# Low Delay Scheduling in Wireless Network

Kyomin Jung
Mathematics, MIT
kmjung@mit.edu

Devavrat Shah
EECS, MIT
devavrat@mit.edu

***Abstract*— In a wireless network, a sophisticated algorithm is required to schedule simultaneous wireless transmissions while satisfying interference constraint that *two neighboring nodes can not transmit simultaneously*. The scheduling algorithm need to be excellent in performance while being simple and distributed[1] so as to be implementable. The result of Tassiulas and Ephremides (1992) imply that the algorithm, scheduling transmissions of nodes in the 'maximum weight[2] independent set' (MWIS) of network graph, is throughput optimal. However, algorithmically the problem of finding MWIS is known to be NP-hard and hard to approximate. This raises the following questions: is it even possible to obtain throughput optimal simple, distributed scheduling algorithm? if yes, is it possible to minimize delay of such an algorithm?**

**Motivated by these questions, we first provide a distributed throughput optimal algorithm for any network topology. However, this algorithm may induce exponentially large delay. To overcome this, we present an order optimal delay algorithm for any non-expanding[3] network topology. Networks deployed in geographic area, like wireless networks, are likely to be of this type. Our algorithm is based on a novel distributed graph partitioning scheme which may be of interest in its own right. Our algorithm for non-expanding graph takes $O(n)$ total message exchanges or $O(1)$ message exchanges per node to compute a schedule.**

## I. Introduction

Wireless networks are becoming architecture of choice in ad-hoc networks and metro-area networks or mesh-networks. The tasks of resource allocation and scheduling are essential for good network utilization. Wireless medium being multi-access makes algorithm design for such network intrinsically different and more challenging than its wireline counterpart. Further, wireless architecture requires that algorithm be distributed and simple.

Despite these challenges, there has been an exciting recent progress based on optimization frame-work to characterize good resource allocation algorithm that combine resource allocation and scheduling (see [1], for example). However, these solutions either assume availability of good scheduling algorithm or use of imperfect scheduling (which will lead to poor performance). In this paper, we are interested in designing simple to implement, distributed and high-performance scheduling algorithms.

[1] In this paper, by distributed we mean that algorithm operating at nodes of the network can only utilize local topological information.

[2] Weight is an appropriate function of queue-sizes and possibly other network parameters.

[3] See section IV for precise definition of the non-expanding graph.

### A. Scheduling in wireless network

We consider an abstract model of wireless network given by graph $G = (V, E)$ with $|V| = n$ wireless nodes and edges represented by $E$. We consider the classical interference model for multi-access channel which imposes the constraint that two neighboring nodes can not transmit simultaneously. Subsequently, simultaneously transmitting nodes must correspond to *independent set* of $G$. When nodes are given weights, the weight of an independent set is the summation of weights of nodes in the independent set.

Based on results of Tassiulas and Ephremides [2] and optimization formulation of resource allocation, a throughput optimal algorithm for resource allocation and scheduling is equivalent to finding 'maximum weight independent set' (MWIS) in $G$ every time, where weight is function of queue-size and other network parameters. We refer interested readers to a recent survey by Lin, Shroff and Srikant [1] where a detailed account of this development is given for wireless network with *node-exclusive* interference model (aka matching constraints).

### B. Previous work

We present a brief summary of previous work on network scheduling algorithms. The result by Tassiulas and Ephremides [2] established that 'max-weight scheduling' policy is throughput optimal for a large class of scheduling problems. This result has been very influential in design of scheduling algorithms since then. Application to input-queued switches led to an excellent development of theory and practice of algorithms for scheduling under matching constraints: notably, the results of [3]–[9]. A recent interest in wireless network has led to proposal of distributed scheduling algorithms under matching constraints [10]–[13]. Most of these algorithms, based on finding maximal matching, guarantee only a constant fraction of throughput. Recently, Modiano, Shah and Zussman [14] exhibited a throughput optimal distributed scheduling algorithm with matching constraints. This algorithm, as discussed in [1], easily extends to provide throughput optimal algorithm for resource allocation and scheduling problem under matching constraints.

Apart from matching constraints, other scheduling constraints have received limited attention primarily due to inherent hardness of the other constraints. For example, Sharma, Mazumdar and Shroff [15] identify that max-weight scheduling with $K-$hop matching constraint becomes an instance of computationally hard combinatorial optimization problem.

They provide a centralized throughput optimal algorithm for unit disk graphs based on work by Hunt et. al. [16]. However, hardness of the max. wt. problem does not imply non-existence of throughput optimal algorithm. Specifically, in this paper we provide throughput optimal *distributed* algorithm (ALGO I) for hard independent set constraint (it will naturally extend to the $K$-hop matching model of [15] as well).

## C. Contribution

The maximum weight independent set (MWIS) algorithm is throughput optimal for our setup. However, finding MWIS is NP-hard [17] and hard to approximate within $n^{1-o(1)}$ ($B/2^{O(\sqrt{\log B})}$ for degree $B$ graph) factor [6]. This raises a challenging question: is it even possible to have any throughput optimal, polynomial (in $n$) time distributed algorithm? if yes, how does it's delay scale ? more generally, is it possible to have both throughput and delay optimal polynomial time distributed algorithm for practical network topology? As the main contribution of this paper, we answer these tantalizing questions in affirmative.

First, we exhibit a distributed throughput optimal scheduling algorithm that takes $O(n^3)$ total operations to compute schedule (section III). By computing schedule once in $O(n^3)$ time, the cost per time is $O(1)$. Such *lazy* schedule is throughput optimal. That is, it is not difficult to have stable scheduling algorithm even when scheduling constraints are very hard. However, this algorithm is likely to induce exponentially large (in $n$) delay. This suggests that the complexity of algorithm trades off with delay rather than throughput.

Next, we present a delay (order) optimal scheduling algorithm that essentially finds excellent approximation to MWIS in $O(n)$ operations in total or $O(1)$ operations per node for 'practical networks' modeled as non-expanding graphs (section IV). The algorithm is distributed and simple. It is based on a new randomized distributed graph partitioning with certain properties. Next, we provide definition and examples of non-expanding graph.[4]

*Non-expanding graphs.* Given a graph $G = (V, E)$, let $D : V \times V \to \mathbb{R}_+$ be a metric on nodes of $V$. A special metric induced by $G$ is the shortest-path distance metric, $D_G : V \times V \to \mathbb{R}_+$ where $D_G(u, v)$ is the length of shortest path connecting $u, v$ ($\infty$ if $u, v$ are not connected). With respect to a given metric $D$ (not necessarily $D_G$), for a given vertex $v \in G$ and $i \in \mathbb{N}$, let $f_v(i) = |\{w \in V : i - 1 < D(v, w) \leq i\}|$, and $F_v(i) = |\{w \in V : D(v, w) \leq i\}| = \sum_{j=1}^{i} f_v(j)$.

**Definition 1** *A graph $G$ is said to be "non-expanding" if there exists a metric $D : V \times V \to \mathbb{R}_+$, constants $\Delta, \beta$ such that*

(0) *(Contracting) $D \leq D_G$, i.e. $D(u, v) \leq D_G(u, v)$, $\forall$ $(u, v) \in V \times V$.*
(1) *(Bounded neighbors) $F_v(1) \leq \Delta$ for all $v \in V$.*
(2) *(Polynomial-growth)[5] $F_v(3i) \leq \beta F_v(i)$ $\forall$ $i$.*
(3) *(Absence-of-thick-boundary)[6] For any $\varepsilon > 0$, there exists constant $\ell(\varepsilon)$ such that*

$$\left[\sum_{i=1}^{\ell(\varepsilon)} f_v(i)^2\right] \leq \varepsilon F_v^2(\ell(\varepsilon)).$$

*Example 1.* Consider a $\sqrt{n} \times \sqrt{n}$ grid graph of $n$ nodes in two-dimension. Then, for $D = D_G$ it is non-expanding as we have $f_v(i) = \Theta(i)$ (with $\ell(\varepsilon) = O(1/\varepsilon)$).

*Example 2:* Suppose there are infinitely many nodes placed in a plane (or even three dimension) so that for some $R > 0$, (a) nodes are connected to each other if they are within distance $R$ of each other, and (b) number of nodes in any disc of radius $\alpha R$ is bounded above by $\gamma$ and below by 1 where $\alpha \in (0, 1/2]$ and $\gamma \geq 1$ are constants. Now consider any square of side-length $N$ in plane. Let $G$ be the graph formed by nodes within this square. Then it is non-expanding as shown in the [18], with respect to metric $D = d/R$ where $d$ is Euclidian distance. Note that such a model captures the nature of wireless nodes deployed in practice.

**Remark:** Finally, some remarks on our results: (a) We consider the single-hop model. However, it should be clear to an informed reader that exactly the same algorithms with different weights will provide desired *optimal* performance: weights being "difference of queue-sizes" under multi-hop model of [2] and weights being appropriate Lagrange parameters for resource allocation in multi-hop network as explained in [1]. (b) The independent set constraint is general enough abstract model to capture any combinatorial scheduling constraint. Thus, our results should extend to a large class of scheduling problem. For example, a natural adaptation of ALGO I for $K$-hop matching model will provide distributed throughput optimal algorithm (thus, answering the question implicitly raised in [15]). (c) We note conceptual similarity of ALGO II with that of [16]. However, inherently the algorithm of [16] is centralized (uses dynamic programing and centralized graph partition) while ours is distributed.

## II. NOTATIONS AND MODEL

As before, let $G = (V, E)$ be the undirected network graph with $|V| = n$. Let $\mathcal{N}(v) = \{u \in V : (u, v) \in E\}$ denote the set of all neighbors of $v \in V$. The time is assumed to be slotted and $\tau \in \mathcal{Z}_+$ denote the time. Each node $v \in V$ is capable of wireless transmission at unit rate to any of its neighbor. We ignore the power control for simplicity but as reader may notice, it can be easily included in the model. At each node, packets (of unit size) are arriving according to an external arrival process. Let $\bar{A}(\tau) = [\bar{A}_v(\tau)]$ denote the cumulative arrival process until time $\tau \in \mathcal{Z}_+$, i.e. $\bar{A}_v(\tau)$ be the total number of packet arrived at node $v$ in the time interval $[0, \tau]$; $\bar{A}(\tau) = \mathbf{0}$. Let $A_v(\tau) = \bar{A}_v(\tau) - \bar{A}_v(\tau - 1)$ be the number of packets arriving at node $v$ in time slot $\tau$. We assume that at most one packet can arrive at a node $v$ in a

[4]A reader may skip this definition and come back to it on reaching section IV.

[5]The condition immediately implies that $F_v(k) \leq k^{\log_3 \beta}$. Hence the name polynomial-growth.

[6]The condition says that no 'boundary' formed by nodes at a particular distance should have most of the nodes till range $\ell(\varepsilon)$.

time slot, i.e. $A_v(\tau) \in \{0,1\}$. Finally, we assume that $A_v(\cdot)$ are Bernoulli i.i.d. random variable with $\Pr(A_v(\tau)=1)=\lambda_v$. Let $\lambda = [\lambda_v]$ denote the arrival rate vector.

For simplicity and ease of explanation, we assume that network is a single-hop[7], i.e. data arriving at a node $v$ is to be sent to one of its neighbors. Let $Q_v(\tau)$ denote the queue-size at node $v$ at time $\tau$ with $Q(\tau) = [Q_v(\tau)]$. We assume the system starts empty, i.e. $Q(0) = \mathbf{0}$. Let $\bar{D}(\tau) = [\bar{D}_v(\tau)]$ denotes the cumulative departure process from $Q(\tau)$; $D(\tau) = [D_v(\tau)]$ denote the number of departures in time slot $\tau$. Then,

$$\begin{aligned} Q(\tau) &= Q(0) + \bar{A}(\tau) - \bar{D}(\tau) = \bar{A}(\tau) - \bar{D}(\tau) \\ &= Q(\tau-1) + A(\tau) - D(\tau). \end{aligned} \tag{1}$$

Departure happens according to the scheduling algorithm which need to satisfy interference constraint that no two neighboring nodes are transmitting data in the same time slot. To this end, let $\mathcal{I}$ denote the set of all independent set of $G$. Then, at each time the scheduling algorithm schedules nodes of an independent set $I \in \mathcal{I}$ to transmit packets. In what follows, we will denote independent set $I$ as vector $I = [I_v]$ with $I_v \in \{0,1\}$ and $I_v = 1$ indicates that node $v$ is in $I$.

We say that a system is *stable* for given $\lambda$ under the particular scheduling policy if

$$\limsup_{\tau \to \infty} \mathbb{E}[Q_v(\tau)] < \infty, \quad \forall\, v \in V.$$

From [2], it is clear that the set of all $\lambda$ for which there exists a scheduling policy so that the system is stable is given by $\Lambda = \mathsf{Co}(\mathcal{I})$, where $\mathsf{Co}(\mathcal{I})$ is the convex hull of $\mathcal{I}$ in $\mathbb{R}^n$. Hence, we call $\mathsf{Co}(\mathcal{I})$ the *throughput region* of the system.

In [2], it was shown that a 'maximum weight independent set' scheduling algorithm is stable for all $\lambda \in \mathsf{Co}(\mathcal{I})$, where the schedule or independent set $I^*(\tau)$ chosen at time $\tau$ is

$$I^*(\tau) = \arg\max_{I \in \mathcal{I}} \langle I, Q(\tau-1)\rangle,$$

with notation that $\langle A, B\rangle = \sum_{v\in V} A_v B_v$. Such an algorithm will be called to provide 100% throughput or through optimal.

As discussed before, finding max. wt. independent set can be computationally hard. In the rest of the paper, we will be interested in designing scheduling algorithms that are: (a) stable, (b) induce low average queue-size (equivalently low delay due to Little's Law) and (c) simple and distributed.

## III. Distributed Stable Algorithm

We describe a simple and distributed stable algorithm, denoted by Algo I. The algorithm uses two distributed sub-routines, Random and Aprx-Cnt, with the following properties, explained later in this section:

**P1.** Random samples independent sets of graph $G$ in distributed manner so that each independent set has get sampled with probability at least $2^{-n}$. It takes total $O(|E|+n) \le O(n^2)$ distributed operations for any $G$.

[7]The model ignores multi-hop situation. However, as explained in [2], the scheduling algorithm remains maximum weight independent set with weights being "difference of queue-sizes". Similarly, in the context of resource allocation as explained in [1], the weights are based on "Lagrange multipliers".

**P2.** Aprx-Cnt($\varepsilon$) takes given independent set and node weights $W$ and produces an estimate of $w_I = \langle I, W\rangle$, say $\hat{w}$, so that $\hat{w} \in ((1-\varepsilon)w_I, (1+\varepsilon)w_I)$ with probability at least $1 - 3^{-n}$ in total $O(n^3)$ distributed operations for any $G$.

```
Algo I
∘ Let $I(\tau)$ be independent set schedule chosen by algorithm at time $\tau$.
∘ At time $\tau+1$, choose $I(\tau+1)$ as follows:
    – Generate a random independent set $R(\tau+1)$ using Random.
    – Obtain estimates $\hat{w}_I, \hat{w}_R$ of weights of $I(\tau)$, $R(\tau+1)$ with respect to $Q(\tau)$ respectively using Aprx-Cnt($\varepsilon/8$).
    – If $\hat{w}_R > \frac{(1+\varepsilon/8)}{(1-\varepsilon/8)}\hat{w}_I$, then set $I(\tau+1) = R(\tau+1)$. Else, set $I(\tau+1) = I(\tau)$.
∘ Repeat the above algorithm every time.
```

Before we establish that algorithm is stable (or throughput optimal), we will describe the sub-routines Random, Aprx-Cnt and their properties useful in the analysis.

### A. Random

The algorithm Random is described as follows. The proof of it satisfying property **P1** can be found in [18].

```
Random
∘ Each node $v \in V$ chooses $I_v = 0$ or 1 with probability 1/2 independently.
∘ If node $v$ finds any $u \in \mathcal{N}(v)$ such that $I_u = 1$, it immediately sets $I_v = 0$.
∘ Now, output $I = [I_v]$ as a sampled independent set.
```

### B. Aprx-Cnt *and its properties*

The purpose of algorithm is to compute summation of node weights (approximately) for given independent set. A useful property of this algorithm is that all nodes obtain the same estimate and hence allows for distributed decision in Algo I. Now, some useful probabilistic facts:

**F1.** Let $X_1, \ldots, X_k$ be independent random variables with exponential distribution and parameters $r_1, \ldots, r_k$. Then, $X_* = \min_{1\le i\le k} X_i$ has exponential distribution with parameter $\sum_{i=1}^k r_i$.

**F2.** Let $Y_1, \ldots, Y_m$ be independent exponential random variables with parameter $r$. Let $S_m = \frac{1}{m}\sum_{i=1}^m Y_i$. Then, for $\gamma \in (0, 1/2)$

$$\Pr\left(S_m \notin (1-\gamma)r^{-1}, (1+\gamma)r^{-1}\right) \le 2\exp\left(-\gamma^2 m/2\right).$$

The **F1** is well-known about exponential distribution; the **F2** follows from Cramer's Theorem [19] about large deviation estimation for exponential distribution.

Given an independent set $I = [I_v]$ and node weights $W = [W_v]$, **F1** and **F2** can be used to compute this weight approximately as follows: each node $v \in V$ draws an independent exponential random variable with parameter $W_v$ (nodes with $W_v = 0$ or $I_v = 0$ do not participate); then they compute minimum, say $X_*$ of these random numbers in distributed fashion by iteratively asking their neighbors for their estimates of minimum. Nodes should terminate this process after $\Theta(n)$ transmissions. Repeat this for $m$ times to obtain minimums $X_*(i), 1 \leq i \leq m$.

Now set $S_m = \frac{1}{m}\sum_{i=1}^{m} X_*(i)$ and declare $Z_m = 1/S_m$ as an estimate of weight of independent set, i.e. $\langle I, W\rangle$.

Now, given small enough $\varepsilon$ it follows from **F1**, **F2** that by selecting $m = O(\varepsilon^{-2}n)$, we obtain estimate of weight of an independent set, say $\hat{w}(I)$ such that

$$\Pr\left(\hat{w}(I) \notin ((1-\varepsilon)\langle I, W\rangle, (1+\varepsilon)\langle I, W\rangle)\right) \quad \leq \quad 3^{-n}. \, (2)$$

Computation of a single minimum over the network can be done in a distributed manner in many ways. We skip the details here in interest of space. However, we refer an interested reader to see [20] for interesting account on such algorithms. The minimum computation takes total $O(n^2)$ exchanges or $O(n)$ per node. This provides property **P2**.

### C. Algo I*: stability and complexity*

**Complexity.** The algorithm Algo I uses two sub-routines, Random and Aprx-Cnt at every time-slot. Properties **P1** and **P2** imply that these two algorithms take $O(n^2)$ and $O(n^3)$ total (network-wide) operations for fixed $\varepsilon$. Thus, algorithm Algo I performs $O(n^3)$ net operations to compute a schedule. The complexity burden can be reduced by making the algorithm *lazy* as follows: find schedule every $T$ time-steps and use the same schedule for in between the $T$ steps. As long as $T$ is finite, the algorithm remains stable and the average queue-size increases only by $O(nT)$. Thus, by choice of $T = \Theta(n^3)$, the same conclusion as in Theorem 1 can be obtained with amortized cost of $O(1)$ operation per time-step.

**Stability.** Now, throughput optimality of Algo I.

**Theorem 1** *The algorithm* Algo I *based on* Random *and* Aprx-Cnt *is stable as long as* $\lambda \in (1-\varepsilon)\mathsf{Co}(\mathcal{I})$ *for any* $\varepsilon > 0$. *Further, (the proof can be found in [18].)*

$$\lim_{\tau\to\infty} \mathbb{E}[\langle Q(\tau), \mathbf{1}\rangle] = O(6^n).$$

## IV. Distributed Stable Algorithm: Lower Delay

The Algo I shows that scheduling problems with hard constraint such as independent set can have extremely simple, distributed and stable algorithms. The Algo I essentially finds independent set schedule whose average weight at any time $\tau$ is $(1-\varepsilon)W^*(\tau) - B_n$, where $W^*(\tau)$ is weight of max. wt. independent set and $B_n$ is some exponentially large constant. The stability follows due to small multiplicative approximation loss of $1-\varepsilon$, but the average queue-size suffers due to large constant $B_n$. This suggests that we need an algorithm that has average weight at least $(1-\varepsilon)W^*(\tau)$.

As noted earlier, finding approximation to max. wt. independent set is computationally hard. That is, there exists graph instances for which finding such approximation will require exponential time, unless $P = NP$. However, the question is: are graphs arising in practice are of this type ? Next, we present algorithm for practical graphs modeled as non-expanding to obtain approximate max. wt. ind. set. schedule.

### A. Graph-Partition *and its properties*

Given a non-expanding graph $G$ (with a metric $D$) and $\varepsilon > 0$, let $\mathbf{L}(\varepsilon) \geq 3$ be a constant such that for any $v \in V$,

$$\sum_{i \leq \mathbf{L}(\varepsilon)} f_v^2(i) \quad \leq \quad \frac{\varepsilon}{3\beta^4(\log\varepsilon^{-1})^2} F_v^2(\mathbf{L}(\varepsilon)). \qquad (3)$$

Let $N(\varepsilon) = \max_v F_v(\mathbf{L}(\varepsilon))$, and define

$$p_v(\varepsilon) \quad = \quad \frac{\beta^2 \log\varepsilon^{-1}}{F_v(2\mathbf{L}(\varepsilon))}. \qquad (4)$$

The Graph-Partition algorithm that partitions graph in good clusters and boundary is described as follows.

Graph-Partition

---

(0) Each $v \in V$ becomes *cluster-center* independently with probability $p_v$.
(1) If $v$ becomes a center, $v$ sends notifying messages to nodes within distance $\mathbf{L}(\varepsilon)$ w.r.t. $D$. To implement it distributively, each node remembers all the nodes within distance $\mathbf{L}(\varepsilon)$ w.r.t. $D$, and spreads a received message if the message is from a node within distance $\mathbf{L}(\varepsilon)$.
(2) A node $w$ takes decision as follows. $w$ is in boundary if either of the following is true:
  (a) $w$ does not receive message from any node at distance $\leq \mathbf{L}(\varepsilon) - 1$.
  (b) If $w$ receives messages from two or more vertices, say $v_1, \ldots, v_k$ so that $D(v_i, w) \leq D(v_{i+1}, w)$ for $1 \leq i \leq k-1$, and $|D(v_1, w) - D(v_2, w)| \leq 2$.
(3) If none of (2)(a)-(2)(b) is satisfied then $w$ is in the cluster of a node $v$ that is closet to $w$ among all nodes from which $w$ has received a message.

---

The Graph-Partition algorithm has the following property (proved in [18]).

**Lemma 2** *Under* Graph-Partition, *each node* $v \in V$ *is in boundary set* $B$ *with probability at most* $2\varepsilon$. *Also, each cluster is of size at most* $N(\varepsilon)$ *and nodes that belong to different clusters are not connected.*

### B. Algorithm Algo II

Now, we present a randomized algorithm that essentially finds an ind. set with average weight $(1-\varepsilon)W^*(\tau)$ in total $O(n)$ message exchanges for any *non-expanding* graph.

The basic idea behind algorithm is as follows: given partition of nodes of $V$ by Graph-Partition into disjoint sets $S_1, \ldots, S_K$ each of size $O(1)$ and boundary set $B$ so that $S_i$ form connected components of $G$ (i.e. no two vertices in different $S_i$s are connected to each other), separated by nodes in $B$. We find exact max. wt. independent set, say $I^i$, restricted to each of $S_i$ using essentially ALGO I multiple (constant) times. Then, form an independent set as $I = \cup I^i$, i.e. set all nodes in $B$ to 0. The $I$ is a valid independent set in $G$. Now, if sum of weights of nodes in $B$ is *small* compared the weight of $I$, then $I$ is a good approximation of max. wt. independent set. This is guaranteed by Lemma 2.

```
ALGO II
(0) At each time $\tau$, algorithm performs steps (1)-(3).
(1) Given $\varepsilon > 0$, partition the graph into $\Theta(n)$ clusters $S_1, \dots, S_{\phi n}$, $\phi \in (0,1)$ and boundary set $B$ using GRAPH-PARTITION. Each $S_i$ has $N_i(\varepsilon)$ nodes, which is at most $N(\varepsilon)$.
    ◦ each node knows whether it is in a cluster or in $B$.
(2) Each cluster $S_i$ do: for $k = 1, \dots, 2.5^{N_i(\varepsilon)}$
    (o) Initially, set $k = 1$ and $I^i(0) = \mathbf{0}$ (i.e. empty set).
    (a) Generate a random independent set $R^i(k)$ using RANDOM.
    (b) Find weight estimate $w^i_R(k), w^i_I(k-1)$ of $R^i(k), I^i(k-1)$ using APRX-CNT($\varepsilon/8$).
    (c) Set $I^i(k) = R^i(k)$ if $w^i_R(k) > \left(\frac{1+\varepsilon/8}{1-\varepsilon/8}\right) w^i_I(k-1)$; else $I^i(k) = I^i(k-1)$.
    (d) Set $k = k+1$ and repeat (a)-(c) till $k = 2.5^{N_i(\varepsilon)}$. When $k = 2.5^{N_i(\varepsilon)}$ call $I^i(k)$ as $I^i(\tau)$ (with abuse of notation).
(3) Declare schedule at time $\tau$, $I(\tau) = \cup I^i(\tau)$, i.e. nodes use their distributively learnt assignment.
```

We state the main theorem about performance of ALGO II whose proof depends on properties of GRAPH-PARTITION and APRX-CNT, which can be found in [18].

**Theorem 3** *Given non-expanding graph $G$, the algorithm* ALGO II *is stable as long as $\lambda \in (1-\delta(\varepsilon))\mathsf{Co}(\mathcal{I})$ for any small enough $\varepsilon > 0$ and*

$$\lim_{\tau\to\infty} \mathbb{E}[\langle Q(\tau), \mathbf{1}\rangle] = O(n),$$

*where $\delta(\varepsilon) = 4\varepsilon(\Delta+2)$. This average queue-size is order optimal, i.e. there exists $\lambda \in (1-\varepsilon)\mathsf{Co}(\mathcal{I})$ for non-expanding $G$ such that for any algorithm*

$$\liminf_{\tau\to\infty} \frac{1}{\tau}\sum_{s=0}^{\tau-1} \mathbb{E}[\langle Q(s), \mathbf{1}\rangle] = \Omega(n).$$

*Further,* ALGO II *takes $O(n)$ total operations with constant dependent on $\varepsilon$.*

## C. ALGO II : *Complexity, Stability and Delay*

**Complexity.** The algorithm ALGO II uses GRAPH-PARTITION, RANDOM and APRX-CNT. The algorithm GRAPH-PARTITION takes $O(n)$ distributed operations since a message generated by each node can traverse at most $O(N^2(\varepsilon))$ times. The ALGO II calls RANDOM and APRX-CNT for $O(2.5^{N(\varepsilon)})$ times for each of the $\Theta(n)$ partitions. But since each partition is of size at most $N(\varepsilon)$, the net operation done by RANDOM and APRX-CNT for each partition is constant (dependent on $N(\varepsilon)$). Subsequently, the total operations performed by ALGO II is $O(n)$ with constant dependent of $\varepsilon$.

**Stability and Delay.** These properties are stated in Theorem 3 which essentially uses Lemma 4 (proved in [18]). Let $I(\tau)$ be schedule chosen by ALGO II at time $\tau$, and the max. wt. independent set be $I^*(\tau)$, i.e. $I^*(\tau) = \arg\max_{I\in\mathcal{I}}\langle Q(\tau-1), I(\tau)\rangle$. Let $W(\tau) = \langle Q(\tau-1), I(\tau)\rangle$, and let $W^*(\tau) = \langle Q(\tau-1), I^*(\tau)\rangle$.

**Lemma 4** *For non-expanding graph $G$ with maximum vertex degree $\Delta$, for any time $\tau$, $\mathbb{E}[W(\tau)] \geq (1-0.5\delta(\varepsilon))W^*(\tau)$, where expectation is with respect to randomness of the algorithm* GRAPH-PARTITION.

## REFERENCES

[1] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *Submitted, available through* csl.uiuc.edu/rsrikant, 2006.
[2] 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*, vol. 37, pp. 1936–1948, 1992.
[3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proceedings of IEEE Infocom*, 1996, pp. 296–302.
[4] N. McKeown, "iSLIP: a scheduling algorithm for input-queued switches," *IEEE Transaction on Networking*, vol. 7, no. 2, pp. 188–201, 1999.
[5] J. Dai and B. Prabhakar, "The throughput of switches with and without speed-up," in *Proceedings of IEEE Infocom*, 2000, pp. 556–564.
[6] L. Trevisan, "Non-approximability results for optimization problems on bounded degree instances," in *ACM STOC*, 2001.
[7] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," vol. 21, no. 4, 2003, pp. 546–559.
[8] D. Shah, "Stable algorithms for input queued switches," in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2001.
[9] D. Shah and D. J. Wischik, "Optimal scheduling algorithm for input queued switch," in *IEEE INFOCOM*, 2006.
[10] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inf. Theory*, vol. 34, 1988.
[11] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in *43rd Allerton conference on Comm. Control and computing*, 2005.
[12] L. Chen, S. H. Low, M. Chang, and J. C. Doyle, "Optimal cross-layer congestion control, routing and scheduling design in ad-hoc wireless networks," in *IEEE INFOCOM*, 2006.
[13] X. Lin and N. B. Shroff, "Impact of imperfect scheduling in wireless networks," in *IEEE INFOCOM*, 2005.
[14] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless network via gossiping," in *ACM SIGMETRICS/Performance*, 2006.
[15] G. Sharma, R. Mazumdar, and N. Shroff, "On the complexity of scheduling in wireless networks," in *ACM Mobicom*, 2006.
[16] H. Hunt-III, M. Marathe, V. Radhakrishnan, S. Ravi, D. Rosenkrantz, and R. Stearns, "Nc-approximation schemes for np- and pspace-hard problems for geometric graphs," *J. Algorithms*, vol. 26, no. 2, pp. 238–274, 1998.
[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.*
[18] K. Jung and D. Shah, "Low delay scheduling in wireless network," *Preprint, available at http://web.mit.edu/devavrat/www/delay.pdf.*
[19] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications.* Jones and Barlett Publishers, 2003.
[20] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *ACM PODC*, 2006.