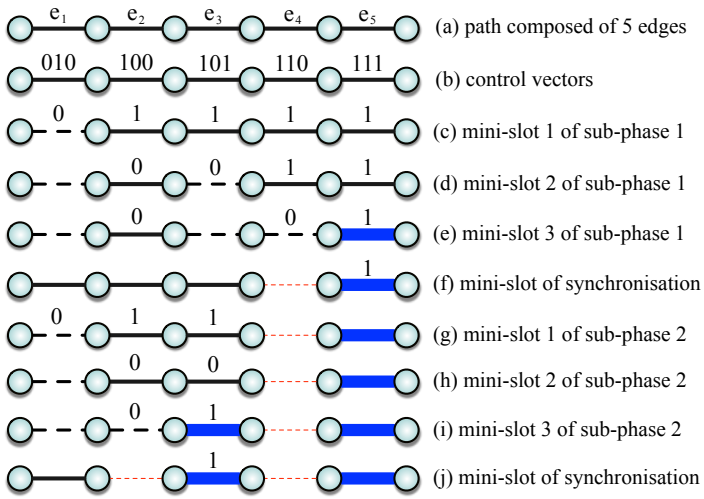



Figure 9: Example showing the necessity of $S \geq T$ in Proposition 2. Active links are indicated with thick bold blue lines, inactive links with thin dashed red lines, undetermined links with thin black lines, and potentially inactive links with dashed black lines.

implying that it would be inactive at the end of the control phase of $t$, thereby contradicting the assumption that it is U or PI at the end of the control phase of t.
In the example of Figure 9 it is the case in the sub-phase 1 for link $e_1$ (respectively $e_3$, $e_4$) which became PI at mini-slot 1 (respectively 2, 3) or in sub-phase 2 for links $e_1$ and $e_2$.

Case 2: Let $f$ be U at the end of mini-slot $T$ of some sub-phase and let $i(f)$ be the last mini-slot of this sub-phase in which link $f$ has sent a signal. We claim that it is due to the existence of a link $f' \in I(f)$ which became PI in a mini-slot $i(f') > i(f)$. Indeed, if link $f$ is U at the end of mini-slot $T$ then it was U at mini-slot $i(f)$. But the latter is due to another undetermined link $f' \in I(f)$, which also sent a signal during mini-slot $i(f)$. Let $v_t(f)$ and $v_t(f')$ be the control vectors of $f$ and $f'$, respectively.
First, note they have the same coordinates for $i = 1, \ldots, i(f)$. To see why, suppose that for some $i < i(f)$, $v_t(f, i)$ and $v_t(f', i)$ differ; then, as both links $f$ and $f'$ were U, the one with the coordinate 0 should have become PI due to the other one, and so could not have been U at mini-slot $i(f)$. Now, since $w_t(f) \neq w_t(f')$ from Lemma 2, there is a mini-slot $l > i(f)$, where $v_t(f, l)$ and $v_t(f', l)$ differ. However, as $i(f)$ was the last mini-slot in which $f$ sent a signal, we have $v_t(f, l) = 0$ and so $v_t(f', l) = 1$. But $f$ being U at the end of mini-slot $T$, this is possible only if, at mini slot $l$, $f'$ is PI. Therefore, $f'$ became PI at some mini-slot $i(f') > i(f)$ as, at mini-slot $i(f)$, it was U with coordinate 1.
In the example of Figure 9, this situation shows up for link $f = e_2$ at the end of sub-phase 1. Indeed, $e_2$ sent its last signal in mini-slot 1 and stayed U in this mini-slot due to link $f' = e_3$. Here, $v_t(e_2, 3) = 0$ and $v_t(e_3, 3) = 1$; but $e_3$F and so $e_2$ stays U at mini-slot 3.

Recall that we suppose $e$ is U or PI at the end of mini-slot $T$ of all sub-phases. Consider a sub-phase

$j$.

- If $e$ is U at the end of mini-slot $T$ of sub-phase $j$ and if $i_1^j$ is the last mini-slot of this sub-phase $j$ in which link $e$ has sent a signal, then by Case 2 there exists a link $e_2^j \in I(e)$ which became PI in a mini-slot $i_2^j > i_1^j \geq 1$, and as a result $i_2^j \geq 2$.

- If $e$ is PI at the end of mini-slot $T$ of sub-phase $j$ and $i_1^j$ denotes the mini-slot in which $e$ became PI, then, by Case 1, there exists a link $e_2^j \in I(e)$ which is PI or U at the end of mini-slot $T$ of sub-phase $j$, with the $i_1^j$-th coordinate of its control vector equal to 1 (or, equivalently, $e_2^j$ has sent a signal at mini-slot $i_1^j$). As we noted in Case 1, $e_2^j$ cannot be A, as otherwise $e$ would be inactive at the end of the sub-phase $j$ (due to the synchronization slot), and so will be Inactive at the end of control phase of $t$.

- If $e_2^j$ is U and $i_2^j$ is the last mini-slot of this sub-phase $j$ in which link $e_2^j$ has sent a signal, then, by Case 2 there exists a link $e_3^j \in I(e_2^j)$ which became PI at a mini-slot $i_3^j > i_2^j \geq i_1^j$, so that $i_3^j \geq 2$.

- If $e_2^j$ is PI, as the $i_1^j$-th coordinate of its control vector is equal to 1, it became PI at a mini-slot $i_2^j > i_1^j \geq 1$, and so $i_2^j \geq 2$. Furthermore, there exists by Case 1 a link $e_3^j \in I(e_2^j)$ which is PI or U or A at the end of mini-slot $T$ of sub-phase $j$.

- If $e_3^j$ is A, we stop with the sequence $e, e_2^j, e_3^j$. That is the case of sub-phase 2 in the example of Figure 9, with the sequence $e_1, e_2, e_3$ respectively PI, PI, A.

  Otherwise, as $e_3^j$ is PI (respectively U), there exists by Case 1 (respectively 2) a link $e_4^j \in I(e_3^j)$ which is A, PI or U. If $e_4^j$ is A, we stop with the sequence $e, e_2^j, e_3^j, e_4^j$; otherwise we continue the process with $e_4^j$ until we find an active link. In the sub-phase 1 in the example of Figure 9, we get the sequence $e_1, e_2, e_3, e_4, e_5$ respectively PI, U, PI, PI, A.

Summarizing the above construction, we have created at the end of mini-slot $T$ of any sub-phase $j$ a sequence of links $\mathbf{S}^j = (e = e_1^j, e_2^j, \ldots, e_{k_j}^j, e_{k_j+1}^j)$ which satisfies the following properties:

(i) link $e_m^j$ is U or PI for $m = 1, \ldots, k_j - 1$ ; link $e_{k_j}^j$ is PI and link $e_{k_j+1}^j$ is A.

(ii) link $e_{m+1}^j \in I(e_m^j)$ for $m = 1, \ldots, k_j$.

(iii) If $e_m^j$ is U and $i_m^j$ is the last mini-slot in which $e_m^j$ has sent a signal, then $e_{m+1}^j$ is PI and became PI at a mini-slot $i_{m+1}^j > i_m^j$.

(iv) If $e_m^j$ is PI and became PI at the mini-slot $i_m^j$, then $e_{m+1}^j$ has sent a signal in mini-slot $i_m^j$.

Properties (iii) and (iv) imply the following property:

(v) Suppose $e_m^j$ and $e_{m'}^j$ became PI respectively at mini-slots $i_m^j$ and $i_{m'}^j$; then if $m' > m$, $i_{m'}^j > i_m^j$. (Said otherwise, a link becomes PI at a mini-slot strictly greater than the mini-slot, where the preceding PI link in the sequence became PI).

It suffices to prove (v) for two consecutive PI links in the sequence. Suppose $e_m^j$ became PI at mini-slot $i_m^j$. If the next PI link is $e_{m+1}^j$, then by property (iv), as $e_{m+1}^j$ has sent a signal in

mini-slot $i^j_m$, it becomes PI at a mini-slot $i^j_{m+1} > i^j_m$. If the link $e^j_{m+1}$ is U, then $e^j_{m+2}$ is PI and is the first PI link after $e^j_m$; by property (iv) $e^j_{m+1}$ has sent a signal in mini-slot $i^j_m$ and so has sent its last signal in the mini-slot $i^j_{m+1} \geq i^j_m$. By property (iii) applied to $e^j_{m+1}$ we have $i^j_{m+2} > i^j_{m+1} \geq i^j_m$.

We now have all the ingredients to prove by induction on $S - j$ the following claim, which will imply that there exists a link which became PI in sub-phase 1 at a mini-slot at least $S + 1$, giving a contradiction when $S \geq T$.

**Claim 1** *At sub-phase $j$, there exists a sequence $\mathbf{S}^j$ as described above whose last PI link $e^j_{k_j}$ became PI at a mini-slot $i^j_{k_j} \geq 2 + S - j$.*

**Proof.** The claim is true for $j = S$ ($S - j = 0$). In fact, for any $j$, $i^j_{k_j} \geq 2$; indeed we have seen that either $e^2_j$ is PI and then $i(e^j_2) \geq 2$ or $e^2_j$ is U and then $i(e^j_3) \geq 2$. By property (v) it follows that the last link became PI at a mini-slot $\geq 2$.
Assume that the claim is true for $j$ and let us prove that it is still true for $(j-1)$ (induction on $S-j$). The induction assumption implies that there exists a sequence $\mathbf{S}^j = (e = e^j_1, e^j_2, \ldots, e^j_{k_j}, e^j_{k_j+1})$ whose last PI link $e^j_{k_j}$ became PI at a mini-slot $i^j_{k_j} \geq 2 + S - j$. Consider the same sequence at sub-phase $j - 1$. Link $e^j_{k_j+1}$ (which was A at the end of sub-phase $j$) cannot be A at the end of the sub-phase $j - 1$, otherwise during the synchronization slot of sub-phase $j - 1$, link $e^j_{k_j}$ would have become Inactive and would have not participated in sub-phase $j$. If link $e^j_{k_j+1}$ is PI at the end of mini-slot $T$ of sub-phase $j - 1$, then by property (v) it became PI at a mini-slot $i^{j-1}_{k_{j+1}} > i^j_{k_j} \geq 2 + S - j$. If link $e^j_{k_j+1}$ is U at the end of mini-slot $T$ of sub-phase $j - 1$, then by Case 2, there exists a link $e^{j-1}_{k_j+2}$ which became PI at a mini-slot $i^{j-1}_{k_{j+2}} > i^j_{k_j} \geq 2 + S - j$.
The construction previously described generates a sequence $\mathbf{S}^{j-1} = (e = e^j_1, e^j_2, \ldots, e^j_{k_j}, e^j_{k_j+1}, \ldots, e^{j-1}_{k_{j-1}+1})$ satisfying all the properties (i)-(iv), and which contains a link (either $e^j_{k_j+1}$ or $e^j_{k_j+2}$) which became PI at a slot $> 2 + S - j$. Therefore, by property (v), the last link of $\mathbf{S}^{j-1}$ also became PI at a mini-slot $i^{j-1}_{k_{j-1}} \geq 2 + S - (j - 1)$. ∎

We are now in position to conclude the proof of the proposition. By the claim above, there exists at sub-phase 1 a PI link $e^1_{k_1}$ which became PI at a mini-slot $i^1_{k_1} \geq S + 1$. However, in a sub-phase there are exactly $T$ mini-slots, which yields a contradiction when $S \geq T$. This completes the proof. In the example of Figure 9 one can see that we need $S \geq 3$; indeed, after sub-phase 2 there is no active link in $I(e_1) \cup \{e_1\}$. ∎

## 4.7 Setting the parameters of `AlgoLog`

Let us now address the communication overhead of `AlgoLog`, denoted by $\mathcal{O}$, defined as the number of mini-slots contained in the control phase of a slot. To obtain an admissible maximal schedule we have to choose, by Proposition 2, $S \geq T$. The best is to choose $S = T$. Therefore, there are $S$ ($= T$) sub-phases in the control phase plus $S - 1$ mini-slots of synchronization; furthermore each sub-phase is composed of $T$ mini-slots. Hence,

$$\mathcal{O} = T^2 + T - 1 \quad \text{with} \quad T = \log_2(\lceil CK + 1 \rceil). \tag{13}$$

Parameters $K$ and $C$ directly impact the overhead of `AlgoLog` since the overhead is an increasing function of the product $C \cdot K$, as shown in (13). We choose for $C$ the smallest possible value (see next paragraph). The choice of $K$ is more delicate as, on one hand, we want a small $K$ to reduce the overhead but, on the other hand, we want a large $K$ to ensure the finest possible discrimination between the ratios $q_t(e)/c(e)$, a key feature for ensuring that (non-interfering) links with the highest backlogs are scheduled first. One possible way to choose $K$ consists in deciding a priori of an upperbound for the overhead $\mathcal{O}$, which in turns implies an upper-bound $K_0$ on $K$. Then, if the ratio $q_t(e)/c(e)$ is bounded by some constant $B$, we can choose $K = K_0$ and $L = \frac{B(K_0-1)}{K_0}$. Finally, we divide the interval $(O, L)$ into $K_0 - 1$ intervals of the same length.

**Computation of** $C$. Recall that computing the minimum value of $C$ consists in determining the minimum number of colors needed to color the links in such a way that two interfering links have different colors. In general, that is an NP-complete problem (in fact it corresponds to a coloring of the vertices of the so called link-interference graph of $G$ whose vertices represent the links of $G$, two vertices being joined if the corresponding links interfere). In the $d$-interference model, there exists a simple greedy algorithm to do this in linear time using $2\Delta(G)^{d+1}$ different colors [11], where $\Delta(G)$ is the maximum degree in $G$. In the $d$-interference model, if $G$ is a path composed of at least $d+2$ links, then we can use only $d+2$ colors. In particular, if $d = 0$ and $G$ is a path then only two colors are needed.
A particular interesting case is when the binary interference model is the primary interference model (i.e. $d = 0$). In this case, the minimum value of $C$ is the edge chromatic number of $G$ (see [13]). Vizing's theorem [13] establishes that for any graph $G$, this number is either $\Delta(G)$ or $\Delta(G) + 1$. A constructive proof of Vizing's theorem can be found in [24]. There also exists a polynomial time algorithm to compute a valid coloring of the links by using at most $\Delta(G) + 1$ colors, therefore giving a $+1$ approximation. Finally, if $G$ is bipartite then the edge chromatic number of $G$ is exactly $\Delta(G)$; therefore when $d = 0$, we have $C = 2$ for any path and $C = 4$ for any 2-dimensional grid.

# 5 Stability of `AlgoLog`

In this section, we address the stability (defined below) of `AlgoLog` when each link is equipped with an infinite capacity buffer. Throughout, $\mathbb{P}$ and $\mathbb{E}$ denote the probability measure and the expectation operator associated with it, respectively.
To make the model as general as possible, we assume that arrivals to links are modulated by a discrete-time, aperiodic, irreducible, and homogeneous Markov chain $\mathbf{Z} := \{z_t,\, t \geq 1\}$, taking values in a countable and finite set $\mathcal{R} = \{1, \ldots, R\}$. Let $\pi = (\pi(1), \ldots, \pi(R))$ be the stationary distribution of $\mathbf{Z}$, and let $\mathbf{P} = [P(i,j)]_{i,j\in\mathcal{R}}$ be its probability transition matrix, with $Pi, j) := \mathbb{P}(z_{t+1} = j \mid z_t = i)$. For a later use, recall that $\pi = \pi\mathbf{P}$ and that $\pi(i) > 0$ for all $i \in \mathcal{R}$.

When $z_t = i$, the arrivals to links in slot $t$ are given by the stochastic sequence $\{A_t(e,i), e \in E\}$, with $A_t(e,i) \in \{0, 1, \ldots\}$ the number of arrivals to link $e$.

Let $q_t(e)$ be the number of pending messages in the buffer (of link) $e$ at the beginning of slot $t \geq 1$. We denote by $x_t(e)$ the number of messages transmitted on link $e$ in slot $t$, with $x_t(e) = 0$ if link $e$ is inactive in slot $t$ and $x_t(e) = \min(q_t(e), c(e))$ if link $e$ is active in slot $t$ (note that in the latter case $q_t(e) > 0$ as otherwise link $e$ cannot be active). The sequence $\{q_t(e),\, t \geq 1\}$ satisfies the following

recursion: for each $t \geq 1$ and each $e \in E$,

$$q_{t+1}(e) = q_t(e) + A_t(e, i) - x_t(e), \tag{14}$$

if $z_t = i$. Notice that the definition of $x_t(e)$ implies that new messages joining buffer $e$ in slot $t$ are not transmitted in this slot. We recall that the decision to activate or not a link in slot $t$ is made by `AlgoLog`, and this decision is a deterministic function of the virtual weight $(w_t(e), e \in E)$ at the beginning of slot $t$, which, by (9), only depends on the queue-lengths vector $\mathbf{q}_t := (q_t(e),\, e \in E)$ and on the coloring vector $\beta_t = (\beta_t(e), e \in E)$.

We assume that:

**A1** for every $e \in E$ and $i \in \mathcal{R}$, $\{A_t(e, i),\, t \geq 1\}$ is a sequence of independent and identically distributed (iid) random variables. Define

$$a(e, i) = \mathbb{E}[A_t(e, i)], \text{ for all } e \in E, i \in \mathcal{R}.$$

Moreover, for each $e$, the $|R|$ iid sequences $\{A_t(e, i),\, t \geq 1\}$, $i \in \mathcal{R}$, are mutually independent. Last, we assume that the sequences $\{A_t(e, i),\, e \in E, i \in \mathcal{R}\}$ and $\{A_{t'}(e, i),\, e \in E, i \in \mathcal{R}\}$ are independent for $t \neq t'$;

**A2** for each $t \geq 1$,

$$m_0 := \max_{e \in E, i \in \mathcal{R}} a(e, i) < \infty, \tag{15}$$

and

$$\sigma_0 := \max_{e \in E, f \in I(e) \cup \{e\}, i \in \mathcal{R}} \mathbb{E}[A_t(e, i) A_t(f, i)] < \infty. \tag{16}$$

Observe that **A1** does not rule out correlated arrivals at links in the same slot.

To show the versatility of the arrival processes defined above, let us specialize the Markov chain $\mathbf{Z}$ and assumption **A1** above as follows. Assume that $\mathcal{R} = \times_{e \in E} \mathcal{R}(e)$, and that $\mathbf{Z} = \{\mathbf{Z}(e), e \in E\}$ with $\mathbf{Z}(e) = \{z_t(e), t \geq 1\}$, $z_t(e) \in \mathcal{R}(e)$, are $|E|$ mutually independent, aperiodic irreducible, homogeneous, and discrete-time Markov chains. Assume further that $A_t(e, z_t(e))$ is the number of arrivals to link $e$ in slot $t$, and that $\{A_t(e, k), t \geq 1, k \in \mathcal{R}(e)\}$ are $|E|$ mutually iid sequences. In other words, we assume that arrivals on each link are modulated by independent Markov chains. Then, for each $e \in E$, $\{A_t(e, z_t(e)), t \geq 1\}$ is the so-called Discrete-time Batch Markov Arrival Process (D-BMAP) and these $|E|$ D-BMAPs are mutually independent. If, in addition, we assume that, for each $e \in E$, the set $\mathcal{R}(e)$ is a singleton, then these arrival processes are all independent iid sequences.

Under assumption **A1**, the process $\mathbf{Y} := \{\mathbf{y}_t := (\mathbf{q}_t, z_t),\, t \geq 2\}$ is a discrete-time, homogeneous, Markov chain on $\mathbb{N}^{|E|} \times \mathcal{R}$. We further assume that (see Remark 1):

**A3** the Markov chain $\mathbf{Y}$ is irreducible on $\mathbb{N}^{|E|} \times \mathcal{R}$. This will hold, in particular, if $\mathbb{P}(A_t(e, i) = k) > 0$ for all $k \geq 0$, $e \in E$, $i \in \mathcal{R}$.

`AlgoLog` is stable if the Markov chain $\mathbf{Y}$ is positive recurrent [23]. In other words, queues do not "build up" when `AlgoLog` is stable. Observe that the system is necessarily stable when all buffers

have finite capacity since in this case the state space is finite.

Let $a(e)$ be the expected number of arrivals in link $e$ in a slot. We have

$$a(e) = \sum_{i \in \mathcal{R}} a(e,i)\pi(i). \tag{17}$$

The following lemma will be used in the proof of Proposition 3 and, more precisely, in the definition of the Lyapounov function used to investigate the stability of the queue-lengths process $\{\mathbf{q}_t, t \geq 1\}$. For any row vector $\mathbf{v}$, we denote by $\mathbf{v}^{Tr}$ its transpose; let $\mathbf{1} = (1, \ldots, 1)^{Tr}$ be the unit vector of dimension $R$ and $\mathbf{I}$ the $R$-by-$R$ identity matrix.

**Lemma 3** *For each $e \in E$, the linear system of $R$ equations and $R$ unknowns, $u(e,1), \ldots, u(e,R)$, given by*

$$(\mathbf{I} - \mathbf{P})\mathbf{u}(e) = (a(e) - a(e,1), \ldots, a(e) - a(e,R))^{Tr}, \tag{18}$$

*has an infinite number of solution, with $\mathbf{u}(e) := (u(e,1), \ldots, u(e,R))^T$.*

**Proof.** Fix $e \in E$. Since $\mathbf{P}$ is an irreducible, stochastic matrix, the $R$-by-$R$ matrix $\mathbf{I} - \mathbf{P}$ has rank $R-1$ (its eigenvalue 0 is simple, since 1 is a simple eigenvalue of $\mathbf{P}$ from Perron-Frobenius theorem) and is therefore not invertible. As a result, either (18) has no solution or it has infinitely many solutions. Let us show that the latter holds by exhibiting a solution.
Introduce the $R$-by-$R$ matrix $\mathbf{M}$ whose first row is $(1, \ldots, 1)$ and $i$th row is the $i$th row of the matrix $\mathbf{I} - \mathbf{P}$, for $i = 2, \ldots, R$. Since $\mathbf{M}$ is invertible,[2] the system

$$\mathbf{M}\mathbf{x} = (\theta, a(e) - a(e,2), \ldots, a - a(e,R))^{Tr}, \tag{19}$$

has a unique solution, where $\theta$ is an arbitrary real number. Expanding (19), we obtain with $\mathbf{x} = (x_1, \ldots, x_R)^{Tr}$,

$$x_i - \sum_{j=1}^{R} P(i,j)x_j = a(e) - a(e,i), \quad i = 2, \ldots, R. \tag{20}$$

Multiplying now both sides of (20) by $\pi_i$ and summing up the resulting equations, gives

$$0 = \sum_{i=2}^{R} \pi_i x_i - \sum_{j=1}^{R} \left( \sum_{i=2}^{R} P(i,j)\pi(i) \right) x_j - a(e) \sum_{i=2}^{R} \pi(i) + \sum_{i=2}^{2} a(e,i)\pi(i). \tag{21}$$

Using the identity $\pi(j) = \sum_{i=1}^{R} P(i,j)\pi(i)$ for $j = 1, \ldots, R$, resulting from the invariant measure

[2] Assume that $\mathbf{M}^{-1}$ does not exist. Since rows $2, \ldots, R$ of $\mathbf{M}$ are linearly independent (as otherwise $\text{rank}(\mathbf{I} - \mathbf{P}) < R-1$), the non-invertibility of $\mathbf{M}$ implies that there exists a linear combination of rows $2, \ldots, R$ of $\mathbf{M}$ equals to the first row of $\mathbf{M}$, i.e. equals to the vector $(1, \ldots, 1)$. Namely, there exit $c_2, \ldots, c_R$ such that $1 = \sum_{i=2}^{R} c_i A(i,j)$ for $j = 1, \ldots, R$, where $\mathbf{A} = [A(i,j)] := \mathbf{I} - \mathbf{P}$. Summing over $j = 1, \ldots, R$ gives $R = \sum_{i=2}^{R} c_i \sum_{j=1}^{R} A(i,j) = 0$ since $\sum_{j=1}^{R} A(i,j) = 0$ for all $i$, which ends up with a contradiction. Therefore, $\mathbf{M}^{-1}$ exists.

equation $\pi = \pi\mathbf{P}$, and (17), we get from (21)

$$\begin{aligned} 0 &= \sum_{i=2}^{R} \pi_i x_i - \sum_{j=1}^{R} (\pi(j) - P(1,j)\pi(1)) x_j - a(1-\pi(1)) + a - a(e,1)\pi(1) \\ &= -\pi(1) \left( x_1 - \sum_{j=1}^{R} P(j,1) x_j - a + a(e,1) \right). \end{aligned}$$

Since $\pi(1) > 0$, the above implies that

$$x_1 - \sum_{j=1}^{R} P(j,1) x_j = a - a(e,1). \tag{22}$$

We then conclude from (20) and (22) that $\mathbf{x}$ is a solution of (18) (notice that we have actually exhibited infinitely many solutions since $\theta$ is arbitrary), which concludes the proof. ∎

Expanding (18) gives

$$u(e,i) = \sum_{j \in \mathcal{R}} P(i,j) u(e,j) + a(e) - a(e,i), \quad i \in \mathcal{R},\, e \in E. \tag{23}$$

The following sufficient stability condition holds:

**Proposition 3 (Sufficient stability condition for single-hop communications)** *Under assumptions* **A1**-**A3** *above, the arrival rate vector* $(a(e), e \in E)$ *stabilizes* `AlgoLog` *if*

$$\sum_{f \in I(e) \cup \{e\}} \frac{a(f)}{c(f)} < 1, \quad \forall e \in E, \tag{24}$$

*when* $c(e) = 1$ *for all* $e \in E$*, and if*

$$\sum_{f \in I(e) \cup \{e\}} \frac{a(f)}{c(f)} < \min(1, L), \quad \forall e \in E, \tag{25}$$

*when* $c(e) > 1$ *for at least one* $e \in E$*.*

Recall that $L \geq 0$ is a free parameter of `AlgoLog`- see Section 4.5. We see from (25) that selecting $L \geq 1$ will improve the stability condition when at least one $c(e)$ is equal to 1.

**Proof of Proposition 3.** Consider the Lyapounov function

$$V(\mathbf{y}) = \sum_{e \in E} \frac{q(e)}{c(e)} \sum_{f \in I(e) \cup \{e\}} \frac{q(f)}{c(f)} - 2 \sum_{e \in E} \frac{q(e)}{c(e)} \sum_{f \in I(e) \cup \{e\}} \frac{u(f,i)}{c(f)}, \tag{26}$$

$\mathbf{y} := (\mathbf{q}, i) \in \mathbb{N}^{|E|} \times \mathcal{R}$, where $\mathbf{q} := (q(e) \in E)$, and for each $f \in E$, $\{u(f,i), i \in \mathcal{R}\}$ is any solution of (18).

Notice that the first double summation in the right hand side of (26) is the Lyapounov function used in [39, Section II-A].

Let us first show that $V(\mathbf{y})$ is bounded from below, as required to apply Foster's criterion. From (26), we obtain

$$V(\mathbf{y}) \geq \left(\sum_{e\in E}\frac{q(e)}{c(e)}\right)^2 - 2\sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{u(f,i)}{c(f)} \geq \sum_{e\in E}\frac{q(e)}{c(e)}\left(\sum_{e\in E}\frac{q(e)}{c(e)} - u_0\right),$$

where $u_0 := 2\max_{j\in\mathcal{R}}\sum_{f\in E}\frac{|u(f,j)|}{c(f)}$. Since $u_0$ is finite (as, for each $f \in E$, $u(f,j)$ is finite for all $j \in \mathcal{R}$ and $c(f) > 0$, and the sets $E$ and $\mathcal{R}$ are finite), the above shows that $V(\mathbf{y}) \geq 0$ for all vectors $\mathbf{q}$ such that $\sum_{e\in E} q(e)/c(e) \geq u_0$. When $\sum_{e\in E} q(e)/c(e) < u_0$ then all $q(e)$ are bounded, and $V(\mathbf{y})$ is bounded from below. This shows that $\inf_{\mathbf{y}} V(\mathbf{y}) \geq -\infty$.

To simplify the notation, we introduce the shorthand $\mathbb{E}_{\mathbf{y}}[\cdot] = \mathbb{E}[\cdot \mid Y_t = \mathbf{y}]$. We will show that Foster's criterion [5, Theorem 1.1, p. 167], [23] applies to $V(\mathbf{y})$, i.e. there exist a finite set $\mathcal{S} \subset \mathbb{N}^{|E|} \times \mathcal{R}$ and a constant $\epsilon > 0$ such that $D(\mathbf{y}) := \mathbb{E}_{\mathbf{y}}[V(\mathbf{y}_{t+1}) - V(\mathbf{y})] < -\epsilon$ for all $\mathbf{y} \notin \mathcal{S}$ and $\sup_{\mathbf{y}\in\mathcal{S}} D(\mathbf{y}) < \infty$.

With $\mathbf{y}_{t+1} = (\mathbf{q}_{t+1}, z_{t+1})$ and $\mathbf{y}_t = (\mathbf{q}, i) = \mathbf{y}$, we have

$$\begin{aligned}
D(\mathbf{y}) = {} & \mathbb{E}_{\mathbf{y}}\left[\sum_{e\in E}\frac{q_{t+1}(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q_{t+1}(f)}{c(f)} - \sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q(f)}{c(f)}\right] \\
& -2\mathbb{E}_{\mathbf{y}}\left[\sum_{e\in E}\frac{q_{t+1}(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{u(f,z_{t+1})}{c(f)} - \sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{u(f,i)}{c(f)}\right] \\
= {} & \mathbb{E}_{\mathbf{y}}\Bigg[\sum_{e\in E}\frac{q_{t+1}(e)-q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q_{t+1}(f)-q(f)}{c(f)} - 2\sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q(f)}{c(f)} \\
& + \sum_{e\in E}\frac{q_{t+1}(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q(f)}{c(f)} + \sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q_{t+1}(f)}{c(f)}\Bigg] \\
& -2\mathbb{E}_{\mathbf{y}}\Bigg[\sum_{e\in E}\frac{q_{t+1}(e)-q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{u(f,z_{t+1})}{c(f)} \\
& + \sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{u(f,z_{t+1})-u(f,i)}{c(f)}\Bigg].
\end{aligned}$$

From the set equality $\{(e,f) \in E^2, f \in I(e)\cup\{e\}\} = \{(e,f) \in E^2, e \in I(f)\cup\{f\}\}$ which holds from the symmetric conflict assumption, we see that

$$\sum_{e\in E}\frac{q_{t+1}(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q(f)}{c(f)} = \sum_{e\in E}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\frac{q_{t+1}(f)}{c(f)},$$

which yields

$$D(\mathbf{y}) = D_1(\mathbf{y}) + D_2(\mathbf{y}), \tag{27}$$

with

$$D_1(\mathbf{y}) := \mathbb{E}_{\mathbf{y}}\left[\sum_{e\in E}\frac{q_{t+1}(e)-q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\left(\frac{q_{t+1}(f)-q(f)}{c(f)}-2\frac{u(f,z_{t+1})}{c(f)}\right)\right]$$
$$+2\mathbb{E}_{\mathbf{y}}\left[\sum_{e\in E:q(e)/c(e)\le L}\frac{q(e)}{c(e)}\sum_{f\in I(e)\cup\{e\}}\left(\frac{q_{t+1}(f)-q(f)}{c(f)}-\frac{u(f,z_{t+1})-u(f,i)}{c(f)}\right)\right]$$

and

$$\begin{aligned}D_2(\mathbf{y}) &:= 2\mathbb{E}_{\mathbf{y}}\Bigg[\sum_{e\in E:q(e)/c(e)>L}\frac{q(e)}{c(e)}\\ &\quad\times\sum_{f\in I(e)\cup\{e\}}\left(\frac{q_{t+1}(f)-q(f)}{c(f)}-\frac{u(f,z_{t+1})-u(f,i)}{c(f)}\right)\Bigg].\end{aligned}$$

We first show that there exists a constant $b_0$ such that $\Delta_1(\mathbf{y}) < b_0$. By using (14), (17), and $0 \le x_t(e) \le c(e)$ for all $t$, $e \in E$, we find

$$\begin{aligned}D_1(\mathbf{y}) &\le \sum_{e\in E}\sum_{f\in I(e)\cup\{e\}}\left(\frac{\mathbb{E}[A_t(e,i)A_t(f,i)]}{c(e)c(f)}+\frac{a(e,i)}{c(e)}+\frac{a(f,i)}{c(f)}+1\right)\\ &\quad+2\max_{f\in E,z\in\mathcal{R}}|u(f,i))|\sum_{e\in E}\left(\frac{a(e,i)}{c(e)}+1\right)\sum_{f\in E}\frac{1}{c(f)}\\ &\quad+2L\sum_{e\in E}\left(\sum_{f\in E}\left(\frac{a(f,i)}{c(f)}+1\right)+2\max_{f\in E,i\in\mathcal{R}}|u(f,i))|\sum_{f\in E}\frac{1}{c(f)}\right)\\ &\le |E|^2\left(\frac{\sigma_0}{c_0^2}+2\frac{m_0}{c_0}+1+2L\left(\frac{m_0}{c_0}+1\right)\right)\\ &\quad+2\frac{|E|^2}{c_0}\left(\frac{m_0}{c_0}+1+2L\right)\max_{f\in E,i\in\mathcal{R}}|u(f,i)| := b_0,\end{aligned}\tag{28}$$

where we have set $c_0 = \min_{e\in E} c(e) > 0$. The bound $b_0$ in (28) is finite as $|E|$, $L$ (defined in Section 4.5), $m_0$, and $\sigma_0$ are finite quantities (see (15) and (16)), and $\max_{f\in E,i\in\mathcal{R}}|u(f,i)| < \infty$ since $u(f,i)$ is finite for all $f \in E$, $i \in \mathcal{R}$, and that $E$ and $\mathcal{R}$ are finite sets.

Let us now focus on $D_2(\mathbf{y})$, $\mathbf{y} = (\mathbf{q}, i)$. From $\mathbb{E}_{\mathbf{y}}[u(f,z_{t+1})] = \sum_{j\in\mathcal{R}} P(i,j)u(f,j) = u(f,i) - a(f) +$

$a(f,i)$ (cf. (23) for the 2nd equality), $D_2(\mathbf{y})$ can be written as

$$\begin{aligned} D_2(\mathbf{y}) &= 2 \sum_{e\in E: q(e)/c(e)>L} \frac{q(e)}{c(e)} \Bigg( \sum_{f\in I(e)\cup\{e\}} \frac{1}{c(f)} \\ &\quad \times \left( a(f,i) - \sum_{j\in\mathcal{R}} P(i,j)u(f,j) + u(f,i) \right) - \sum_{f\in I(e)\cup\{e\}} \frac{\mathbb{E}[x_t(f)\,|\,\mathbf{q}_t=\mathbf{q}]}{c(f)} \Bigg) \\ &= 2 \sum_{e\in E: q(e)/c(e)>L} \frac{q(e)}{c(e)} \left( \sum_{f\in I(e)\cup\{e\}} \frac{a(f)}{c(f)} - \sum_{f\in I(e)\cup\{e\}} \frac{\mathbb{E}[x_t(f)\,|\,\mathbf{q}_t=\mathbf{q}]}{c(f)} \right). \end{aligned} \tag{29}$$

Define $\alpha = 1$ if $c(e) = 1$ for all $e \in E$, and $\alpha = \min(1, L)$ if $c(e) < 1$ for at least one $e \in E$. By (24)-(25) there exists $\epsilon' > 0$ such that $\sum_{f\in I(e)\cup\{e\}} a(f)/c(f) < \alpha - \epsilon'$. Therefore, by (29),

$$D_2(\mathbf{y}) \le 2 \sum_{e\in E: q(e)/c(e)>L} \frac{q(e)}{c(e)} \left( \alpha - \epsilon' - \sum_{f\in I(e)\cup\{e\}} \frac{E[x_t(f)\,|\,\mathbf{q}_t=\mathbf{q}]}{c(f)} \right). \tag{30}$$

Let us show that when $q(e)/c(e) > L$

$$\sum_{f\in I(e)\cup\{e\}} \frac{\mathbb{E}[x_t(f)\,|\,\mathbf{q}_t=\mathbf{q}]}{c(f)} \ge \alpha. \tag{31}$$

Assume first that $c(e) = 1$ for all $e \in E$. This implies that $q(e) \ge 1$ (recall $q(e)$ is an integer). Proposition 2 then says that at least one link in $I(e)\cup\{e\}$ is active, namely, there exists $f \in I(e)\cup\{e\}$ such that $q(f) \ge 1$ and $x_t(f) = \min(q(f), c(f)) = 1$, which proves (31) with $\alpha = 1$.
Assume now that $c(f) > 1$ for some $f \in E$. Because $q(e)/c(e) > L$, we may again invoke Proposition 2 to conclude that at least one link in $I(e)\cup\{e\}$ is active. If $e$ is active then $x_t(e) = \min(c(e), q(e)) \ge \min(c(e), c(e)L) = c(e)\min(1, L) = \alpha c(e)$ which proves (31). If $e$ is not active then there exists an active link in $I(e)$, say $f$, with virtual weight, $w_t(f)$, larger than the virtual weight of $e$. If $w_t(f) > w_t(e)$, then $q(f)/c(f) \in I_{K-1}$ by the definition of the virtual weights or, equivalently, $q(f)/c(f) > L$ and again (31) holds. Combining (30) and (31) yields

$$D_2(\mathbf{y}) \le -2\epsilon' \sum_{e\in E: q(e)/c(e)>L} \frac{q(e)}{c(e)}. \tag{32}$$

In summary, cf. (27), (28), and (32),

$$\begin{aligned} D(\mathbf{y}) &\le -2\epsilon' \sum_{e\in E: q(e)/c(e)>L} \frac{q(e)}{c(e)} + b_0 \\ &= -2\epsilon' \sum_{e\in E} \frac{q(e)}{c(e)} + 2\epsilon' \sum_{e\in E: q(e)/c(e)\le L} \frac{q(e)}{c(e)} + b_0 \\ &\le -2\epsilon' \sum_{e\in E} \frac{q(e)}{c(e)} + 2\epsilon' L|E| + b_0. \end{aligned} \tag{33}$$

Take $\epsilon > 0$ and let $M_0$ be any constant such that $M_0 > \frac{2\epsilon' L|E| + b_0 + \epsilon}{2\epsilon'}$. Let $\mathcal{S} = \{\mathbf{q} : \sum_{e \in E} q(e)/c(e) \leq M_0\} \times \mathcal{R}$; we conclude from (33) that $D(\mathbf{y}) \leq -\epsilon$, for all $\mathbf{y} \notin \mathcal{S}$ so that Foster's criterion applies with the finite set $\mathcal{S}$, since clearly $\sup_{\mathbf{y} \in S} D(\mathbf{y}) < \infty$ as the set $S$ is finite. This concludes the proof since irreducibility and Foster's criterion imply that the Markov chain $\mathbf{Y}$ is positive recurrent (i.e. stable) [5, Lemma 1.1, p. 168]. ∎

**Remark 1** *Due to the generality of the network topology, the interference sets, the statistical assumptions, and the behavior of* `AlgoLog` *it may be difficult to guarantee the irreducibility of the Markov chain* $\mathbf{Y}$*. If so, we can adopt the definition of stability in [37]. Theorem 3.1 in [37] ensures that Proposition 3 holds for this alternative definition.*

# 6 Numerical results

In this section, we investigate via simulations the performance of `AlgoLog`. More precisely, we study the time evolution of the queue lengths over a certain time-window (number of slots). The simulations have been carried out for path networks (Section 6.1), square grid networks (Section 6.2), and random networks (Section 6.3). For each topology we investigate both the evolution of the largest queue length and of the average queue length over tens of thousands of slots. In Section 6.1, for path networks and $d = 0$, we compare the total sum of the weights of the links of the matching found by `AlgoLog` in a given slot to the total sum of the weights of an optimal matching. In Section 6.2, we compare the performance of `AlgoLog` to that of the `Augmenting Paths Algorithm` for a grid topology. In Sections 6.1-6.2 results are obtained when the sufficient stability conditions found in Proposition 3 are not met by all links. We did this to show that by no means conditions in Proposition 3 are necessary, and to investigate the system behavior when (24)-(25) do not hold. In contrast, these conditions hold in Section 6.3.

## 6.1 Path network

In this section, we consider a path network and the primary node interference model ($d = 0$). We first study the ratio between the total sum of the weights (or queue lengths) of the links present in the matching computed by `AlgoLog` and the total sum of the weights of the links present in an optimal matching for centralized wireline network. Figure 10 displays this ratio for a path $G = (V, E)$ composed of $|E| = 50$ links for 1 000 tests. At each test, we assign a weight at every $e \in E$ as follows: $P(q_t(e) = i) = 1/51$ for $i = 0, 1, \ldots, 50$. Observe that, for each test, `AlgoLog` finds a matching yielding a ratio larger than 0.80 and, in most experiments, this ratio is larger than 0.95.

We now investigate the evolution of the queue length at each buffer. We consider a path composed of $|E| = 100$ links that we observe during 100 000 consecutive time slots. Since $d = 0$, we can take $C = 2$ (see discussion at the end of Section 4.7). With the initial coloring $(\chi_1(1), \chi_1(2), \chi_1(3), \ldots, \chi_1(100)) = (1, 2, 1, \ldots, 2)$, we have from (7) that $(\chi_t(1), \chi_t(2), \chi_t(3), \ldots, \chi_t(100))$ is equal to $(1, 2, 1, \ldots, 2)$ (respectively $(2, 1, 2 \ldots, 1)$) if t is an odd (respectively even) integer. We take $K = 1000$, $L = 999$, and the capacity of every link $e \in E$ is $c(e) = 18$. The arrival process is an iid process defined as follows: with probability $p(e)$, $c(e)$ messages join link $e$ in a slot and with probability $1 - p(e)$ no message joins link $e$, so that $a(e) = p(e)c(e)$. We assume that $a(e) = a_1$ (respectively $a(e) = a_2$) for $e = 1, 3, 5, \ldots, 99$ (respectively for $e = 2, 4, 6, \ldots, 100$). Figures 12(a), 12(b), and 12(c) display the largest queue length in each slot for pairs $(a_1, a_2) = (16, 1)$, $(a_1, a_2) = (12, 4)$, and $(a_1, a_2) = (8, 8)$,

respectively. Notice that links $2, \ldots, 99$ do not meet the sufficient stability conditions in Proposition 3; indeed, $\sum_{f \in I(e) \cup \{e\}} \frac{a(f)}{c(f)} = \frac{33}{18}$ when $e$ is an even number and is equal to 1 when $e$ is an odd number. On the other hand, condition (24) holds for links 1 and 100.
In these figures, we observe that the size of the largest queue is always smaller than 400, 180, and 140, respectively. It is also worth noting that the network is stable in Figures 12(a)-12(c) in the sense that the largest queue length does not increase as a function of $t$, despite the fact that the sufficient stability condition (25) is not satisfied for all $e \in E$, as discussed above.
We finally investigate the impact of the value of $K$ on the largest and average size of the queues for a path composed of 100 links, with $d = 0$, $C = 2$, $c(e) = 1$ for all $e \in E$. The arrival process is similar to that in Figures 12(a)-12(c) with $a(e) = 0.75$ (i.e. $p(e) = 0.75$) for $e = 1, 3, 5, \ldots, 99$ and $a(e) = 0.2$ (i.e. $p(e) = 0.2$) for $e = 2, 4, 6, \ldots, 100$. Figure 11 displays the largest and the average queue-length after 50 000 slots as a function of $K$ for $K = 2^i$ for $i = 0, 1, \ldots, 7$ and $L = K - 1$. The plots illustrate the performance-overhead tradeoff of $K$, a higher $K$ yielding a lower largest and average queue lengths; however, most of the gains come with relatively small values of $K$. Here too, the system appears to be stable although condition (24) in Proposition 3 is not satisfied as $\sum_{f \in I(e) \cup \{e\}} \frac{a(f)}{c(f)} = 1.15$ when $e$ is an odd number and $\sum_{f \in I(e) \cup \{e\}} \frac{a(f)}{c(f)} = 1.7$ when $e$ is an even number.

## 6.2 Square grid network

In this section, we compare the performance of `AlgoLog` to that of the `Augmenting Paths Algorithm` proposed in [7]. The simulations have been carried out for a square grid $G = (V, E)$ composed of $|V| = 121$ nodes and $|E| = 220$ links (see Figure 13). For every link $e \in E$, the capacity of $e$ is $c(e) = 1$.

Figure 14 displays the evolution of the largest and average queue-lengths over 100 000 time-slots under `AlgoLog` for $d = 1$, with $C = 4$, $K = 1\,000$, and $L = 999$. The arrival rates $(a(e), e \in E)$ are reported in Figure 13 (the arrival process is the same as in Section 6.1, namely, with probability $p(e)$, 1 message joins link $e$ and with probability $1 - p(e)$ no message joins link $e$ in a slot; here $p(e) = 0.7$ or $p(e) = 0.1$ as indicated in Figure 13). The minimum, maximum, and average values of $\sum_{f \in I(e) \cup \{e\}} \frac{a(f)}{c(f)}$ in (24) are given by 2.2, 5.9, and 5.412, respectively, thereby showing the sufficient condition (24) does not hold. However, the numerical results indicate that the network appears to be stable under theses arrival rates.

Figures 15(a)-15(c) display the evolution of the largest and average queue-lengths over $50\,000 - 200\,000$ time-slots under `AlgoLog` for $C = 4$, $K = 1\,000$, and $L = 999$, and under the `Augmenting Paths Algorithm`. As described in Section 3, the `Augmenting Paths Algorithm` has two input parameters: $k$ and $p$. Like in [7], the simulations have been carried out in Figures 15(a)-15(c) for $\{k = 2, p = 0.2\}$, $\{k = 3, p = 0.2\}$, and $\{k = 3, p = 0.1\}$, respectively. In Figure 15(a)-(c), $d = 0$ as the `Augmenting Paths Algorithm` is only defined for the primary interference model. In Figure 15(a) (respectively Figure 15(b), Figure 15(c)) the arrival rates are $(\lambda a(e), e \in E)$ with $\lambda = 0.90$ (respectively $\lambda = 0.95$, $\lambda = 0.97$), where $(a(e), e \in E)$ are given in Figure 14. The minimum, maximum, and average values of $\sum_{f \in I(e) \cup \{e\}} \frac{\lambda a(f)}{c(f)}$ in (24) are given by $0.4\lambda$, $1.9\lambda$, and $1.771\lambda$, respectively, Fso that (24) does not hold for $\lambda = 0.90, 0.95, 0.97$. Again, it is worth noting from Figures 15(a)-15(c) that these arrival rates stabilize the grid network although (24) is violated. We observe that the flexibility

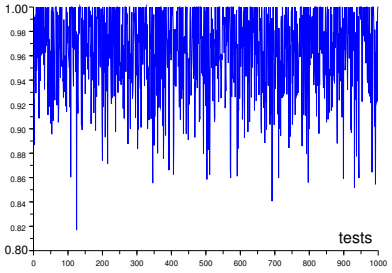



Figure 10: Ratio between the sum of the weights of the links of the matching found by `AlgoLog` and the sum of the weights of the links of an optimal matching for a path composed of 50 links for 1 000 tests, $d = 0$, and $P(q_t(e) = i) = 1/51$ for $0 \leq i \leq 50$, $e \in E$.

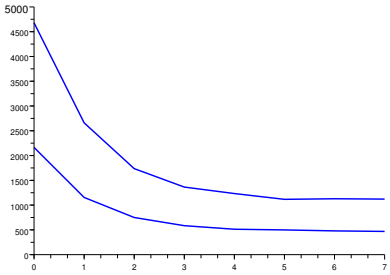
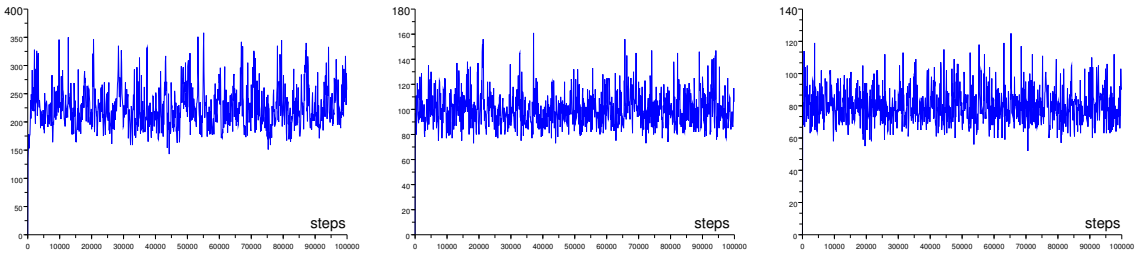



Figure 11: Largest and average queue length as function of $K = 2^i$ for $i = 0, 1, \ldots, 7$ after 10 000 slots for a path composed of 100 links for $d = 0$, $C = 2$, $L = K - 1$, $c(e) = 1$, $a(e) = 0.75$ for odd links and $a(e) = 0.2$ for even links.




(a) $(a_1, a_2) = (16, 1)$. (b) $(a_1, a_2) = (12, 4)$. (c) $(a_1, a_2) = (8, 8)$.

Figure 12: Evolution of the largest queue-length for a path network composed of 100 links for `AlgoLog` during 50 000 slots for $d = 0$, $C = 2$, $K = 1\,000$, $L = 999$, $c(e) = 18$, $a(e) = a_1$ for odd links and $a(e) = a_2$ for even links.

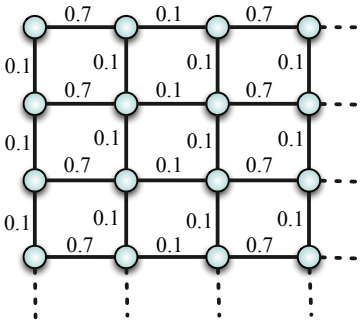



Figure 13: Grid network. Values near the links give $a(e)$ for all $e \in E$.

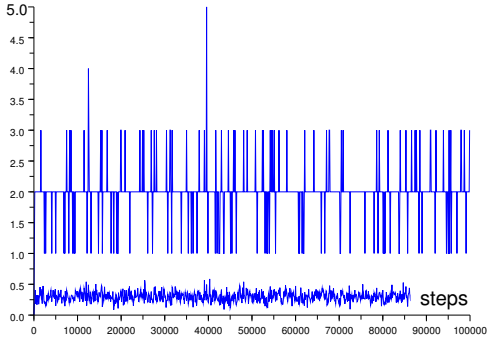
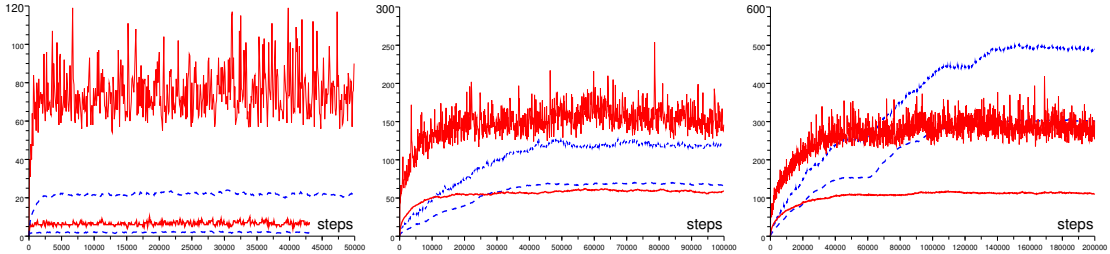



Figure 14: Evolution of the largest and average queue-lengths for the grid network in Fig. 13 under `AlgoLog` with $d = 1$, $C = 4$, $K = 1\ 000$, $L = 999$.




(a) $\lambda = 0.90$ (b) $\lambda = 0.95$ (c) $\lambda = 0.97$

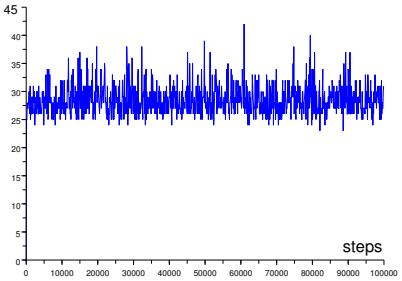
Figure 15: Evolution of the largest and average queue-lengths for a grid network composed of 121 nodes and 200 links under either the `Augmenting Paths Algorithm` (with $k = 2$, $p = 0.2$) or `AlgoLog` (with $d = 0$, $C = 4$, $K = 1\ 000$, $L = 999$) with $(c(e) = 1, e \in E)$ and $(a(e), e \in E)$ given in Fig. 13. Arrival rates are $(\lambda a(e), e \in E)$ with $\lambda = 0.90, 0.95, 0.97$ in Fig. 15(a)-15(c), respectively, where $(a(e), e \in E)$ are given in Fig. 13. Red straight lines represent `Augmenting Paths Algorithm` and blue dotted lines represent `AlgoLog`.

of `AlgoLog` - in terms of working for arbitrary values of $d$ as opposed to the `Augmenting Paths Algorithm` which only works for $d = 0$ - comes at a price for high arrival rates, as can be seen in Figure 15(c), where the `Augmenting Paths Algorithm` performs better than `AlgoLog`.

### 6.3 Random network

To conclude, we apply `AlgoLog` to a random network, generated as follows: given a square grid $G' = (V, E')$, we first delete link $e$ with probability $p$ for all $e \in E$, and then add a link with probability $q$ between any pair of nodes located within a given Euclidean distance. This process yields a new (random) transmission graph $G = (V, E)$. Figure 8 shows an instance of a random graph composed of $|V| = 48$ nodes and $|E| = 91$ links.

The figure on the right displays the evolution of the largest queue-length under `AlgoLog` for a random graph composed of 200 nodes and 424 links. The parameters of `AlgoLog` are $d = 0$, $C = 20$, $K = 10\ 000$, $L = 9\ 999$. For every $e \in E$, $c(e) = 20$. The arrivals per slot on link $e \in E$ are iid and follow a uniform distribution in $[0, \kappa(e)]$, with $\kappa(e) \leq 40$. The parameters $(\kappa(e), e \in E)$ have been chosen so that the sufficient stability conditions (25) hold.




## 7 AlgoLog: from link scheduling to node scheduling

Up to now, we have focused on the link version of `AlgoLog` - referred to as the link model - following in that the models used in the literature. On a theoretical point of view, the link model is natural, as ultimately arrivals and buffers are positioned on the links. In practice, however, things are different as all decisions are made by the nodes, and not by the links which are immaterial entities. There are many ways to go from a link model to a node model, depending on the technology that is used. As already noted in Section 1, most of the node models use a symmetric directed transmission graph, so that links are directed. In the following, a directed link is called an arc. In the control phase of `AlgoLog`, a link sends a (control) signal heard by all links interfering with $e$, namely, by all links in $I(e)$ (cf. line 6 of the pseudo-code of `AlgoLog` in Table 2); decisions are taken in each sub-phase according to that. In a node model, nodes are in charge of the sending and hearing processes. We could design some new algorithm(s) adapted to the node model, but this would imply to redo the entire analysis. Instead, we prefer to indicate how to emulate the link model, so that all precedings results can be reused. To this end, we will emulate a sub-phase of `AlgoLog` (cf. lines 5-10 in Table 2). However, this has to be done in a distributed manner and with the smallest possible signal overhead.
We will show how to do it with for the version of `AlgoLog` where links are directed (arcs), and only in the primary node interference model, where two arcs interfere if they share a common node.

### 7.1 Emulating lines 5-6 of AlgoLog

In the following, the wording 'an arc $e$ sending' stands for 'an arc $e$ sending a signal' (which happens when $s(e) = U$ and $\nu_{t,e}(i) = 1$). To decide if an arc $e$ sending becomes A (Active) or stays U

(undetermined) (and similarly to decide if an arc $e = (u, v)$ not sending becomes PI (potentially inactive) or stays U), node $u$ has to know if there is another arc $e'$ sending and interfering with $e$. Notice that, necessarily, node $u$ knows if arc $e' = (u, v')$ is sending. However, $u$ does not know if there are other arcs interfering with it. Therefore, order to apply Algolog at the node level, there are two problems to address:

**Problem 1:** How can node $u$ know that there is an arc $e' = (w, u)$ sending?

**Problem 2:** How can node $u$ know that for each arc $e = (u, v)$, there exists some arc $e'$ sending with end node $v$, either of the form $e' = (v, w)$ or $e' = (w, v)$?

The node version of `AlgoLog` is denoted `AlgoLogNodes`.

We first address **Problem 1**. If $u$ does not hear any signal from its neighbors, then clearly there is no sending arc $e' = (w, u)$. On the other hand, if $u$ hears a signal from some node $w$, then it needs to determine whether or not a signal is aimed to it.
From now on, we assume (**Hypothesis H**) that if, in the control phase, node $u$ sends a signal to a neighboring node $v$ then $v$ knows that $u$ has sent a signal to it.

We now show how, under **H**, one can emulate, in the primary node model, a mini-slot $i$ of sending on the arc $e = (u, v)$ by dividing it in two sub-mini-slots, denoted by $i_1$ and $i_2$, that is how one can emulate the behavior of lines 5-6 of `AlgoLog`. As a result, the overhead will only be multiplied by two. During sub-mini-slot $i_1$ (of $i$) of `AlgoLogNodes` the following happens:

**Sub-mini-slot $i_1$:** For each arc $e = (u, v)$ such that $s(e) = U$ and $\nu_{t,e}(i) = 1$, then node $u$ sends a signal in the direction of node $v$, and $v$ knows it thanks to **H**.

Within the sub-mini-slot $i_1$, if node $u$ detects a signal from some node $w$, then it knows that arc $e' = (w, u)$ is sending and, therefore, interfering with all the arcs with origin $u$.

It remains to deal with interference of an arc $e = (u, v)$ with some arc $e'$ due to node $v$, that is Problem 2. To answer the problem we design sub-mini-slot $i_2$ as follows:

**Sub-mini-slot $i_2$:**

- Case 1: If in sub-mini-slot $i_1$ node $v$ has sent a signal, then $v$ sends a signal (busy) to all of its neighbors;
- Case 2: If in sub-mini-slot $i_1$ node $v$ has not sent a signal, but has heard a signal from at least two of its neighbors, then $v$ sends a signal (busy) to all of its neighbors;
- Case 3: If in sub-mini-slot $i_1$ node $v$ has not sent any signal and has heard a signal from exactly one of its neighbors (say node $u$), then $v$ sends a signal (busy) to all of its neighbors, except to node $u$.

## 7.2 Emulating lines 7-10 of `AlgoLog`

After the two sub-mini-slots $i_1$ and $i_2$, a node $u$ can determine for each arc $e = (u, v)$ if there is an arc sending and interfering with $e$. That happens exactly in the following three cases:

- Case a: In sub-mini-slot $i_1$, $u$ sends a signal to some node $v'$;
- Case b: In sub-mini-slot $i_1$, node $u$ hears a signal from some node $w$;
- Case c: In sub-mini-slot $i_2$, node $u$ hears a signal (busy) from node $v$.

Indeed, in these three cases there is an arc $e'$ sending and interfering with $e = (u, v)$, namely, in Case a $e' = (u, v')$, in Case b $e' = (w, u)$, and in Case c $e' = (v, w)$ if the signal busy was sent in sub-mini-slot $i_2$ due to Case 1 or $e' = (w, v)$ if the signal busy was sent in sub-mini-slot $i_2$ due to Case 2 or Case 3. Furthermore, if we are not in one of Cases 1-3, then there is no arc interfering with $e$.

Depending on what happens in sub-mini-slots $i_1$ and $i_2$, a node $u$ can know when there is an interference between an arc $e = (u, v)$ and another arc $e'$, so that it is able to make decisions which exactly emulate decisions taken in the link model (cf. lines 7-10 in Figure 2). More precisely, for an arc $e = (u, v)$ with origin $u$ and such that $s(e) = U$,

- if $e$ is sending (i.e. $\nu_t(f, i) = 1$) and if there is no arc $e'$ sending and interfering with $e$, then node $u$ changes the state of $e$ from $U$ to $A$;
- if $e$ is not sending (i.e. $\nu_t(f, i) = 0$) and if there is an arc $e'$ sending and interfering with $e$, then node $u$ changes the state of $e$ from $U$ to $PI$.

As an illustration of the above, consider the oriented grid displayed in Figure 16. Here, we assume that at the beginning of sub-mini-slot $i$, all the arcs are undetermined and five of them satisfy $\nu_{t,e}(i) = 1$ namely $(u_1, u_2)$, $(u_4, u_7)$, $(u_5, u_4)$, $(u_8, u_7)$, and $(u_8, u_9)$ and, so, are sending. In sub-mini-slot $i_1$ (Figure 16(b)), $u_1$ sends a signal to $u_2$ (respectively $u_4$ to $u_7$, $u_5$ to $u_4$ and $u_8$ both to $u_7$ and $u_9$). In sub-mini-slot $i_2$ (Figure 16(c)), according to Case 1, $u_1$, $u_4$, $u_5$, $u_8$ being involved in at least one transmission send a signal (busy) to all of their neighbors; according to Case 2, $u_7$ sends a signal (busy) to all of its neighbors; according to Case 3, $u_2$ (resp $u_9$) sends a signal to all of its neighbors except $u_1$ (resp $u_8$); $u_3$ and $u_6$ do nothing. Cases 1-3 correspond to arcs labeled busy $i \in \{1, 2, 3\}$ in Figure 16(c).

We now describe the decisions taken by the nodes after the sub-mini-slots $i_1$ and $i_2$, for the oriented grid displayed in Figure 16(d). The state of an arc is represented by a dash line for PI (Potentially Inactive), a solid line for U (Undetermined), and Blue bold for A (Active). For each arc $e = (u, v)$, the label 'case $x$' with $x \in \{a, b, c\}$, indicates for what reason node $u$ knows that there is another arc arc sending and interfering with $e$.

For sake of simplicity, let us only focus on nodes $u_1$, $u_2$, $u_3$, and $u_5$ in Figure 16(d). Start with node $u_1$. For the arc $(u_1, u_4)$, by Case a, there is an interference with the sending arc $(u_1, u_2)$. Therefore, node $u_1$ changes the state of the arc $(u_1, u_4)$ from $U$ to $PI$. For the arc $(u_1, u_2)$, there is no arc sending and interfering with it. Therefore, node $u_1$ changes the state of the arc $(u_1, u_2)$ from $U$ to $A$.
Consider node $u_2$. In sub mini-slot $i_1$, it hears a signal from $u_1$ and so it knows, by Case b, that the arc $(u_1, u_2)$ interferes with the three arcs $(u_2, u_1)$ $(u_2, u_3)$, and $(u_2, u_5)$; as a result, $u_2$ changes the state of these three arcs from $U$ to $PI$.
Consider now node $u_5$. For the arcs $(u_5, u_2)$, $(u_5, u_6)$, and $(u_5, u_8)$, node $u_5$ knows, by Case a, that the sending arc $(u_5, u_4)$ interferes with them, and so node $u_5$ changes its state from $U$ to $PI$. For the arc $(u_5, u_4)$, node $u_5$ has heard a signal busy from $u_4$ in mini-slot $i_2$, (Case 1 as $u_4$ is sending in min-slot $i_1$ to $u_7$); therefore, $u_5$ does nothing for this arc (i.e. the state of $(u_5, u_4)$ remains at $U$).

Finally, consider node $u_3$. This node hears a signal from $u_2$ in sub-mini-slot $i_2$ (Case 3 as $u_2$ heard in sub-min-slot $i_1$ a signal from $u_1$). Therefore, by Case c, $u_3$ knows that there is a sending arc interfering in $u_2$ with the arc $(u_3, u_2)$; hence, it changes the state of $(u_3, u_2)$ from $U$ to $PI$. For the arc $(u_3, u_6)$, there is no arc sending interfering with it. Therefore, $u_3$ does nothing for this arc (i.e. the state of $(u_3, u_6)$ remains at $U$).

### 7.3 Emulating lines 11-16 of `AlgoLog`

We can also emulate the synchronization slot (lines 11-12 of `AlgoLog` in Figure 2) with two sub-mini-slots identical to the sub-mini-slots $i_1$ and $i_2$. Each node $u$ will know, exactly as above, for an arc $e$ if there is an arc $e'$ sending and interfering with it (Cases a,b,c) and, therefore, can decide to change the state of the arcs $PI$ to $U$ if there is no interference (emulating lines 13-14 of `AlgoLog`) or to $I$ if there is an interference (emulating lines 15-16 of `AlgoLog`).

Let us illustrate this process on Figure 16(e). In the first sub-mini-slot of synchronization, $u_1$ (which knows that $(u_1, u_2)$ is active) sends a signal of synchronization to $u_2$. Then, in the second sub-mini-slot of synchronization, $u_1$ sends a signal busy (Case 1) and $u_2$ sends a signal busy (Case 3). Note that Case 2 does not appear in this example. Then, $u_1$ will change the state of $(u_1, u_4)$ from $PI$ to $I$ as there is an interference with $(u_1, u_2)$ (Case a). Node $u_2$ will change the state of $(u_2, u_1)$ from $PI$ to $I$, and $u_4$ will change the state of $(u_4, u_1)$ from $PI$ to $I$ as for these two arcs there is an interference with $(u_1, u_2)$ (Case b). Finally, due to the interference with $(u_1, u_2)$ (Case c), $u_2$ will change the state of $(u_2, u_3)$ and $(u_2, u_5)$ from $PI$ to $I$ and $u_3$ and $u_5$ will both change the state of $(u_3, u_2)$ and $(u_5, u_2)$ from $PI$ to $I$. All the other arcs of the figure which are $PI$ will be changed to $U$.

Results obtained in Sections 7.1-7.3 are collected in the following proposition:

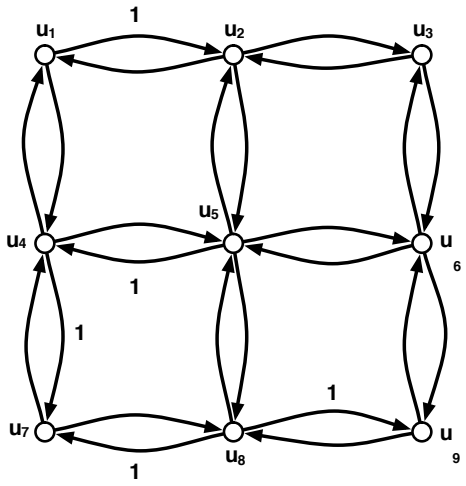
**Proposition 4** *Under the primary node interference model and under Hypothesis* **H**, *`AlgoLogNodes` emulates the decisions made by `AlgoLog` (lines 5-16 in Figure 2).*

## 8 Conclusion

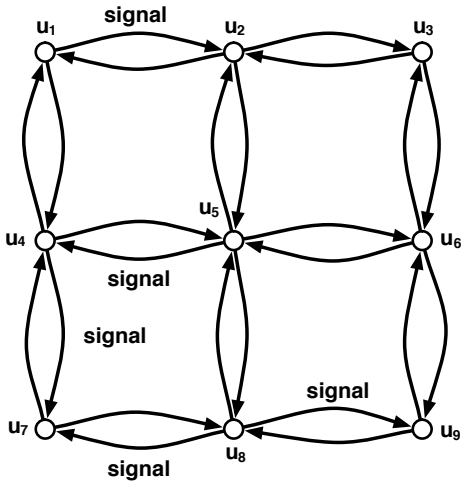
We have devised the first (to the best of our knowledge) distributed algorithm - referred to as `AlgoLog`- for the transmission scheduling problem in wireless networks with constant overhead and arbitrary binary interference. We have proved that the set of active links at each slot is maximal. We have proposed sufficient stability conditions and have investigated performance of `AlgoLog` via simulations. We have also shown how the algorithm can be emulated at the node level in the primary node interference model. In terms of future work, it would be interesting to design from scratch an algorithm at the node level and to analyze it. Also, it would be interesting to establish a better stability condition and, ideally, to characterize the entire stability region of `AlgoLog`.

## 9 Acknowledgements

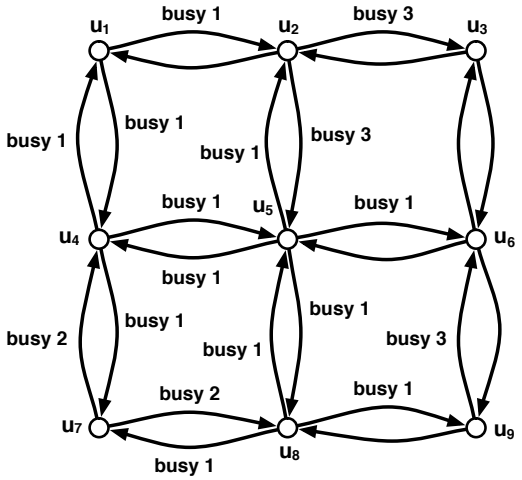
The authors are grateful to Alain Jean-Marie for useful discussions during the course of this work. The authors thank the referees for their helpful remarks and suggestions which have greatly contributed to improving the paper.



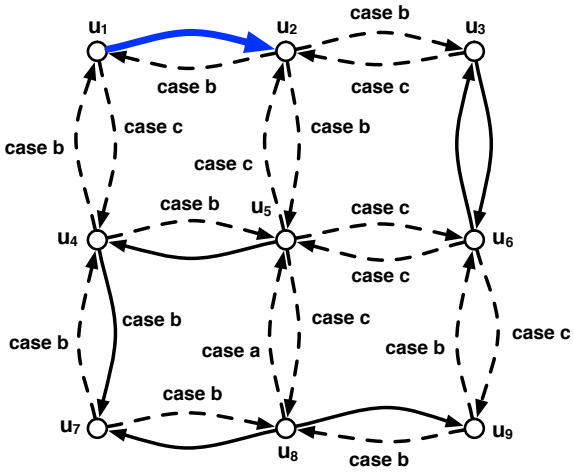

(a) Symmetric oriented graph
and first bit of the control vector



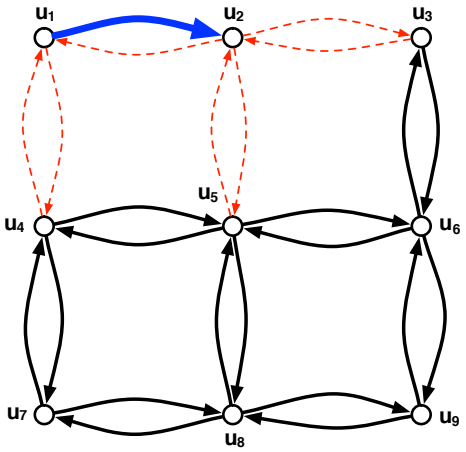

(b) Sub-mini-slot $i_1$




(c) Sub-mini-slot $i_2$




(d) Decisions after sub-mini-slot $i_2$




(e) After sub-mini-slot of synchronisation



Figure 16: Two sub-mini-slots and a sub-mini-slot of synchronisation of `AlgoLogNodes` (that emulates `AlgoLog`) for an oriented grid network. Active links are indicated with thick bold blue lines, inactive links with thin dashed red lines, undetermined links with thin black lines, and potentially inactive links with dashed black lines.

# References


[1] F. Baccelli and B. Błaszczyszyn. Stochastic Geometry and Wireless Networks Volume II: Applications. Now Publishers, 2010.

[2] H. Balakrishnan, C.L. Barrett, V.S.A. Kumar, M.V. Marathe, and S. Thite. The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks. IEEE, J. Selected Areas in Communication, 22(6):1069–1079, 2004.

[3] J-C. Bermond, D. Mazauric, V. Misra, and P. Nain. A distributed scheduling algorithm for wireless networks with constant overhead and arbitrary binary interference. In ACM SIGMETRICS Performance Evaluation Review, volume 38, pages 345–346, New York, NY, USA, June 14-18 2010.

[4] V. Bonifaci, R. Klasing, P. Korteweg, L. Stougie, and A. Marchetti-Spaccamela. Data gathering in wireless networks. In Arie Koster and Xavier Muñoz, editors, Graphs and Algorithms in Communication Networks, Studies in Broadband, Optical, Wireless and Ad Hoc Networks, pages 357–377. Springer-Verlag, 2010.

[5] P. Brémaud. Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues, volume 31 of Texts in Applied Mathematics. Springer, 1999.

[6] A. Brzezinski, G. Zussman, and E. Modiano. Enabling distributed throughput maximization in wireless mesh networks: a partitioning approach. In Proc. ACM MobiCom, pages 26–37, Los Angeles, CA, USA, September 24-29, 2006.

[7] L.X. Bui, S. Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. IEEE/ACM Transactions on Networking, 17(5):1467–1480, 2009.

[8] K. Cameron. Induced matchings. Discrete Applied Mathematics, 24(1-3):97–102, 1989.

[9] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar. Throughput and fairness guarantees through maximal scheduling in wireless networks. IEEE Transactions on Information Theory, 54(2):572–594, 2008.

[10] H. Chen, X. Xie, and H. Wu. A queue-aware scheduling algorithm for multihop relay wireless cellular networks. In Proc. IEEE Mobile WiMAX Symposium, pages 63–68, Napa Valley, CA, USA, July 9-10, 2009.

[11] R. Diestel. Graph Theory (Graduate Texts in Mathematics), 1997.

[12] A. Eryilmaz, O. Asuman, and E. Modiano. Polynomial complexity algorithms for full utilization of multi-hop wireless networks. In Proc. IEEE INFOCOM, pages 499–507, Anchorage, AK, USA, May 6-12, 2007.

[13] S. Fiorini and R.J. Wilson. Edge-colourings of graphs, volume 16 of Research Notes in Mathematics. Pitman, 1977.

[14] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. IEEE/ACM Transactions on Networking, 17(6):1846–1859, December 2009.

[15] P. Gupta and P.R. Kumar. The capacity of wireless networks. IEEE Transactions on Information Theory, 46(2):388–404, March 2000.

[16] L. Jiang, D. Shah, J. Shin, and J. Walrand. Distributed random access algorithm: scheduling and congestion control. IEEE Transactions on Information Theory, 56(12):6182–6207, December 2010.

[17] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in a wireless networks. In Proc. 46th Allerton Conference on Communication, Control, and Computing, Urbana-Champaign, IL, USA, 2008.

[18] L. Jiang and J. Walrand. Approaching throughput-optimality in distributed CSMA scheduling algorithms with collisions. IEEE/ACM Transactions on Networking, 19(3):816–829, 2011.

[19] R. Klasing, N. Morales, and S. Pérennes. On the complexity of bandwidth allocation in radio networks. Theoretical Computer Science, 406(3):225 – 239, 2008.

[20] V.S.A. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet-scheduling in wireless ad-hoc networks. In Proc. ACM-SIAM SODA, pages 1021–1030, New Orleans, LO, USA, January 11-13, 2004.

[21] L. Lovász and M.D. Plummer. Matching Theory, volume 29 of Annals of Discrete Mathematics. North-Holland, 1986.

[22] R. Mazumdar, G. Sharma, and N. Shroff. Maximum weighted matching with interference constraints. in Proc. FAWN, Pisa, Italy, March 2006.

[23] S.P. Meyn and R.L. Tweedie. Markov Chains and Stochastic Stability. Springer-Verlag, 1993.

[24] J. Misra and D. Gries. A constructive proof of vizing's theorem. Information Processing Letters, 41, 1992.

[25] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In Proc. ACM SIGMETRICS - IFIP PERFORMANCE, volume 34, pages 27–38, Saint Malo, France, June 26-30, 2006.

[26] S. Rajagopalan, D. Shah, and J. Shin. Network adiabatic theorem: an efficient randomized protocol for contention resolution. In Proc. ACM SIGMETRICS - IFIP PERFORMANCE, volume 37, pages 133–144, Seattle, WA, USA, June 15-19, 2009.

[27] S. Sanghavi, L. Bui, and R. Srikant. Distributed link scheduling with constant overhead. In Proc. ACM SIGMETRICS, pages 313–324, San Diego, CA, USA, June 12-16, 2007.

[28] D. Shad and D. Wischik. Log-weight scheduling in switched networks. Queueing Systems (QUESTA), 71(1–2):97–136, June 2012.

[29] D. Shah and J. Shin. Randomized scheduling algorithm for queueing networks. Ann. Appl. Probab., 22(1):128–171, February 2012.

[30] D. Shah, J. Shin, and P. Tetali. Medium access using queues. In Proc. IEEE 52nd Annual Symposium on Foundations of Computer Sciences (FOCS), Palm Springs, CA, USA, October 2011.

[31] D. Shah and D. Wischik. Switched networks with maximum weight policies: Fluid approximation and multiplicative state space collapse. Ann. Appl. Probab., 22(1):70–127, February 2012.

[32] H. Shokri-Ghadikolaei, C. Fischione, and E. Modiano. On the accuracy of interference models in wireless communications. In ICC 2016, IEEE international Conference on Communications, pages 1–6, 05 2016.

[33] H. Shokri-Ghadikolaei, C. Fischione, and E. Modiano. Interference model similarity index and its applications to mmwave networks: Extended version. IEEE Transactions on Wireless Communications, 10 2017.

[34] F. Simatos, N. Bouman, and S. Borst. Lingering issues in distributed scheduling. In Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '13, pages 141–152, New York, NY, USA, 2013. ACM.

[35] L.J. Stockmeyer and V.V. Vazirani. NP-completeness of some generalizations of the maximum matching problem. Information Processing Letters, 15(1):14–19, 1982.

[36] L. Tassiulas. Scheduling and performance limits of networks with constantly changing topology. IEEE Transactions on Information Theory, 43(3):1067–1073, 1997.

[37] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Transactions on Automatic Control, 37(12), December 1992.

[38] P-J. Wan. Multiflows in multihop wireless networks. In Proc. ACM MobiHoc, pages 85–94, New Orleans, LO, USA, May 18-21, 2009.

[39] X. Wu, R. Srikant, and J.R. Perkins. Queue length stability of maximal greeedy schedules in wireless networks. In Proc. of Information Theory and Applications Inaugural Workshop, San Diego, CA, USA, February 6-10, 2006.