Node-based Service-Balanced Scheduling for Provably Guaranteed Throughput and Evacuation Time Performance

Bo Ji, Gagan R. Gupta, and Yu Sang

***Abstract*—This paper focuses on the design of *provably efficient online* link scheduling algorithms for multi-hop wireless networks. We consider single-hop flows and the one-hop interference model. The objective is twofold: 1) *maximize the throughput* when the flow sources continuously inject packets into the network, and 2) *minimize the evacuation time* when there are no future packet arrivals. The prior work mostly employs the link-based approach, which leads to throughput-efficient algorithms but often does not guarantee satisfactory evacuation time performance. In this paper, we adopt a novel *node-based approach* and propose a *service-balanced online* scheduling algorithm, called NSB, which gives balanced scheduling opportunities to the nodes with heavy workload. We rigorously prove that NSB guarantees to achieve an efficiency ratio no worse (or no smaller) than $2/3$ for the throughput and an approximation ratio no worse (or no greater) than $3/2$ for the evacuation time. It is remarkable that NSB is both throughput-optimal and evacuation-time-optimal if the underlying network graph is bipartite. Further, we develop a lower-complexity NSB algorithm, called LC-NSB, which provides the same performance guarantees as NSB. Finally, we conduct numerical experiments to elucidate our theoretical results.**

I. Introduction

Resource allocation is a vital and challenging problem in wireless networks. Various functionalities in different layers (transport, network, MAC, and PHY) need to be carefully designed so as to efficiently allocate network resources and achieve optimal or near-optimal network performance. Among these critical functionalities, link scheduling in the MAC layer, which, at each time decides which subset of non-interfering links can transmit data, is perhaps the most challenging component and has attracted a great deal of research effort in the past decades (see [1], [2] and references therein).

In this paper, we focus on the design of *provably efficient online* link scheduling algorithms for *multi-hop* wireless networks with *single-hop* flows under the *one-hop* interference model[1]. The objective is twofold: 1) *maximize the throughput* (i.e., stabilize the network under any feasible traffic load) when the flow sources continuously inject packets into the network, and 2) *minimize the evacuation time* (i.e., minimize the time interval needed for evacuating all the existing packets) when there are no future packet arrivals. Both throughput and evacuation time are important metrics of network performance. While throughput is widely shared as the first-order performance metric, which characterizes the long-term average traffic load that can be supported by the network, evacuation time is a critical metric in the settings without future arrivals. One practical example of such settings is environmental monitoring and data collection in wireless sensor networks, where all the measurement data periodically generated by different nodes at the same time need to be transferred to one or multiple sinks within a minimum time for further processing. Evacuation time metric is also related to the delay performance in scenarios with arrivals [4], as well as meeting real-time scheduling requirements (e.g., packet delivery with deadlines) [5], [6].

However, these different metrics may lead to conflicting scheduling decisions – an algorithm designed for optimizing one metric may be detrimental to the other metric (see [4] for such examples). Therefore, it is challenging to design an efficient scheduling algorithm that can provide provably guaranteed performance for both metrics at the same time.

While throughput has been extensively studied since the seminal work by Tassiulas and Ephremides [7] and is now well understood, evacuation time is much less studied. In the no-arrival setting, the minimum evacuation time problem can be mapped to the *multigraph edge coloring problem*[2], where each multi-edge corresponds to a packet waiting to be transmitted over the link between the nodes of the multi-edge, and each color corresponds to a feasible schedule (or a matching). This is a well-known NP-hard problem [8]. Hence, a rich body of research on edge coloring has focused on developing approximation algorithms (see [9] for a good survey). These algorithms employ a popular recoloring approach that requires computing the colors all at once, and yield a complexity that depends on the number of multi-edges. This, however, renders them unsuitable for application in a network with arrivals. Because there could be a large number of packets (or multi-edges) in the network, and the complexity could become impractically high. Therefore, it is desirable to have an *online* scheduling algorithm that at each time quickly computes one schedule (or color) based on the current network state (e.g., the queue lengths) and yields a complexity that only depends on the node count n and/or the link count m.

B. Ji and Y. Sang are with Dept. of CIS at Temple University, Philadelphia, PA 19122, USA. G. R. Gupta is with AT&T Labs, USA. Emails: boji@temple.edu, gagan.gupta@iitdalumni.com, yu.sang@temple.edu.

[1]The packets of a single-hop flow traverse only one link before leaving the network. The one-hop interference model is also called the node-exclusive or the primary interference model [1], [3], where two links sharing a common node cannot be active at the same time.

[2]In a multigraph, more than one edge, called multi-edge, is allowed between two nodes. The multigraph edge coloring problem is to find the minimum number of colors, such that each multi-edge is assigned a color, and two multi-edges sharing a common node cannot have the same color.

Most existing online scheduling algorithms either make scheduling decisions based on the link load (such as Maximum Weighted Matching (MWM) [7] and Greedy Maximal Matching (GMM) [10], [11]) or are load agnostic (such as Maximal Matching (MM) [10], [12]). While these algorithms are throughput-efficient, none of them can guarantee an approximation ratio better (or smaller) than 2 for the evacuation time [4]. In contrast, several prior work [4], [13]–[15] proposes algorithms based on the node workload (i.e., packets to transmit or receive), such as the Lazy Heaviest Port First (LHPF) algorithms, which are both throughput-optimal and evacuation-time-optimal in input-queued switches (which can be described as bipartite graphs) [4]. The *key intuition* behind the node-based approach is that the minimum evacuation time is lower bounded by the largest workload at the nodes and the odd-size cycles, and this lower bound is asymptotically tight [16]. The link-based approach fails to respect this crucial fact and thus leads to unsatisfactory evacuation time performance.

While the node-based approach seems quite promising, performance of the node-based algorithms is less understood, and the studies are mainly limited to bipartite graphs [4], [13], [14]. Very recent work in [15] considers general network graphs and shows that the Maximum Vertex-weighted Matching (MVM) algorithm can guarantee an approximation ratio no worse (or no greater) than $3/2$ for the evacuation time. However, throughput performance of MVM remains unknown.

There is several other related work. In [17], the authors study the connection between throughput and (expected) minimum evacuation time, but no algorithms with provable performance guarantees are provided. Some work [5], [18], [19] considers the minimum evacuation time problem for multi-hop flows, but in some special scenarios only (e.g., special network topologies or wireline networks without interference).

In this paper, *the goal is to develop efficient online link scheduling algorithms that can provide provably guaranteed performance for both throughput and evacuation time.* We summarize our contributions as follows.

First, we propose a Node-based Service-Balanced (NSB) scheduling algorithm that makes scheduling decisions based on the node workload and whether the node was scheduled in the previous time-slot(s). NSB has a complexity of $O(m\sqrt{n}\log n)$. We rigorously prove that NSB guarantees to achieve an approximation ratio no worse (or no greater) than $3/2$ for the evacuation time and an efficiency ratio no worse (or no smaller) than $2/3$ for the throughput. It is remarkable that NSB is both throughput-optimal and evacuation-optimal if the underlying network graph is bipartite. The *key novelty* of NSB is that it takes a node-based approach and gives balanced scheduling opportunities to the bottleneck nodes with heavy workload. A novel application of *graph-factor theory* allowed us to analyze how NSB schedules the heavy nodes (Lemma 6).

Second, from the proofs for NSB, we learn that in order to achieve the same performance guarantees, what really matters is the priority or the ranking of the nodes, rather than the exact weight of the nodes. Using this insight, we develop the Lower-Complexity NSB (LC-NSB) algorithm. We show that LC-NSB can provide the same performance guarantees as NSB, while enjoying a lower complexity of $O(m\sqrt{n})$.

Algorithm	Complexity	γ (Throughput)		η (Evacuation time)	
		General	Bipartite	General	Bipartite
MWM	$O(mn)$	1	1	2	2
GMM	$O(m\log m)$	$\geq 1/2$	$\geq 1/2$	2	2
MM	$O(m)$	$\geq 1/2$	$\geq 1/2$	2	2
MVM	$O(m\sqrt{n}\log n)$	unknown	1	$\leq 3/2$	1
NSB	$O(m\sqrt{n}\log n)$	$\geq 2/3$	1	$\leq 3/2$	1
LC-NSB	$O(m\sqrt{n})$	$\geq 2/3$	1	$\leq 3/2$	1

TABLE I: Performance comparison of NSB and LC-NSB with several most relevant online algorithms in the literature. The efficiency ratio γ and the approximation ratio η are used for comparing the performance of throughput and evacuation time, respectively. (See formal definitions of γ and η in Section II.) For both γ and η, a value closer to 1 is better. The complexity provided here is for making a scheduling decision at each time.

In Table I, we summarize the guaranteed performance of NSB and LC-NSB as well as several most relevant online algorithms in the literature. *To the best of our knowledge, none of the existing algorithms strike a more balanced performance guarantees than NSB and LC-NSB in both dimensions of throughput and evacuation time.* Finally, we conduct numerical experiments to validate our theoretical results and compare the empirical performance of various algorithms.

The remainder of this paper is organized as follows. First, we describe the system model and the performance metrics in Section II. Then, we propose the NSB algorithm and analyze its performance in Section III. A lower-complexity NSB algorithm with the same performance guarantees is developed in Section IV. Finally, we conduct numerical experiments in Section V and make concluding remarks in Section VI.

II. System Model

We consider a multi-hop wireless network described as an undirected graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of links. We use n and m to denote the node count and the link count, respectively. Nodes are wireless transmitters/receivers and links are wireless channels between two nodes. The set of links touching node $i \in V$ is defined as $L(i) \triangleq \{l \in E \mid i \text{ is an end node of link } l\}$. We assume a time-slotted system with a single frequency channel. We also assume unit link capacities, i.e., a link can transmit at most one packet in each time-slot when active. However, our analysis can be extended to the general scenario with heterogeneous link capacities by considering the workload defined as $\lceil \text{number of packets/link capacity} \rceil$. We consider the *one-hop* interference model, under which a feasible schedule corresponds to a *matching* (i.e., a subset L of links satisfying that no two links in L share a common node). A matching is called *maximal*, if no more links can be added to the matching without violating the interference constraint. We let $\mathcal{M}$ denote the set of all matchings over G.

In this paper, we focus on the link scheduling problem and assume single-hop flows (that traverse only one link). We let $A_l(k)$ denote the cumulative amount of workload (or packet) arrivals at link $l \in E$ up to time-slot k, (including time-slot k). By slightly abusing the notations, we let $A_i(k) \triangleq \sum_{l \in L(i)} A_l(k)$ denote the cumulative amount of workload arrivals at node $i \in V$ up to time-slot k. (Indices l and i correspond to links and nodes, respectively; similar for other notations.) We assume that the arrival process $\{A_l(k), k \geq 0\}$ satisfies the strong law of large numbers (SLLN): with probability one,

$$\lim_{k \to \infty} \frac{A_l(k)}{k} = \lambda_l \tag{1}$$

for all links $l \in E$, where λ_l is the mean arrival rate of link l. Let $\lambda \triangleq [\lambda_l : l \in E]$ denote the arrival rate vector. We assume that the arrival processes are independent across links. Note that the process $\{A_i(k), k \geq 0\}$ also satisfies SLLN: with probability one, $\lim_{k \to \infty} \frac{A_i(k)}{k} = \lambda_i$ for all nodes $i \in V$, where $\lambda_i \triangleq \sum_{l \in L(i)} \lambda_l$ is the mean arrival rate for node i.

Let $Q_l(k)$ be the queue length of link l in time-slot k, and let $D_l(k)$ be the cumulative number of packet departures at link l up to time-slot k. We assume that there are a finite number of initial packets in the network at the beginning of time-slot 0. Let $Q_i(k) \triangleq \sum_{l \in L(i)} Q_l(k)$ be the amount of workload at node $i \in V$ (i.e., the number of packets waiting to be transmitted to or from node i) in time-slot k, and let $D_i(k) \triangleq \sum_{l \in L(i)} D_l(k)$ be the amount of cumulative workload served at node $i \in V$ up to time-slot k. We also call $Q_i(k)$ and $D_i(k)$ as the queue length and the cumulative departures at node i in time-slot k, respectively.

Without loss of generality, we assume that only links with a non-zero queue length can be activated. Let $M_l = 1$ if a matching $M \in \mathcal{M}$ contains link l, and $M_l = 0$ otherwise. Let $H_M(k)$ be the number of time-slots in which matching M is selected as a schedule up to time-slot k. We set by convention that $A_i(0) = 0$, $D_i(0) = 0$, and $D_i(-1) = 0$ for all $i \in V$. The queueing equations of the system are as follows:

$$Q_i(k) = Q_i(0) + A_i(k) - D_i(k-1), \tag{2}$$

$$D_i(k) = \sum_{M \in \mathcal{M}} \sum_{\tau=1}^{k} \sum_{l \in L(i)} M_l \cdot (H_M(\tau) - H_M(\tau - 1)), \tag{3}$$

$$\sum_{M \in \mathcal{M}} H_M(k) = k. \tag{4}$$

Next, we define system stability as follows.

Definition 1. *The network is rate stable if with probability one,*

$$\lim_{k \to \infty} \frac{D_l(k)}{k} = \lambda_l, \tag{5}$$

for all $l \in E$ and for any arrival processes satisfying Eq. (1).

Note that we consider *rate stability* for ease of presenting our main ideas. Strong stability can similarly be derived if we make stronger assumptions on the arrival processes [20].

We define the *throughput region* of a scheduling algorithm as the set of arrival rate vectors for which the network remains rate stable under this algorithm. Further, we define the *optimal throughput region*, denoted by Λ^*, as the union of the throughput regions of all possible scheduling algorithms. A scheduling algorithm is said to have an *efficiency ratio* γ if it can support any arrival rate vector λ strictly inside $\gamma\Lambda^*$. Clearly, we have $\gamma \in [0, 1]$. In particular, a scheduling algorithm with an efficiency ratio $\gamma = 1$ is *throughput-optimal*, i.e., it can stabilize the system under any feasible load. We also give the definition of another important region Ψ as follows:

$$\Psi \triangleq \{\lambda \mid \lambda_i \leq 1 \text{ for all } i \in V\}. \tag{6}$$

Clearly, we have $\Lambda^* \subseteq \Psi$ since at most one packet can be transmitted from or to a node in each time-slot.

As we mentioned earlier, in the settings without future packet arrivals, the performance metric of interest is the evacuation time, defined as the time interval needed for evacuating all the initial packets. Let T^P denote the evacuation time of scheduling algorithm P, and let $\mathcal{X}'$ denote the minimum evacuation time over all the possible algorithms, (which is equivalent to the *chromatic index* in the graph coloring literature). A scheduling algorithm is said to have an *approximation ratio* η if it has an evacuation time no greater than $\eta\mathcal{X}'$ in any network graph with any finite number of initial packets. Clearly, we have $\eta \geq 1$. In particular, a scheduling algorithm with an approximation ratio $\eta = 1$ is *evacuation-time-optimal*.

III. Node-based Service-Balanced Algorithm

In this section, we propose a novel Node-based Service-Balanced (NSB) scheduling algorithm and analyze its performance. Specifically, we prove that NSB guarantees an approximation ratio no worse (or no greater) than $3/2$ for the evacuation time (Subsection III-B) and an efficiency ratio no worse (or no smaller) than $2/3$ for the throughput (Subsection III-C). Further, we show that NSB is both throughput-optimal and evacuation-time-optimal in bipartite graphs (Subsection III-D). To the best of our knowledge, none of the existing algorithms strike a more balanced performance guarantees than NSB in both dimensions of throughput and evacuation time.

A. Algorithm

We start by introducing Maximum Vertex-weighted Matching (MVM), which will be a key component of the NSB algorithm. Let w_i denote the weight of node i. We will later describe how to assign the node weights. Also, let $w(M) \triangleq \sum_{i: M \cap L(i) \neq \emptyset} w_i$ denote the weight of matching M, i.e., the sum of the weight of the nodes matched by M. A matching M^* is called an MVM if it has the maximum weight among all the matchings, i.e., $M^* \in \operatorname{argmax}_{M \in \mathcal{M}} w(M)$. A very useful property of MVM is that if there exists a matching that matches the s heaviest nodes (i.e., the s nodes with the largest weights), then an MVM will match all of them too. This property is a result of [21]. We restate it in Lemma 1, which will be frequently used in the proofs of our main results.

Lemma 1 (Lemma 6 of [21]). *For any given positive integer $s \leq n$, suppose there exists a matching that matches the s heaviest nodes, then an MVM also matches all of them too.*

Now, we describe the operations of the NSB algorithm. We first give some additional definitions and notations. Recall that $Q_i(k)$ denote the workload of node i in time-slot k. Let $\Delta(k) \triangleq \max_{i \in V} Q_i(k)$ denote the largest node queue length in time-slot k. A node i is called *critical* in time-slot k if it has the largest queue length, i.e., $Q_i(k) = \Delta(k)$; a node i is called *heavy* in time-slot k if its queue length is no smaller than $\frac{n-1}{n}\Delta(k)$. (Our results also hold if we replace $\frac{n-1}{n}$ with any $\alpha \in [\frac{n-1}{n}, 1)$.) It will later become clearer why such a threshold is chosen. We use $\mathcal{C}(k)$ and $\mathcal{H}(k)$ to denote the set of critical nodes and the set of heavy nodes in time-slot k, respectively. Let $R_i(k) \triangleq D_i(k) - D_i(k-1)$ denote whether node i is matched in time-slot k or not, and define

$$U_i(k) \triangleq \begin{cases} R_i(k-1)R_i(k-2) & \text{if } k = 3k'+2; \\ R_i(k-1) & \text{otherwise,} \end{cases} \tag{7}$$

for some integer $k' \geq 0$. Note that $U_i(k)$ is either 1 or 0, depending on whether node i was matched in the previous time-slot (or in both of the previous two time-slots) or not.

Then, in time-slot k, we assign a weight to node i as

$$w_i \triangleq \begin{cases} Q_i(k)(2 - U_i(k)) & \text{if } i \in \mathcal{H}(k); \\ Q_i(k) & \text{otherwise,} \end{cases} \tag{8}$$

and the NSB algorithm finds an MVM based on the assigned node weight w_i's in every time-slot. Note that links with a zero queue length will not be considered when MVM is computed. According to Eq. (8), all the nodes are divided into two groups: the heavy nodes that were not scheduled in the previous time-slot(s) have a weight that is twice their workload, and all the other nodes have a weight equal to their workload. Within the same group, a node with a larger workload has a larger weight.

Remark: The key intuition that the NSB algorithm can provide provable guarantees for both evacuation time and throughput is the following: in the no-arrival settings, it ensures that all the critical nodes will be scheduled at least twice within every three consecutive time-slots, and in the settings with arrivals, it ensures that all the critical nodes in the fluid limit (which are among the heavy nodes in the original stochastic system) will be scheduled at least twice within every three consecutive time-slots. This comes from the following properties of NSB: 1) it results in the desired priority or ranking of the nodes by assigning the node weights according to Eq. (8); 2) it finds an MVM based on the assigned node weights in every time-slot, and MVM guarantees to match all the heaviest nodes whenever possible (Lemma 1).

B. Evacuation Time Performance

In this subsection, we analyze the evacuation time performance of NSB in the settings without arrivals. The main result is presented in Theorem 1.

Theorem 1. *The NSB algorithm has an approximation ratio no greater than $3/2$ for the evacuation time performance.*

Proof. Recall that for a given network with initial packets waiting to be transmitted, $\mathcal{X}'$ denotes the minimum evacuation time, and T^{NSB} denotes the evacuation time of NSB. We want to show $T^{\text{NSB}} \leq \frac{3}{2}\mathcal{X}'$. Recall that $\Delta(0)$ denotes the maximum node queue length in time-slot 0. If $\Delta(0) = 1$, this is trivial as $T^{\text{NSB}} = \mathcal{X}'$. Now, suppose $\Delta(0) \geq 2$. Then, the result follows immediately from 1) Proposition 1: under NSB, the maximum node queue length decreases by at least two within three consecutive time-slots, i.e., $T^{\text{NSB}} \leq \frac{3}{2}\Delta(0)$, and 2) an obvious fact: it takes at least $\Delta(0)$ time-slots to drain all the packets over the links incident to a node with maximum queue length, i.e., $\Delta(0) \leq \mathcal{X}'$. □

Therefore, it remains to prove Proposition 1.

Proposition 1. *Suppose the maximum node queue length in a given time-slot is no smaller than two. Under the NSB algorithm, the maximum node queue length decreases by at least two within three consecutive time-slots.*

We first restate a useful result of [22] in Lemma 2, which will be used in the proof of Proposition 1.

Lemma 2 (Theorem 1 of [22]). *Let G be a loopless multigraph (i.e., no edge connects a node to itself) with maximum degree Δ. Let G_Δ denote the subgraph of G induced by all the nodes having maximum degree. If G_Δ is bipartite, then there exists a matching in G that matches every node of maximum degree.*

Now, we are ready to prove Proposition 1. The proof follows a similar argument as in the proof for MVM in [15]. Hence, we give a sketch of the proof below and provide the detailed proof in our online technical report [23] for completeness.

Note that in any given time-slot, the network together with the present packets can be mapped to a loopless multigraph, where each multi-edge corresponds to a packet waiting to be transmitted over the link connecting the end nodes of the multi-edge. By slightly abusing the notation, we use $G(k)$ to denote the multigraph at the beginning of time-slot k. Hence, the degree of node i in $G(k)$ is equivalent to the node queue length $Q_i(k)$, and the maximum node degree of $G(k)$ is equivalent to $\Delta(k)$. We also let $M(k)$ denote the matching found by NSB in time-slot k. Now, consider three consecutive time-slots $\{p, p+1, p+2\}$, where $p = 3k'$ for some integer $k' \geq 0$. Suppose $\Delta(p) \geq 2$ at the beginning of time-slot p. Then, we want to show that under NSB, the maximum degree will be at most $\Delta(p) - 2$ at the end of time-slot $p+2$. We proceed the proof in two steps: 1) we first show that the maximum degree will decrease by at least one in the first two time-slots p and $p+1$ (i.e., the maximum degree will be at most $\Delta(p) - 1$ at the end of time-slot $p+1$), and then, 2) show that if the maximum degree decreases by exactly one in the first two time-slots (i.e., the maximum degree is $\Delta(p) - 1$ at the end of time-slot $p+1$), the maximum degree must also decrease by one in time-slot $p+2$ and becomes $\Delta(p) - 2$ at the end of time-slot $p+2$. Lemmas 1 and 2 will be used in the proof of Proposition 1.

C. Throughput Performance

Next, we analyze the throughput performance of the NSB algorithm in the settings with arrivals. The main result is presented in Theorem 2.

Theorem 2. *The NSB algorithm has an efficiency ratio no smaller than* $2/3$ *for the throughput performance.*

We start our analysis by constructing the fluid limit model as in [20], [24]. To begin with, we extend the processes $Y = Q, D, H$ to continuous time $t \geq 0$ as $Y(t) = Y(\lfloor t \rfloor)$. Hence, $Q(t)$, $D(t)$ and $H(t)$ are right continuous with left limits. Then, we construct the Markov process $X = \{X(t) : t \geq 0\}$ that describes system dynamics as

$$X(t) = \{Q(\tau), D(\tau), H(\tau), \tau \in [\max\{t-2, 0\}, t]\}. \quad (9)$$

Then, using the techniques of Theorem 4.1 of [24], we can show that for almost all sample paths and for all positive sequences $x_r \to \infty$, there exists a subsequence x_{r_j} with $x_{r_j} \to \infty$ as $j \to \infty$ such that the following convergence holds uniformly over compact (*u.o.c.*) intervals of time t:

$$\frac{A_i(x_{r_j} t)}{x_{r_j}} \to \lambda_i t, \forall i \in V, \quad (10)$$

$$\frac{Q_i(x_{r_j} t)}{x_{r_j}} \to q_i(t), \forall i \in V, \quad (11)$$

$$\frac{D_i(x_{r_j} t)}{x_{r_j}} \to d_i(t), \forall i \in V, \quad (12)$$

$$\frac{H_M(x_{r_j} t)}{x_{r_j}} \to h_M(t), \forall M \in \mathcal{M}. \quad (13)$$

We present the fluid model equations as follows:

$$q_i(t) = q_i(0) + \lambda_i t - d_i(t), \forall i \in V, \quad (14)$$

$$\tfrac{d}{dt} d_i(t) = \sum_{M \in \mathcal{M}} \sum_{l \in L(i)} M_l \cdot \tfrac{d}{dt} h_M(t), \forall i \in V, \quad (15)$$

$$\sum_{M \in \mathcal{M}} h_M(t) = t. \quad (16)$$

Any such limit (q, d, h) is called a *fluid limit*. Note that $q_i(\cdot), d_i(\cdot)$ and $h_i(\cdot)$ are absolutely continuous functions and are differentiable at almost all times $t \geq 0$ (called *regular* times). Taking the derivative of both sides of (14) and substituting (15) into it, we obtain

$$\begin{aligned} \tfrac{d}{dt} q_i(t) &= \lambda_i - \tfrac{d}{dt} d_i(t) \\ &= \lambda_i - \sum_{M \in \mathcal{M}} \sum_{l \in L(i)} M_l \cdot \tfrac{d}{dt} h_M(t). \end{aligned} \quad (17)$$

Next, borrowing the results of [25], we give the definition of *weak stability* and state Lemma 3, which establishes the connection between rate stability of the original system and weak stability of the fluid model.

Definition 2. *The fluid model of a network is weakly stable if for every fluid model solution* (q, d, h) *with* $q(0) = 0$, *one has* $q(t) = 0$ *for all regular times* $t \geq 0$.

Lemma 3. *A network is rate stable if the associated fluid model is weakly stable.*

We now prove Theorem 2 using fluid limit techniques.

Proof. We want to show that given any arrival rate vector λ strictly inside $\frac{2}{3}\Lambda^*$, the system is rate stable under the NSB algorithm. Note that λ is also strictly inside $\frac{2}{3}\Psi$ (i.e., $\lambda_i < \frac{2}{3}$ for all $i \in V$) since $\Lambda^* \subseteq \Psi$. We define $\epsilon \triangleq \min_{i \in V}(\frac{2}{3} - \lambda_i)$. Clearly, we must have $\epsilon > 0$.

To show rate stability of the original system, it suffices to show weak stability of the fluid model due to Lemma 3. We start by defining the following Lyapunov function:

$$V(q(t)) = \max_{i \in V} q_i(t). \quad (18)$$

Since $V(q(t))$ is a non-negative function, given $q(0) = 0$, in order to show $V(q(t)) = 0$ and thus $q(t) = 0$ for all regular times $t \geq 0$, it suffices to show that if $V(q(t)) > 0$ for $t > 0$, then $V(q(t))$ has a negative drift and decreases at least at a given rate. Then, the stability of the fluid model follows from Definition 2. Therefore, we want to show that for all regular times $t > 0$, if $V(q(t)) > 0$, we have $\frac{d}{dt} V(q(t)) \leq -\epsilon$.

We first fix time t and let $q_{max} = V(q(t)) = \max_{i \in V} q_i(t)$. Define the set of critical nodes in the fluid limits at time t as

$$\mathcal{C} \triangleq \{i \in V \mid q_i(t) = q_{max}\}. \quad (19)$$

Also, let $\hat{q}_{max}$ be the largest queue length in the fluid limits among the remaining nodes, i.e., $\hat{q}_{max} = \max_{i \in V \setminus \mathcal{C}} q_i(t)$. Since the number of nodes is finite, we have $\hat{q}_{max} < q_{max}$. Choose β small enough such that $\hat{q}_{max} < q_{max} - 3\beta$ and $\beta < \frac{1}{2n-1} q_{max}$. This implies

$$q_{max} - \beta > \frac{n-1}{n}(q_{max} + \beta). \quad (20)$$

Recall that $q(t)$ is absolutely continuous. Hence, there exists a small δ such that the queue lengths in the fluid limits satisfy the following conditions for all times $\tau \in (t, t+\delta)$:

(C1) $q_i(\tau) \in (q_{max} - \frac{\beta}{2}, q_{max} + \frac{\beta}{2})$ for all $i \in \mathcal{C}$;

(C2) $q_i(\tau) < q_{max} - \frac{5\beta}{2}$ for all $i \notin \mathcal{C}$.

Let x_{r_j} be a positive subsequence for which the convergence to the fluid limit holds. Consider j large enough such that $|\frac{Q_i(x_{r_j}\tau)}{x_{r_j}} - q_i(\tau)| < \frac{\beta}{2}$ for all $\tau \in (t, t+\delta)$. Considering the neighborhood around time t, we define a set of consecutive time-slots in the original system as $T \triangleq \{\lceil x_{r_j} t \rceil, \lceil x_{r_j} t \rceil + 1, \ldots, \lfloor x_{r_j}(t+\delta) \rfloor\}$, which corresponds to the scaled time interval $(t, t+\delta)$ in the fluid limits.

Lemma 4 states that NSB, all the critical nodes at scaled time t in the fluid limits will be scheduled at least twice within every three consecutive time-slots of T.

Lemma 4. *Under the NSB algorithm, all the nodes in* $\mathcal{C}$ *will be scheduled at least twice within every three consecutive time-slots of* T.

We will prove Lemma 4 immediately after the current proof. For now, assume Lemma 4 holds. Then, for all $i \in \mathcal{C}$, we have

$$\begin{aligned} &\sum_{M \in \mathcal{M}} \sum_{l \in L(i)} M_l \cdot (H_M(x_{r_j}(t+\delta)) - H_M(x_{r_j} t)) \\ &\geq \frac{2}{3}(\lfloor x_{r_j}(t+\delta) \rfloor - \lceil x_{r_j} t \rceil - 3), \end{aligned} \quad (21)$$

and therefore, we have

$$\begin{aligned}
&\sum_{M\in\mathcal{M}}\sum_{l\in L(i)} M_l \cdot \tfrac{d}{dt}h_M(t)\\
&=\lim_{\delta\to 0}\sum_{M\in\mathcal{M}}\sum_{l\in L(i)} M_l \cdot \frac{h_M(t+\delta)-h_M(t)}{\delta}\\
&\overset{(a)}{=}\lim_{\delta\to 0}\lim_{j\to\infty}\sum_{M\in\mathcal{M}}\sum_{l\in L(i)} \frac{M_l\cdot(H_M(x_{r_j}(t+\delta))-H_M(x_{r_j}t))}{x_{r_j}\delta}\\
&\overset{(b)}{\geq}\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\frac{2}{3}(\lfloor x_{r_j}(t+\delta)\rfloor-\lceil x_{r_j}t\rceil-3)}{x_{r_j}\delta}=\frac{2}{3},
\end{aligned} \tag{22}$$

where (a) and (b) are from Eqs. (13) and (21), respectively. Then, it follows from Eq. (17) that for all $i \in \mathcal{C}$, we have $\frac{d}{dt}q_i(t) \leq \lambda_i - \frac{2}{3} \leq -\epsilon$.

Also, from conditions (C1) and (C2), every node $i \notin \mathcal{C}$ has a queue length strictly smaller than that of a critical node in $\mathcal{C}$ for the entire duration $(t, t+\delta)$. Thus, it follows that $\frac{d}{dt}V(q(t)) \leq -\epsilon$, which implies that the fluid model is weakly stable. This completes the proof by applying Lemma 3. □

Now, it remains to prove Lemma 4. The proof will rely on a novel application of graph-factor theory – Lemma 6. Hence, before proving Lemma 4, we introduce some additional notations and prove Lemma 6 first.

By slightly abusing the notations, we also use $G = (V, E)$ to denote a multigraph. Let $g = [g_v : v \in V]$ and $f = [f_v : v \in V]$ be vectors of positive integers satisfying

$$0 \leq g_v \leq f_v \leq d_G(v), \text{ for all } v \in V, \tag{23}$$

where $d_G(v)$ is the degree of node v in G counting multiplicities, and a loop associated with node v counts 2 towards the degree of v. A (g, f)-factor is a subgraph F of G with

$$g_v \leq d_F(v) \leq f_v, \text{ for all } v \in V. \tag{24}$$

Note that if vectors g, f satisfy $g_v, f_v \in \{0, 1\}$ for all $v \in V$, then the edges of a (g, f)-factor form a matching over G.

Let $G_{g=f}$ denote the subgraph of G induced by nodes v for which $g_v = f_v$. Let $[x]^+ \triangleq \{x, 0\}$. We restate a result of [26] in Lemma 5, which will be used in the proof of Lemma 6.

Lemma 5 (Theorem 1.3 of [26], Property I). *Let multigraph G and vectors g, f be given. Suppose $G_{g=f}$ is bipartite. Then, G has a (g, f)-factor if and only if for all $S \subseteq V$,*

$$\sum_{v\in S} f_v \geq \sum_{v\notin S}[g_v - d_{G-S}(v)]^+. \tag{25}$$

Next, we state Lemma 6 below and provide its proof.

Lemma 6. *Let G be a multigraph with maximum degree Δ. Let $Z \subseteq V$ be a subset of nodes, and let G_Z denote the subgraph of G induced by Z. Suppose the following conditions are satisfied: (i) all the nodes of Z are heavy nodes, i.e., $Z \subseteq \{v \in V \mid d_G(v) \geq \frac{n-1}{n}\Delta\}$, and (ii) G_Z is bipartite. Then, there exists a matching over G that matches every node of Z.*

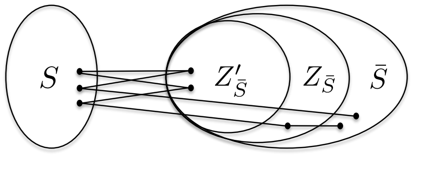


Fig. 1: An illustration for the relationship of the sets in the proof of Lemma 6.

Proof. We will use Lemma 5 to prove Lemma 6. Assume $n > 1$. First, construct vectors g, f in the following way: set $g_v = f_v = 1$ for node $v \in Z$, and set $g_v = 0$, $f_v = 1$ otherwise. Clearly, we have $G_{g=f} = G_Z$, and the edges of a (g, f)-factor form a matching over graph G that matches every node of Z.

Suppose (i) all the nodes of Z are heavy nodes, and (ii) G_Z is bipartite. Now, consider any subset of nodes $S \subseteq V$, we want to show that Eq. (25) is satisfied. Let $\bar{S} = V \backslash S$ be the complementary set of S. Let $Z_{\bar{S}} = Z \cap \bar{S}$, and let $Z'_{\bar{S}} = \{v \in Z_{\bar{S}} \mid \text{ all the neighboring nodes of } v \text{ are in } S\}$. The relationship of these sets is illustrated in Fig. 1. It is easy to see that $\sum_{v\in S} f_v = |S|$ as $f_v = 1$ for all $v \in V$. Also, a little thought gives $\sum_{v\notin S}[g_v - d_{G-S}(v)]^+ = |Z'_{\bar{S}}|$. This is because any node $v \notin S$ must belong to one of the following three cases: 1) If $v \notin Z$, we have $g_v = 0$, and thus, $[g_v - d_{G-S}(v)]^+ = 0$; 2) If $v \in Z'_{\bar{S}}$, we have $g_v = 1$ and $d_{G-S}(v) = 0$, and thus, $[g_v - d_{G-S}(v)]^+ = 1$; 3) If $v \in Z_{\bar{S}} \backslash Z'_{\bar{S}}$, we have $g_v = 1$ and $d_{G-S}(v) \geq 1$, and thus, $[g_v - d_{G-S}(v)]^+ = 0$.

Hence, in order to show Eq. (25), it remains to show $|S| \geq |Z'_{\bar{S}}|$. We prove it by contradiction. Suppose $|S| < |Z'_{\bar{S}}|$. Since S and $Z'_{\bar{S}}$ are disjoint, we have $|S| + |Z'_{\bar{S}}| \leq n$, and thus, $|S| < n$. We let $d_G(Z) = \sum_{i\in Z} d_G(i)$ denote the total degree of a subset of nodes $Z \subseteq V$ in G and state three obvious facts:

(F1) $d_G(Z'_{\bar{S}}) \geq \frac{n-1}{n}\Delta|Z'_{\bar{S}}|$;
(F2) $d_G(S) \leq \Delta|S|$;
(F3) $d_G(Z'_{\bar{S}}) \leq d_G(S)$.

Note that (F1) is from the fact that every node of Z has a degree no smaller than $\frac{n-1}{n}\Delta$, (F2) is trivial, and (F3) is from the fact that all the neighboring nodes of nodes in $Z'_{\bar{S}}$ belong to S. Then, by combining the above facts, we obtain

$$\frac{n-1}{n} \leq \frac{|S|}{|Z'_{\bar{S}}|}. \tag{26}$$

This further implies $|Z'_{\bar{S}}| - |S| \leq \frac{|S|}{n-1} \leq 1$, as $|S| < n$. Hence, we must have $|Z'_{\bar{S}}| = |S| + 1$, because $|Z'_{\bar{S}}| > |S|$. Substituting this back into Eq. (26), we derive $\frac{n-1}{n} \leq \frac{|S|}{|S|+1}$. This implies $|S| \geq n-1$, and thus $|S| + |Z'_{\bar{S}}| \geq 2n-1$, which contradicts the fact that $|S| + |Z'_{\bar{S}}| \leq n$.

Therefore, we have $|S| \geq |Z'_{\bar{S}}|$ and Eq. (25) is satisfied. From Lemma 5, there exists a (g, f)-factor, and from the way we construct vectors g and f, there exists a matching over graph G that matches every node of Z. □

Remark: Lemma 6 generalizes Lemma 2 and is of critical importance in proving the guaranteed throughput performance of NSB in general graphs. Moreover, it will play a key role in establishing both throughput optimality and evacuation time optimality for NSB in bipartite graphs (see Section III-D).

Now, we are ready to prove Lemma 4.

Proof of Lemma 4. Recall that $\mathcal{C}$ is the set of critical nodes in the fluid limits at time t (Eq. (19)). We want to show that under the NSB algorithm, all the nodes in $\mathcal{C}$ will be scheduled at least twice within every three consecutive time-slots of T.

First, recall that j is large enough such that $|\frac{Q_i(x_{r_j}\tau)}{x_{r_j}} - q_i(\tau)| < \frac{\beta}{2}$ for all times $\tau \in (t, t+\delta)$. Hence, from condition (C1) and (C2), the queue lengths in the original system satisfy the following conditions for all time-slots $k \in T$:

(C1*) $Q_i(k) \in x_{r_j}(q_{max} - \beta, q_{max} + \beta)$ for all $i \in \mathcal{C}$;

(C2*) $Q_i(k) < x_{r_j}(q_{max} - 2\beta)$ for all $i \notin \mathcal{C}$.

On account of condition (C1*) and Eq. (20), all the nodes in $\mathcal{C}$ are heavy nodes in all the time-slots of T, i.e., $Q_i(k) \geq \frac{n-1}{n}\Delta(k)$ for all $i \in \mathcal{C}$ and for all $k \in T$.

Note that in any time-slot, the network together with the present packets can be modeled as a multigraph, where each multi-edge corresponds to a packet. Recall that we use $G(k)$ to denote the multigraph at the beginning of time-slot k. Note that if there are no packets waiting to be transmitted over a link, no multi-edge connecting the end nodes of this link will appear in $G(k)$. Also, recall that $M(k)$ denotes the matching found by the NSB algorithm in time-slot k. Now, consider three consecutive time-slots $\{p, p+1, p+2\}$ of T, where $p = 3k'$ for some integer $k' \geq 0$. We want to show that under the NSB algorithm, every node in $\mathcal{C}$ will get scheduled in at least two time-slots of $\{p, p+1, p+2\}$. We proceed the proof in two steps: 1) we first show that all the nodes in $\mathcal{C}$ will be scheduled at least once in the first two time-slots p and $p+1$, and 2) then show that all the nodes in $\mathcal{C}$ that were scheduled exactly once in the first two time-slots, will get scheduled in time-slot $p+2$.

We start with step 1). Let $\mathcal{C}'$ denote the set of nodes in $\mathcal{C}$ that were not scheduled in time-slot p. It is a trivial case if $\mathcal{C}' = \emptyset$. Therefore, suppose $\mathcal{C}' \neq \emptyset$, i.e., there exists at least one node in $\mathcal{C}$ that was not scheduled in time-slot p. Then, it suffices to show that all the nodes in $\mathcal{C}'$ must be scheduled in time-slot $p+1$ under the NSB algorithm. Note that matching $M(k)$ must be a maximal matching over $G(k)$ for every time-slot k. Since $M(p)$ is a maximal matching, the nodes in $\mathcal{C}'$ must form an independent set at the beginning of time-slot $p+1$, excluding the multi-edges corresponding to the new packet arrivals at the beginning of time-slot $p+1$.

Note that it is a trivial case if $|\mathcal{C}'| = 1$. So we consider the case of $|\mathcal{C}'| \geq 2$ and prove it by contradiction. Suppose there exist two adjacent nodes $i, j \in \mathcal{C}'$. Then, none of the edges incident to either i or j was in matching $M(p)$. This implies that the multi-edge between i and j could be added to matching $M(p)$ in time-slot p, which, however, contradicts the fact that $M(p)$ is a maximal matching. Therefore, the nodes in $\mathcal{C}'$ must form an independent set at the beginning of time-slot $p+1$. Clearly, the subgraph induced by all the nodes in $\mathcal{C}'$ (which form an independent set) is bipartite. Note that conditions (C1*) and (C2*) still hold even without accounting for the new packet arrivals. Then, by Lemma 6, there exists a matching that matches all the nodes in $\mathcal{C}'$ at the beginning of time-slot $p+1$ before new packet arrivals. Clearly, such a matching still exists even if the multi-edges corresponding to the newly arrived packets in time-slot $p+1$ are added to the grpah. Note that $M(p+1)$ is an MVM over $G(p+1)$ with the assigned weights (as in Eq. (8)). Now, if all the nodes in $\mathcal{C}'$ are among the ones with the heaviest weights, then it implies from Lemma 1 that matching $M(p+1)$ also matches all the nodes in $\mathcal{C}'$. This is indeed true due to conditions (C1*) and (C2*), as well as the weight assignments in Eq. (8): every node in $\mathcal{C}'$ was not scheduled in time-slot p, and thus has a weight larger than $2x_{r_j}(q_{max} - \beta)$, while any node in $V \backslash \mathcal{C}'$ cannot have a weight larger than $\max\{2x_{r_j}(q_{max} - 2\beta), x_{r_j}(q_{max} + \beta)\}$.

Now, we prove step 2). Let $\mathcal{C}''$ denote the set of nodes in $\mathcal{C}$ that were scheduled exactly once in time-slots p and $p+1$. We want to show that all the nodes in $\mathcal{C}''$ will get scheduled in time-slot $p+2$. Note that all the nodes in $\mathcal{C}''$ are among the ones with the heaviest weights. This is true due to conditions (C1*) and (C2*), as well as the weight assignments in Eq. (8): every node in $\mathcal{C}''$ was scheduled exactly once in time-slots p and $p+1$, and thus has a weight larger than $2x_{r_j}(q_{max} - \beta)$, while any node in $V \backslash \mathcal{C}''$ cannot have a weight larger than $\max\{2x_{r_j}(q_{max} - 2\beta), x_{r_j}(q_{max} + \beta)\}$. Further, let $G_{\mathcal{C}''}$ denote the subgraph induced by all the nodes in $\mathcal{C}''$ at the beginning of time-slot $p+2$, excluding all the multi-edges corresponding to the packets that arrived in time-slot $p+1$ and $p+2$. If $G_{\mathcal{C}''}$ is bipartite, then again by Lemmas 6 and 1, following the same argument as in step 1), we can show that all the nodes in $\mathcal{C}''$ are matched by $M(p+2)$ in time-slot $p+2$. Therefore, it remains to show that $G_{\mathcal{C}''}$ is bipartite.

Next, we prove that $G_{\mathcal{C}''}$ is bipartite by contradiction. Suppose $G_{\mathcal{C}''}$ contains an odd cycle, say C. Then, no two adjacent nodes of C were matched by $M(p+1)$ in time-slot $p+1$. This is true due to the following. Suppose there exist two adjacent nodes of C, say i and j, matched by $M(p+1)$. Since i and j are in $\mathcal{C}''$, both of them were matched exactly once in time-slots p and $p+1$ from the definition of $\mathcal{C}''$. This implies that both i and j were not matched in time-slot p, i.e., we have $i, j \in \mathcal{C}'$. However, given that i and j are adjacent, this contradicts what we have shown earlier – the nodes in $\mathcal{C}'$ form an independent set. Therefore, no two adjacent nodes of C were matched by $M(p+1)$ in time-slot $p+1$. This, along with the fact that cycle C is of odd size, implies that cycle C must contain two adjacent nodes that were not matched by $M(p+1)$ in time-slot $p+1$. This further implies that the multi-edge between these two adjacent nodes can be added to $M(p+1)$, which contradicts the fact that $M(p+1)$ is a maximal matching over $G(p+1)$. Therefore, the induced subgraph $G_{\mathcal{C}''}$ must be bipartite. This completes the proof of step 2) and that of Lemma 4. □

D. Optimality in Bipartite Graphs

So far, we have focused on analyzing the performance of NSB in general graphs and have shown that NSB guarantees an approximation ratio no greater than $3/2$ for the evacuation time and an efficiency ratio no smaller than $2/3$ for the throughput. Considering the fact that NSB makes decisions based on the node workload and does not handle odd-size cycles (which could also be bottlenecks), the theoretical performance guarantees achieved by NSB are quite remarkable. We believe that NSB will perform better if odd-size cycles do not form bottlenecks. This is also observed in our simulation results in Section V. In this subsection, we will show that NSB is both throughput-optimal and evacuation-time-optimal in bipartite graphs. This result is stated in Theorem 3, which will be proved using Lemmas 1 and 6.

Theorem 3. *The NSB algorithm is both throughput-optimal and evacuation-time-optimal in bipartite graphs.*

Proof. We first consider evacuation time optimality. Recall that for a given network with initial packets waiting to be transmitted, $\Delta(0)$ denotes the maximum node degree at the very beginning, and $\mathcal{X}'$ denotes the minimum evacuation time. If the underlying network is bipartite, there are no odd-size cycles, and we have $\mathcal{X}' = \Delta(0)$. Hence, as long as the maximum degree decreases by one in every time-slot, the minimum evacuation time can be achieved. By Lemma 6 and the fact that the underlying network graph is bipartite, we know that in every time-slot, there exists a matching that matches all the heavy nodes, including all the critical nodes, and by Lemma 1, NSB also matches all the critical nodes in every time-slot. Therefore, NSB is evacuation-time-optimal.

Then, we consider throughput optimality. The analysis follows a similar line of argument as in the proof of Theorem 2 for general graphs. The difference is that we now consider any given arrival rate vector λ strictly inside Λ^* (and thus strictly inside Ψ, i.e., $\lambda_i < 1$ for all $i \in V$) and need to show that all the nodes in $\mathcal{C}$ will be scheduled in every time-slot of T (rather than in two of three consecutive time-slots of T as in Lemma 4 for general graphs). This is true from Lemmas 6 and 1. We omit the detailed proof here. $\square$

The NSB algorithm has a complexity of $O(m\sqrt{n}\log n)$, as the complexity of finding an MVM is $O(m\sqrt{n}\log n)$ [21]. One important question is whether we can develop lower-complexity algorithms that provide the same performance guarantees. We answer this question in the next section.

IV. A Lower-Complexity NSB Algorithm

Through the analysis for the NSB algorithm, we obtain the following insights: In order to achieve the same performance guarantees as NSB, what really matters is the priority or the ranking of the nodes, rather than the exact weight of the nodes. If we assign the node weights in a way that they are bounded integers, and the nodes still have the desired priority or ranking as in the NSB algorithm, we can develop a new algorithm with a lower complexity. Thanks to the results of [27], [28], an $O(m\sqrt{n})$-complexity implementation of MVM[3] can be derived if the maximum node weight is a bounded integer independent of n and m.

[3]This can be done by setting the weight of an edge to the sum of the weight of its two end nodes and finding an MWM based on the new edge weights using the techniques developed in [27], [28].

Next, we propose such an algorithm, called the Lower-Complexity NSB (LC-NSB). Recall that $U_i(k)$ indicates whether node i was matched in the previous time-slot (or in both of the previous two time-slots, as defined in Eq. (7)). Also, recall that $\mathcal{C}(k)$ and $\mathcal{H}(k)$ denote the set of critical nodes and the set of heavy nodes in time-slot k, respectively. In time-slot k, we assign a weight to node i as

$$w_i \triangleq \begin{cases} 5 - 2U_i(k) & \text{if } i \in \mathcal{C}(k); \\ 4 - 2U_i(k) & \text{if } i \in \mathcal{H}(k)\backslash\mathcal{C}(k); \\ 1 & \text{otherwise.} \end{cases} \tag{27}$$

Then, the LC-NSB algorithm finds an MVM based on the assigned node weight w_i's in every time-slot. Note that LC-NSB has a very similar way of assigning the node weights as NSB. However, the key difference is that we now divide all the nodes into five priority groups by assigning the node weights only based on whether it is a heavy (or critical) node and whether it was scheduled in the previous time-slot(s), while in the NSB algorithm, the actual workload is used in the weight assignments. This slight yet crucial change leads to a lower-complexity algorithm with the same performance guarantees. Note that in Eq. (27), we give a higher priority to the critical nodes in order to guarantee the evacuation time performance.

Theorem 4. *The LC-NSB algorithm has an approximation ratio no greater than* $3/2$ *for the evacuation time and has an efficiency ratio no smaller than* $2/3$ *for the throughput.*

Moreover, the LC-NSB algorithm is both throughput-optimal and evacuation-time-optimal in bipartite graphs.

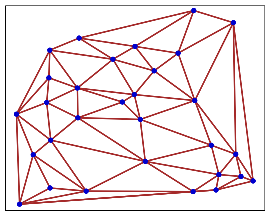
Proof. The proof follows similarly as in the proofs for the NSB algorithm and is thus omitted. $\square$

Remark: Although LC-NSB can provide the same performance guarantees as NSB, we would expect that LC-NSB may have (slightly) worse empirical performance compared to NSB, since NSB has a more fine-grained priority differentiation among all the nodes. We indeed make such observations in our simulation results in Section V. In order to improve the empirical performance, we can introduce more priority groups for the non-heavy nodes under LC-NSB rather than all being in the same priority group (of weight 1 as in Eq. (27)). As long as the number of priority groups is a bounded integer independent of n and m, the complexity remains $O(m\sqrt{n})$.

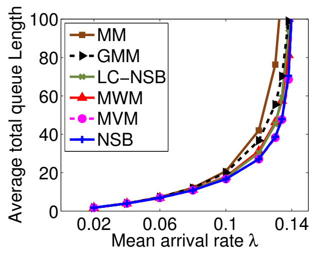
V. Numerical Results

In this section, we conduct numerical experiments to elucidate our theoretical results. We also compare the empirical performance of our proposed NSB and LC-NSB algorithms with several most relevant algorithms as listed in Table I.

We first focus on a randomly generated triangular mesh topology with 30 nodes and 79 links as shown in Fig. 2a. The simulation codes are written in C++. We assume that the arrivals are *i.i.d.* over all the links with unit capacity. The mean arrival rate of each link is λ, and the instantaneous



(a) Topology



(b) Performance comparison

Fig. 2: Simulations for a triangular mesh topology.

arrivals to each link follow a Poisson distribution in each time-slot. In Fig. 2b, we plot the average total queue length in the system against the arrival rate λ. We consider several values of λ as indicated in Fig. 2b. For each value of λ, the average total queue length is an average of 10 independent simulations. Each individual simulation runs for a period of 10^5 time-slots. We compute the average total queue length by excluding the first 5×10^4 time-slots in order to remove the impact of the initial transient state. Note that this network topology contains odd-size cycles. Hence, our proposed NSB and LC-NSB only guarantee to achieve $2/3$ of the optimal throughput. However, the simulation results in Fig. 2b show that NSB and LC-NSB algorithms both empirically achieve the optimal throughput performance. This is because the odd-size cycles do not form the bottlenecks in this setting. We also observe that NSB and MVM perform very closely and exhibit the best delay performance.

We have obtained more simulation results by considering different network topologies. The results show that NSB and LC-NSB both have very good empirical performance in all the scenarios we consider. Due to space limitations, we provide more simulation results in our online technical report [23].

VI. Conclusion

In this paper, we studied the link scheduling problem for multi-hop wireless networks with single-hop flows and focused on designing efficient online algorithms with provably guaranteed throughput and evacuation time performance. We developed two node-based service-balanced algorithms and showed that none of the existing algorithms strike a more balanced performance guarantees than our proposed algorithms in both dimensions of throughput and evacuation time. An interesting future direction is to consider more general scenarios with multi-hop flows, where it becomes much more challenging to provide provably good evacuation time performance.

References

[1] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[2] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

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

[4] G. R. Gupta, "Delay efficient control policies for wireless networks," *Ph.D. thesis, Purdue University*, 2009.

[5] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in wirelessHART networks," in *European Control Conference (ECC)*. IEEE, 2009, pp. 4320–4325.

[6] K. Kang, K.-J. Park, L. Sha, and Q. Wang, "Design of a crossbar VOQ real-time switch with clock-driven scheduling for a guaranteed delay bound," *Real-Time Systems*, vol. 49, no. 1, pp. 117–135, 2013.

[7] 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, no. 12, pp. 1936–1948, 1992.

[8] I. Holyer, "The NP-completeness of edge-coloring," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 718–720, 1981.

[9] M. Stiebitz, D. Scheide, B. Toft, and L. M. Favrholdt, *Graph Edge Coloring: Vizing's Theorem and Goldberg's Conjecture*. John Wiley & Sons, 2012.

[10] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-Layer congestion control in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302–315, 2006.

[11] C. Joo, X. Lin, and N. Shroff, "Greedy Maximal Matching: Performance Limits for Arbitrary Network Graphs Under the Node-exclusive Interference Model," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2734–2744, 2009.

[12] X. Wu, R. Srikant, and J. Perkins, "Scheduling Efficiency of Distributed Greedy Scheduling Algorithms in Wireless Networks," *IEEE Transactions on Mobile Computing*, pp. 595–605, 2007.

[13] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proceedings of IEEE INFOCOM*, 1998, pp. 792–799.

[14] V. Tabatabaee and L. Tassiulas, "MNCM: a critical node matching approach to scheduling for input buffered switches with no speedup," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 294–304, 2009.

[15] B. Ji and J. Wu, "Node-based scheduling with provable evacuation time," in *Proceedings of IEEE CISS*, March 2015.

[16] J. Kahn, "Asymptotics of the chromatic index for multigraphs," *Journal of Combinatorial Theory, Series B*, vol. 68, no. 2, pp. 233–254, 1996.

[17] L. Georgiadis, G. S. Paschos, L. Libman, and L. Tassiulas, "Minimal evacuation times and stability," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 931–945, 2015.

[18] L. Tassiulas and J. Joung, "Performance measures and scheduling policies in ring networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 5, pp. 576–584, 1995.

[19] L. Tassiulas, "Cut-through switching, pipelining, and scheduling for network evacuation," *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 88–97, 1999.

[20] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying service rates," *Probability in the Engineering and Informational Sciences*, vol. 18, no. 02, pp. 191–217, 2004.

[21] T. H. Spencer and E. W. Mayr, "Node weighted matching," in *Automata, Languages and Programming*. Springer, 1984, pp. 454–464.

[22] R. P. Anstee and J. R. Griggs, "An application of matching theory of edge-colourings," *Discrete Mathematics*, vol. 156, no. 1, pp. 253–256, 1996.

[23] B. Ji, G. R. Gupta, and Y. Sang, "Node-based service-balanced scheduling for provably guaranteed throughput and evacuation time performance," December 2015. [Online]. Available: http://arxiv.org/abs/1512.02328

[24] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," *The Annals of Applied Probability*, pp. 49–77, 1995.

[25] M. Bramson, *Stability of queueing networks*. Springer, 2008.

[26] R. P. Anstee, "Simplified existence theorems for (g, f)-factors," *Discrete applied mathematics*, vol. 27, no. 1, pp. 29–38, 1990.

[27] C.-C. Huang and T. Kavitha, "Efficient algorithms for maximum weight matchings in general graphs with small edge weights," in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2012, pp. 1400–1412.

[28] S. Pettie, "A simple reduction from maximum weight matching to maximum cardinality matching," *Information Processing Letters*, vol. 112, no. 23, pp. 893–898, 2012.