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

Yu Sang, Gagan R. Gupta, and Bo Ji, *Member, IEEE*

Abstract—This paper focuses on the design of *provably efficient online* link scheduling algorithms for multi-hop wireless networks. We consider single-hop traffic and the one-hop interference model. The objective is twofold: 1) *maximizing the throughput* when the flow sources continuously inject packets into the network, and 2) *minimizing 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 propose a novel Node-based Service-Balanced (NSB) online scheduling algorithm. NSB aims to give scheduling opportunities to heavily congested nodes in a balanced manner, by maximizing the total weight of the scheduled nodes in each scheduling cycle, where the weight of a node is determined by its workload and whether the node was scheduled in the previous scheduling cycle(s). 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.

Index Terms—Wireless scheduling, node-based approach, service-balanced, throughput, evacuation time, provable performance guarantees

1 Introduction

Resource allocation is an important problem in wireless networks. Various functionalities at 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 at 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 [2], [3] 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* traffic under the *one-hop* interference model[1]. 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 also of critical importance due to the following reasons. First, draining all existing packets within a minimum amount of time is a major concern in the settings without future arrivals. One practical example is environmental monitoring using wireless sensor networks, where all measurement data periodically generated by different nodes at the same time, need to be transmitted to one or multiple sinks for further processing. Second, evacuation time is also highly correlated with the delay performance in the settings with arrivals. For example, evacuation-time optimality is a necessary condition for the strongest delay notion of sample-path optimality [8]. Third, it is quite relevant to timely transmission of delay-sensitive data traffic (e.g., deadline-constrained packet delivery) [9], [10].

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 [8] 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 [11] and is now well understood, evacuation time is much less studied. In the no-arrival setting, the minimum evacuation time problem is equivalent to the *multigraph*[2] *edge coloring problem* due to the following: each multi-edge corresponds to a packet

1. The packets of single-hop traffic traverse only one link before leaving the system. The one-hop interference model is also called the node-exclusive or the primary interference model, where two links sharing a common node cannot be active at the same time. This model can properly represent practical wireless networks based on Bluetooth or FH-CDMA technologies [3], [4], [5], [6], [7].

- *Y. Sang and B. Ji are with the Department of CIS, Temple University, Philadelphia, PA 19122. E-mail: {yu.sang, boji}@temple.edu.*
- *G.R. Gupta is with AT&T Labs, San Ramon, CA 94583. E-mail: gagan.gupta@iitdalumni.com.*

Manuscript received 18 May 2017; revised 14 Sept. 2017; accepted 17 Nov. 2017. Date of publication 27 Nov. 2017; date of current version 29 June 2018. (Corresponding author: Bo Ji.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2017.2777828

2. In a multigraph, more than one edge, called multi-edge, is allowed between two nodes.

waiting to be transmitted over the link between the nodes of the multi-edge; each color corresponds to a feasible schedule (or a matching); finding the chromatic index (i.e., 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) is equivalent to minimizing the time for evacuating all the packets by finding a matching at a time. Since edge-coloring is a classic NP-hard problem [12], a rich body of research has focused on developing approximation algorithms (see [13] for a good survey). These algorithms employ a popular recoloring technique 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. This is because the complexity would become impractically high when there are a large number of packets (or multi-edges) in the network. 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.

Most existing online scheduling algorithms either make scheduling decisions based on the link load (such as Maximum Weighted Matching (MWM) [11] and Greedy Maximal Matching (GMM) [6], [7]) or are load agnostic (such as Maximal Matching (MM) [6], [14]). While these algorithms are throughput-efficient, none of them can guarantee an approximation ratio better (or smaller) than 2 for the evacuation time [8]. In contrast, several prior work [8], [15], [16], [17] 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) [8]. 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 [18]. Hence, giving a higher priority to scheduling nodes with heavy workload leads to better evacuation time performance, while the link-based approach that fails to respect this crucial fact results in unsatisfactory evacuation time performance.

While the node-based approach seems quite promising, the scheduling performance of the node-based algorithms is not well understood, and the existing studies are mostly limited to bipartite graphs [8], [15], [16]. Very recent work of [17] 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 [19], the authors study the connection between throughput and (expected) minimum evacuation time, but no algorithms with provable performance guarantees are provided. The work of [9], [20], [21] considers the minimum evacuation time problem for multi-hop traffic in some special scenarios (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.

TABLE 1
Performance Comparison of NSB and LC-NSB with Several Most Relevant Online Algorithms in the Literature

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	$O(m\sqrt{n}\log n)$	?	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

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 2.) 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.

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-time-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* is adopted to analyze how NSB schedules the heavy nodes (Lemma 4).

Second, from the performance analysis 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 $O(m\sqrt{n})$.

In Table 1, we summarize the guaranteed performance of NSB and LC-NSB as well as several most relevant online algorithms in the literature. *As can be seen, 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 2. Then, we propose the NSB algorithm and analyze its performance in Section 3. A lower-complexity NSB algorithm with the same performance guarantees is developed in Section 4. Finally, we conduct numerical experiments in Section 5 and make concluding remarks in Section 6. Some detailed proofs are provided in Section 7.

2 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. The node count and the link count are denoted by $n = |V|$ and $m = |E|$, 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 $L(i) \triangleq \{l \in E \,|\, i \text{ is an end node of link } l\}$. 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 of links, L, that satisfies 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.

As in several previous work (e.g., [6], [22], [23], [24], [25]), we focus on link scheduling at the MAC layer, and thus we only consider single-hop traffic. 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 (including 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,

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

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, $\lim_{k \to \infty} A_i(k)/k = \lambda_i$ 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 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 M is selected as a schedule up to time-slot k. We set by convention that $A_i(k) = 0$ and $D_i(k) = 0$ for all $i \in V$ and for all $k \leq 0$. 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,*

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

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 [26].

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 network under any feasible load. We also define another important region Ψ by considering bottlenecks formed by the nodes:

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

Clearly, we have $\Lambda^* \subseteq \Psi$ because at most one packet can be transmitted from or to a node in each time-slot. Similarly, any odd-size cycle Z could also be a bottleneck because at most $(|Z| - 1)/2$ out of the $|Z|$ links of the odd-size cycle can be scheduled at the same time. For example, the total arrival rate summed over all edges of a triangle must not exceed $1/3$ because at most one out of the three links of the triangle can be scheduled in each time-slot. For the theoretical analyses, we consider bottlenecks formed only by the nodes, which is sufficient for deriving our analytical results. We provide more discussions about odd-size cycles in Sections 3.4 and 5.1.

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 draining 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 possible algorithms. 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*.

In this paper, the goal is to develop efficient online link scheduling algorithms that can simultaneously provide provably good performance in both dimensions of throughput and evacuation time, measured through the efficiency ratio (i.e., the larger the value of γ, the better) and the approximation ratio (i.e., the smaller the value of η, the better), respectively. Note that the throughput performance has been extensively studied under quite general models in the literature (see [2], [3] and references therein), where multi-hop traffic, general interference models, and time-varying

TABLE 2
Summary of Notations

Symbol	Meaning
G	Network topology as an undirected graph
V	Set of nodes
E	Set of links
n	Number of nodes
m	Number of links
$L(i)$	Set of links touching node $i \in V$
M	A matching over G
$\mathcal{M}$	Set of all the matchings over G
λ_l	Mean arrival rate of link l
λ_i	Mean arrival rate of node i
Λ^*	Optimal throughput region
Ψ	An outer bound of Λ^*; see Eq. (6)
γ	Efficiency ratio (for throughput performance)
T^P	Evacuation time of scheduling algorithm P
$\mathcal{X}'$	Minimum evacuation time
η	Approximation ratio (for evacuation time performance)
$A_l(k)$	Cumulative arrivals at link l up to time-slot k
$A_i(k)$	Cumulative arrivals at node i up to time-slot k
$D_l(k)$	Cumulative departures at link l up to time-slot k
$D_i(k)$	Cumulative departures at node i up to time-slot k
$Q_l(k)$	Queue length at link l in time-slot k
$Q_i(k)$	Workload at node i in time-slot k
$H_M(k)$	Number of time-slots in which matching M is selected as a schedule up to time-slot k
$\Delta(k)$	Largest node workload in time-slot k
$\mathcal{C}(k)$	Set of critical nodes in time-slot k
$\mathcal{H}(k)$	Set of heavy nodes in time-slot k
$R_i(k)$	Whether node i is matched in time-slot k or not
$U_i(k)$	See Eq. (7)
$w_i(k)$	Weight of node i in time-slot k
$w(M)$	Weight of matching M

channels (which can model mobility and fading) have been considered. However, the evacuation time performance is much less understood. As we have mentioned earlier, even in the setting we consider (assuming single-hop traffic, the one-hop interference model, and fixed link capacities), the minimum evacuation time problem is already very challenging (i.e., NP-hard). Considering multi-hop traffic adds another layer of difficulty. This is mainly because of the dependence between the upstream and downstream queues, since the arrival process to an intermediate queue is no longer exogenous, but instead, it is the departure process of its previous-hop queue. In addition to link scheduling, we also need to decide which flow's packets will be transmitted when a link is activated. This further complicates the minimum evacuation time problem.

For quick reference, we summarize the key notations of this paper in Table 2.

3 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 (Section 3.2) and an efficiency ratio no worse (or no smaller) than $2/3$ for the throughput (Section 3.3). Further, we show that NSB is both throughput-optimal and evacuation-time-optimal in bipartite graphs (Section 3.4). 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.

3.1 Algorithm

We start by introducing Maximum Vertex-weighted Matching (MVM) [8], [27], 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., $M^* \in \operatorname{argmax}_{M\in\mathcal{M}} w(M)$. In [27], a very useful property of MVM is proven. We restate it in Lemma 1, which will be frequently used in the proofs of our main results.

Algorithm 1. Node-Based Service-Balanced (NSB)

1: In each time-slot k:
2: **for** each node $i \in V$ **do**
3: Assign node weight $w_i(k)$ based on Eq. (8)
4: **end for**
5: Exclude links $l \in E$ with $Q_l(k) = 0$
6: Find an MVM M^* over G with node weight $w_i(k)$'s, i.e.,

$$M^* \in \operatorname*{argmax}_{M\in\mathcal{M}} w(M) \triangleq \sum_{i:M\cap L(i)\neq\emptyset} w_i(k)$$

7: **for** each link $l \in E$ **do**
8: **if** $M_l^* = 1$ **then**
9: Transmit one packet over link l
10: **else**
11: No transmission over link l
12: **end if**
13: **end for**

Lemma 1 (Lemma 6 of [27]). *For any positive integer $s \leq n$, suppose that there exists a matching that matches the s heaviest nodes. Then, an MVM matches all of these s nodes too.*

Now, we consider frames each consisting of three consecutive time-slots and describe the operations of the NSB algorithm. We first give some additional definitions and notations. Recall that $Q_i(k)$ denotes 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 $(n-1)/n \cdot \Delta(k)$. (Our results also hold if we replace $(n-1)/n$ with any $\alpha \in [(n-1)/n, 1)$.) 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

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

where k' is the frame index. Note that $U_i(k)$ is either 1 or 0 and will be used in Eq. (8) to determine whether a node needs to get a higher scheduling priority in time-slot k or

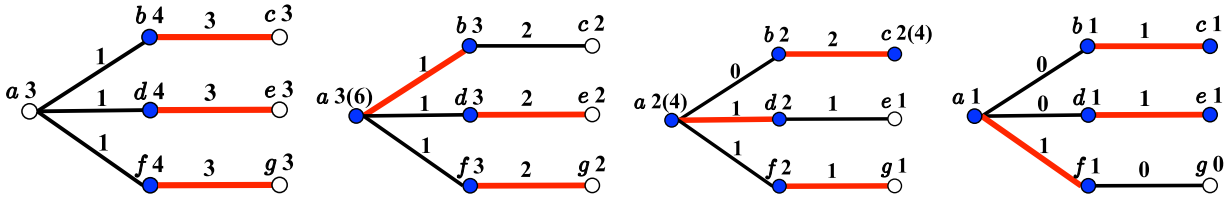


Fig. 1. An illustration of the operations of NSB in four time-slots. The network setting is presented in Fig. 1a, where there are seven nodes $\{a, b, , g\}$. In each subfigure, the number above each link denotes the number of packets waiting to be transmitted over that link at the beginning of each time-slot. For simplicity, we assume no future packet arrivals in this example. The node degree (i.e., the sum of queue lengths over all the links touching the node) and the node weight (in the parenthese) are both labeled after the node name; however, the node weight is not labeled if it is equal to the node degree. The heavy nodes are highlighted in blue. Take Fig. 1b for example: the heavy nodes are a, b, d, and f; node a has a degree of 3 and has a weight of 6; node b has a degree and a weight both equal to 3. Note that although both nodes a and b are heavy nodes in time-slot 1, the weight of a equals twice the node degree because it was not scheduled in time-slot 1 (i.e., $U_a(2) = 0$). The thick red lines denote the links activated in each time-slot.

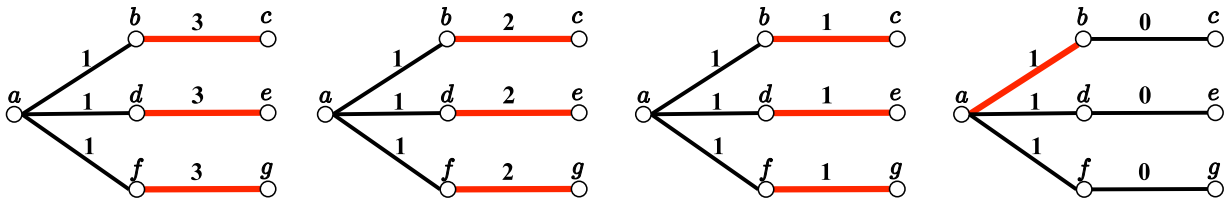


Fig. 2. An illustration of the operations of MWM in four time-slots. The labels are similar to that in Fig. 1. The node degree and the node weight are not labeled because they are irrelevant under MWM.

not. Specifically, the weight of a heavy node i is doubled if $U_i(k) = 0$ (i.e., this heavy node i did not receive enough service in the previous time-slot(s)) such that node i has a higher priority of being scheduled in time-slot k.

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

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

The NSB algorithm finds an MVM [27] based on the assigned node weight $w_i(k)$'s in each time-slot. Note that links with a zero queue length will not be considered when MVM is computed. According to Eq. (8), the nodes are divided into two groups: the heavy nodes that were not scheduled in the previous time-slot(s) have a weight twice their workload, and all the other nodes have a weight equal to their workload. Within each group, a node with a larger workload has a larger weight. To help illustrate the operations of the NSB algorithm, we provide its pseudo code in Algorithm 1.

In addition, to help the reader better understand the operations of the NSB algorithm, we also present a simple example in Fig. 1, which demonstrates the system evolution within four time-slots. Note that a certain tie-breaking rule is applied in this example. Using a different tie-breaking rule, different schedules could be selected. For instance, in time-slot 0 we may activate link (a, b) instead of link (b, c) along with links (d, e) and (f, g). However, tie-breaking rules do not affect our analysis. As can be seen from Fig. 1, NSB drains all initial packets by the end of time-slot 3. For comparison, using the same example, in Fig. 2 we also demonstrate the system evolution under the MWM algorithm. Similarly, we observe that after time-slot 3, MWM needs two more time-slots to completely drain all initial packets. In the next section, we will use a similar example to show that a link-based algorithm like MWM needs about twice as much time as that of a node-based algorithm like NSB to evacuate all initial packets in the network.

3.2 Evacuation Time Performance

In this section, 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 $T^{\text{NSB}} \leq 3/2 \cdot \mathcal{X}'$. Recall that $\Delta(0)$ denotes the maximum node queue length in time-slot 0. If $\Delta(0) = 1$, this is trivial as $T^{\text{NSB}} = \mathcal{X}' = 1$. Now, suppose $\Delta(0) \geq 2$. Then, the result follows immediately from 1) Proposition 1 (stated after this proof): under NSB, the maximum node queue length decreases by at least two within each frame, i.e., $T^{\text{NSB}} \leq 3/2 \cdot \Delta(0)$, 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., $\Delta(0) \leq \mathcal{X}'$. □

Next, we state a key proposition (Proposition 1) used for proving Theorem 1.

Proposition 1. *Consider any frame. Suppose the maximum node queue length is no smaller than two at the beginning of a frame. Under the NSB algorithm, the maximum node queue length decreases by at least two by the end of the frame.*

We provide the detailed proof of Proposition 1 in Section 7.1 and give a sketch of the proof below. Note that in any time-slot, the network together with the present packets

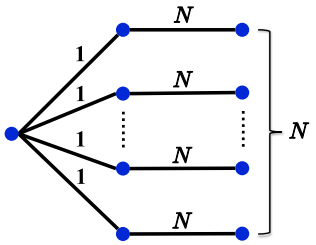


Fig. 3. A network with $2N+1$ nodes, where N is a positive integer. The number above each link denotes the number of initial packets waiting to be transmitted over that link.

can be represented as 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. We use $G(k)$ to denote the multigraph at the beginning of time-slot k, and use $M(k)$ to denote the matching found by the NSB algorithm in 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 equal to $\Delta(k)$. Now, consider any frame k' consisting of three consecutive time-slots $\{p, p+1, p+2\}$, where $p = 3k'$. Suppose that the maximum node queue length is no smaller than two at the beginning of frame k', i.e., $\Delta(p) \geq 2$ at the beginning of time-slot p. Then, we want to show that under the NSB algorithm, 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 $p+1$), then the maximum degree must decrease by one in time-slot $p+2$, and becomes $\Delta(p)-2$ at the end of time-slot $p+2$.

Remark. The key intuition that the NSB algorithm can provide provable evacuation time performance is that all the critical nodes are ensured to be scheduled at least twice within each frame (Proposition 1). 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, and if a critical node was not scheduled in the first time-slot of the frame, then in the second time-slot of the frame, MVM guarantees to match all such critical nodes (Lemma 1); 3) similarly, if a critical node was not scheduled in both of the first two time-slots of the frame, then MVM guarantees to match all such critical nodes in the third time-slot of the frame.

In addition, we use an example to illustrate why a link-based algorithm like MWM could result in a bad evacuation time performance. Consider the network topology presented in Fig. 3. Since MWM aims to maximize the total weight summed over all scheduled links in each time-slot, it will choose the matching consisting of all the edges with N packets. This pattern repeats until all links have one packet (after $N-1$ time-slots). Then, it takes additional N or $N+1$ time-slots to drain all the remaining packets, depending on the tie-breaking rule. This results in inefficient schedules that consist of one link only about half the time, and thus, requires a total of $2N-1$ or $2N$ time-slots. Another link-based algorithm GMM performs similarly. On the other hand, as we will show in Section 3.4, NSB is evacuation-time-optimal in this example and needs only $\Delta = N+1$ time-slots since the graph is bipartite. The system evolution under NSB and MWM for a special case of $N=3$ can be found in Figs. 1 and 2, respectively.

3.3 Throughput Performance

Next, we analyze the throughput performance of NSB 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 will employ fluid limit techniques [26], [28], [29] to prove Theorem 2. Fluid limit techniques are useful for two main reasons: (i) it removes irrelevant randomness in the original stochastic system such that the considered system becomes deterministic and thus, the analysis can be simplified; (ii) the algorithm exhibits some special properties in the fluid limit (e.g., Lemma 3), which do not exist in the original stochastic system.

Before proving Theorem 2, we construct the fluid model and state some definitions and a lemma that will be used in the proof. First, we extend the process $Y = A, Q, D, H$ to continuous time $t \geq 0$ by setting $Y(t) = Y(\lfloor t \rfloor)$. Hence, $A(t)$, $Q(t)$, $D(t)$, and $H(t)$ are right continuous with left limits. Then, using the techniques of [29], 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:

$$\frac{A_i(x_{r_j}t)}{x_{r_j}} \to \lambda_i t, \text{for all } i \in V, \tag{9}$$

$$\frac{Q_i(x_{r_j}t)}{x_{r_j}} \to q_i(t), \text{ for all } i \in V, \tag{10}$$

$$\frac{D_i(x_{r_j}t)}{x_{r_j}} \to d_i(t), \text{ for all } i \in V, \tag{11}$$

$$\frac{H_M(x_{r_j}t)}{x_{r_j}} \to h_M(t), \text{ for all } M \in \mathcal{M}. \tag{12}$$

Since the proof of the above convergence is standard, we provide the proof in the appendix for completeness.

Next, we present the fluid model equations as follows:

$$q_i(t) = q_i(0) + \lambda_i t - d_i(t), \text{ for all } i \in V, \tag{13}$$

$$\tfrac{d}{dt}d_i(t) = \sum_{M \in \mathcal{M}} \sum_{l \in L(i)} M_l \cdot \tfrac{d}{dt}h_M(t), \text{ for all } i \in V, \tag{14}$$

$$\sum_{M \in \mathcal{M}} h_M(t) = t. \tag{15}$$

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 (13) and substituting (14) into it, we obtain

$$\begin{aligned}\tfrac{d}{dt}q_i(t); &= \lambda_i - \tfrac{d}{dt}d_i(t)\\ &= \lambda_i - \sum_{M\in\mathcal{M}}\sum_{l\in L(i)} M_l \cdot \tfrac{d}{dt}h_M(t).\end{aligned} \tag{16}$$

Borrowing the results of [29], we give the definition of *weak stability* and state Lemma 2, 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 2 (Theorem 3 of [29]). *A network is rate stable if the associated fluid model is weakly stable.*

We are now ready to prove Theorem 2.

Proof of Theorem 2. We want to show that given any arrival rate vector λ strictly inside $2/3\cdot\Lambda^*$, the system is rate stable under the NSB algorithm. Note that λ is also strictly inside $2/3\cdot\Psi$ (i.e., $\lambda_i < 2/3$ for all $i\in V$) since $\Lambda^*\subseteq\Psi$. We define $\epsilon\triangleq\min_{i\in V}(2/3-\lambda_i)$. 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 2. We start by defining the following Lyapunov function:

$$V(q(t)) = \max_{i\in V} q_i(t). \tag{17}$$

For any regular time $t\geq 0$, we define the drift of $V(q(t))$ as its derivative, denoted by $\frac{d}{dt}V(q(t))$.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. This is due to a simple result in Lemma 1 of [29].Therefore, we want to show that for all regular times $t>0$, if $V(q(t))>0$, then $\frac{d}{dt}V(q(t))\leq-\epsilon$.

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

$$\mathcal{C}\triangleq\{i\in V\,|\,q_i(t)=q_{\max}(t)\}. \tag{18}$$

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

$$q_{\max}(t)-\beta > \frac{n-1}{n}(q_{\max}(t)+\beta). \tag{19}$$

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 condition for all times $\tau\in(t,t+\delta)$and for all nodes $i\in V$:

$$q_i(\tau)\in(q_i(t)-\beta/2, q_i(t)+\beta/2). \tag{20}$$

This further implies that the following conditions hold:

(C1) $q_i(\tau)\in(q_{\max}(t)-\beta/2, q_{\max}(t)+\beta/2)$ for all $i\in\mathcal{C}$;
(C2) $q_i(\tau)<q_{\max}(t)-5\beta/2$ for all $i\notin\mathcal{C}$,
where (C2) is from Eq. (20) and $q_i(t)\leq\hat{q}_{\max}(t)<q_{\max}(t)-3\beta$ for all $i\notin\mathcal{C}$.

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

Lemma 3 states that NSB, all the critical nodes at scaled time t in the fluid limits will be scheduled at least twice within each frame of interval T. □

Lemma 3. *Under the NSB algorithm, all the nodes in $\mathcal{C}$ will be scheduled at least twice within each frame of interval T.*

We provide the proof of Lemma 3 in Section 7.2. For now, we assume that Lemma 3 holds. Note that interval T contains at least $(\lfloor x_{r_j}(t+\delta)\rfloor-\lceil x_{r_j}t\rceil-3)/3$ complete frames. Then, from Lemma 3, we have that for all $i\in\mathcal{C}$,

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

and therefore, we have

$$\begin{aligned}&\sum_{M\in\mathcal{M}}\sum_{l\in L(i)} M_l\cdot\tfrac{d}{dt}h_M(t)\\ &=\lim_{\delta\to0}\sum_{M\in\mathcal{M}}\sum_{l\in L(i)} M_l\cdot\frac{h_M(t+\delta)-h_M(t)}{\delta}\\ &\overset{(a)}{=}\lim_{\delta\to0}\lim_{j\to\infty}\sum_{M\in\mathcal{M}}\sum_{l\in L(i)}\frac{M_l\cdot(H_M(x_{r_j}(t+\delta))-H_M(x_{r_j}t))}{x_{r_j}\delta}\\ &\geq\overset{(b)}{\lim_{\delta\to0}}\lim_{j\to\infty}\frac{2/3\cdot(\lfloor x_{r_j}(t+\delta)\rfloor-\lceil x_{r_j}t\rceil-3)}{x_{r_j}\delta}\\ &=2/3,\end{aligned} \tag{22}$$

where (a) is from the convergence in Eq. (12) and (b) is from Eq. (21). Then, it follows from Eq. (16) that for all $i\in\mathcal{C}$, we have $\frac{d}{dt}q_i(t)\leq\lambda_i-2/3\leq-\epsilon$.

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, we have $\frac{d}{dt}V(q(t))\leq-\epsilon$, which implies that the fluid model is weakly stable. Then, we complete the proof by applying Lemma 2.

Remark. The key intuition that the NSB algorithm can provide provable throughput performance is that all the critical nodes in the fluid limit are ensured to be scheduled at least twice within each frame of interval T (Lemma 3). Similar to the provable evacuation time performance, this comes from the desired weight assignment (Eq. (8)) and an important property of MVM (Lemma 1).

As we mentioned earlier, the proof of Lemma 3 is provided in Section 7.2. However, we want to emphasize that the proof relies on a novel application of graph-factor theory, which is stated in Lemma 4. Lemma 4 is of critical importance in proving the guaranteed throughput

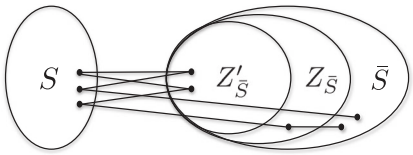


Fig. 4. An illustration for the relationship of the sets in the proof of Lemma 4.

performance of the NSB algorithm 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 3.4).

Next, we give some additional notations that are needed to state Lemma 4. By slightly abusing the notations, we also use $G = (V, E)$ to denote a multigraph, which is possibly not loopless. Let $d_G(v)$ be the degree of node v in G, where a loop associated with node v counts 2 towards the degree of v. Also, let G_Z denote the subgraph of G induced by a subset of nodes $Z \subseteq V$.

Lemma 4. *Let $G = (V, E)$ be a multigraph with maximum degree Δ. Consider a subset of nodes $Z \subseteq V$. Suppose that the following conditions are satisfied: (i) all the nodes of Z are heavy nodes, i.e., $Z \subseteq \{v \in V \,|\, d_G(v) \geq (n-1)/n \cdot \Delta\}$, and (ii) G_Z is bipartite. Then, there exists a matching of G that matches every node of Z.*

Proof. We first introduce some additional notations. Let $g = [g_v : v \in V]$ and $f = [f_v : v \in V]$ be vectors of positive integers satisfying

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

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 of G. Let $G_{g=f}$ denote the subgraph of G induced by nodes v for which $g_v = f_v$, and let $[x]^+ \triangleq \max\{x, 0\}$. We restate a result of [30] in Lemma 5, which will be used in the proof. □

Lemma 5 (Theorem 1.3 of [30], Property I). *Let multigraph G and vectors g, f be given. Suppose that $G_{g=f}$ is bipartite. Then, G has a (g, f)-factor if and only if for all node subsets $S \subseteq V$,*

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

We are now ready to prove Lemma 4. Suppose that (i) all the nodes of Z are heavy nodes and (ii) G_Z is bipartite. We construct vectors g, f by setting $g_v = f_v = 1$ for node $v \in Z$ and $g_v = 0$, $f_v = 1$ otherwise. With our constructed vectors g and f, not only a (g, f)-factor forms a matching of graph G, but this matching also matches every node of Z. Therefore, it suffices to show that G has a (g, f)-factor.

Next, we apply Lemma 5 to show that G has a (g, f)-factor. Note that $G_{g=f} = G_Z$ and G_Z is bipartite. Then, it remains to show that Eq. (25) is satisfied for any subset of nodes $S \subseteq V$. Let $\bar{S} = V \backslash S$ be the complementary set of S. Let $Z_{\bar{S}} = Z \cap \bar{S}$, and let $Z'_{\bar{S}} = \{v \in Z_{\bar{S}} \,|\,$ all the neighboring nodes of v are in $S\}$. The relationship of these sets is illustrated in Fig. 4. Clearly, we have $\sum_{v \in S} f_v = |S|$ 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 $v \notin Z$, then $g_v = 0$, and thus, $[g_v - d_{G-S}(v)]^+ = 0$;
(2) If $v \in Z_{\bar{S}} \backslash Z'_{\bar{S}}$, then $g_v = 1$ and $d_{G-S}(v) \geq 1$, which implies $[g_v - d_{G-S}(v)]^+ = 0$;
(3) If $v \in Z'_{\bar{S}}$, then $g_v = 1$ and $d_{G-S}(v) = 0$, which implies $[g_v - d_{G-S}(v)]^+ = 1$.

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. Then, we state three obvious facts:

(F1) $d_G(Z'_{\bar{S}}) \geq (n-1)/n \cdot \Delta |Z'_{\bar{S}}|$;
(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 $(n-1)/n \cdot \Delta$, (F2) is trivial, and (F3) is from the definition of $Z'_{\bar{S}}$ that all of its neighboring nodes belong to S. Then, by combining the above facts, we obtain

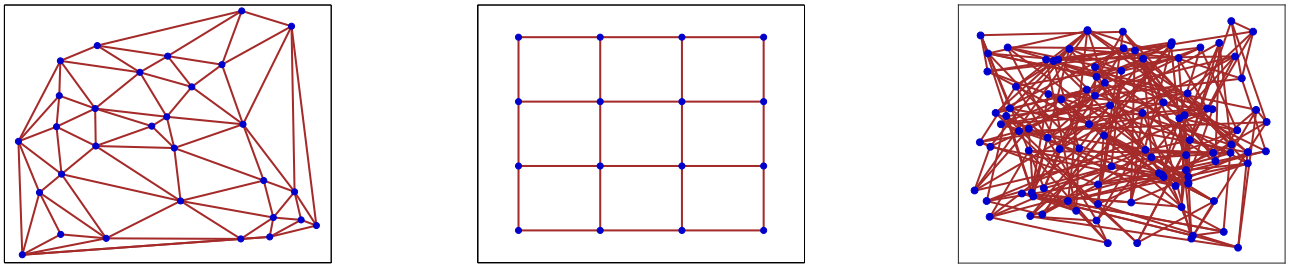
$$\frac{n-1}{n} \leq \frac{|S|}{|Z'_{\bar{S}}|}. \tag{26}$$

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

Therefore, we have $|S| \geq |Z'_{\bar{S}}|$, and thus, Eq. (25) is satisfied. Then, Lemma 5 implies that graph G has a (g, f)-factor, which forms a matching of graph G that matches every node of Z. This completes the proof.

3.4 Optimality in Bipartite Graphs

As we have explained in the introduction, the minimum evacuation time is lower bounded by the largest workload at the nodes and the odd-size cycles. Hence, both the nodes and the odd-size cycles could be bottlenecks. Ideally, it would be best to consider the workload of both the nodes and the odd-size cycles when making scheduling decisions. However, this may render the algorithm very complex because it is much more difficult to explicitly consider all the odd-size cycles in a graph. Hence, we have focused on designing the node-based algorithms (such as the NSB algorithm) that do not explicitly handle the odd-size cycles. Without considering all the bottlenecks, these algorithms may not be able to achieve the best achievable performance in general. However, the theoretical results of Theorems 1 and 2 are quite remarkable in the sense that even without considering the odd-size cycles (which could also be bottlenecks), the NSB algorithm can guarantee an approximation ratio no greater than $3/2$ for the evacuation time and an efficiency ratio no smaller than $2/3$ for the throughput. We believe that NSB will perform better if odd-size cycles do not form bottlenecks. This is also observed from our simulation results in Section 5.



(a) Triangular Mesh (b) 4×4 grid (c) Random topology

Fig. 5. Simulation topologies.

In this section, we will show that NSB is both throughput-optimal and evacuation-time-optimal in bipartite graphs, where there are no odd-size cycles. This result is stated in Theorem 3, whose proof needs to apply Lemmas 4 and 1 and follows a similar line of analysis to that of Theorems 1 and 2 for general graphs. The detailed proof is provided in the appendix for completeness.

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

Remark. The NSB algorithm has a complexity of $O(m\sqrt{n}\log n)$, as the complexity of finding an MVM is $O(m\sqrt{n}\log n)$ [27]. 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.

4 A Lower-Complexity NSB Algorithm

Through the analysis for the NSB algorithm, we obtain the following important 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. This insight comes from the following: Note that under the NSB algorithm, the weight of each node is only used in the MVM component (line 6 of Algorithm 1). In the performance analysis of NSB (i.e., Theorems 1, 2, and 3), the only property of MVM we use is Lemma 1, which is concerned about the s heaviest nodes (i.e., the s highest ranking nodes) rather than about the exact weight of the nodes. Hence, if we assign the node weights in a way such that, the weights are bounded integers and the nodes still have the desired priority or ranking as in the NSB algorithm, then we can develop a new algorithm with a lower complexity. Thanks to the results of [31], [32], an $O(m\sqrt{n})$-complexity implementation[3] of MVM can be derived if the maximum node weight is a bounded integer independent of n and m.

Next, we propose such an algorithm, called the Lower-Complexity NSB (LC-NSB). Similarly as in NSB, we consider frames each consisting of three consecutive time-slots. 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

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

Then, the LC-NSB algorithm finds an MVM based on the assigned node weight $w_i(k)$'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. The proof follows a similar line of analysis to that for the NSB algorithm and is provided in the appendix for completeness.

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.*

Remark. Although the LC-NSB algorithm 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 5. 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 $O(m\sqrt{n})$.

5 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 1.

5.1 Throughput Performance

To evaluate the throughput performance, we run the simulations on three different network topologies as shown in Fig. 5. We first focus on a randomly generated triangular

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 [31], [32].

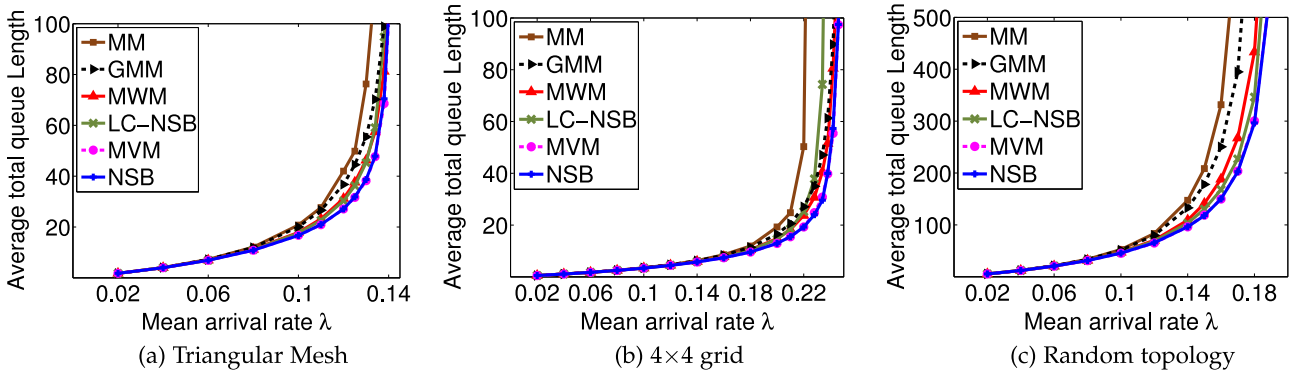


Fig. 6. Simulation results for Poisson arrivals.

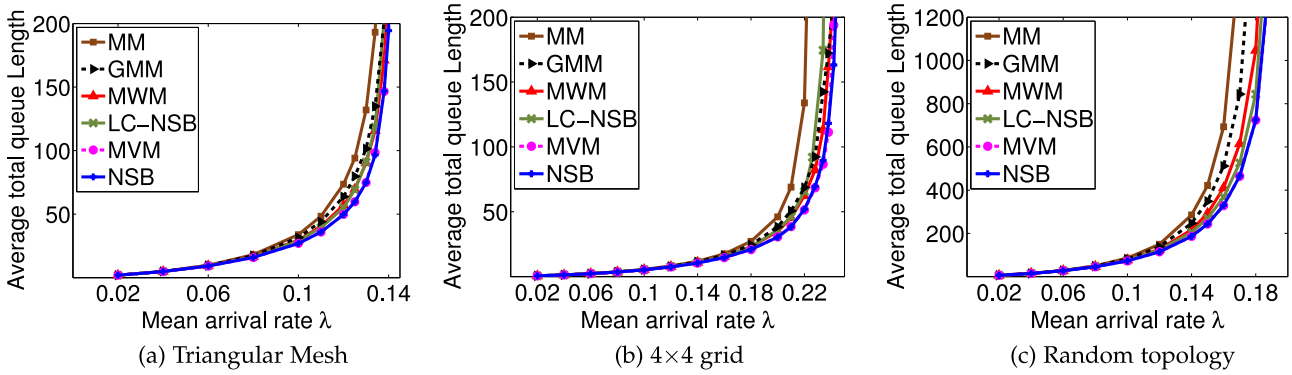


Fig. 7. Simulation results for file arrivals, where the file arrival probability is $p = 0.1$ and the file size follows Poisson distribution with mean λ/p.

mesh topology with 30 nodes and 79 links as shown in Fig. 5a. The simulations are implemented using 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 arrivals to each link follow a Poisson distribution in each time-slot. In Fig. 6a, 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. 6a. 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. 6a show that NSB and LC-NSB algorithms both empirically achieve the optimal throughput performance. This is because the odd-size cycles (i.e., all the triangles in this case) do not form the bottlenecks in this setting. For example, a triangle requires $\lambda \leq 1/3$ because at most one of its three links can be scheduled in each time-slot. However, in Fig. 5a there exists a node touched by seven links, which requires $\lambda \leq 1/7$. As the load λ increases, such a node will become congested sooner than any triangle and thus forms a scheduling bottleneck.

We also conduct simulations for a 4×4 grid topology with 16 nodes and 24 links (Fig. 5b) and a randomly generated dense topology with 100 nodes and 248 links (Fig. 5c). All the other simulation settings are the same as that for the triangular mesh topology. The simulation results are presented in Fig. 6. The observations we make are similar to that for the triangular mesh topology, except that in the grid topology, LC-NSB has higher delays when the mean arrival rate approaches the boundary of the optimal throughput region. This is due to the following two reasons: 1) Under LC-NSB, all the non-heavy nodes are in the same priority group of weight 1, so a non-heavy node with a large workload would have a similar chance of getting scheduled as a non-heavy node with a small workload; 2) In the grid topology, the bottleneck nodes (i.e., the four nodes in the center) are all adjacent. Note that in most cases, at least one of the bottleneck nodes is the critical node and will have the highest priority. If another bottleneck node is adjacent to the critical node but is not a heavy node, this bottleneck node will get a lower chance of being scheduled, since the critical node could be matched with other node. This inefficiency does not occur under NSB, since the node weights of the non-heavy nodes are their workload and are more fine-grained. Hence, a non-heavy node with a large workload will have a higher priority than a non-heavy node with a small workload.

In order to evaluate the throughput performance under different arrival patterns, we also run simulations for file arrivals. Specifically, the arrivals have the following pattern: in each time-slot, there is a file arrival with probability p, and no file arrival otherwise; the file size follows Poisson distribution with mean λ/p. The simulation results for $p = 0.1$ are presented in Fig. 7. Similar observations to that for Poisson arrivals can be made, except that all the algorithms have larger delays due to a more bursty arrival pattern. In addition, we also consider more realistic arrival patterns where the arrivals in each time-slot follow the Zipf law, which is commonly used to model the Internet traffic [33]. We assume a support of $[0, 1, \ldots, 999]$ for the Zipf

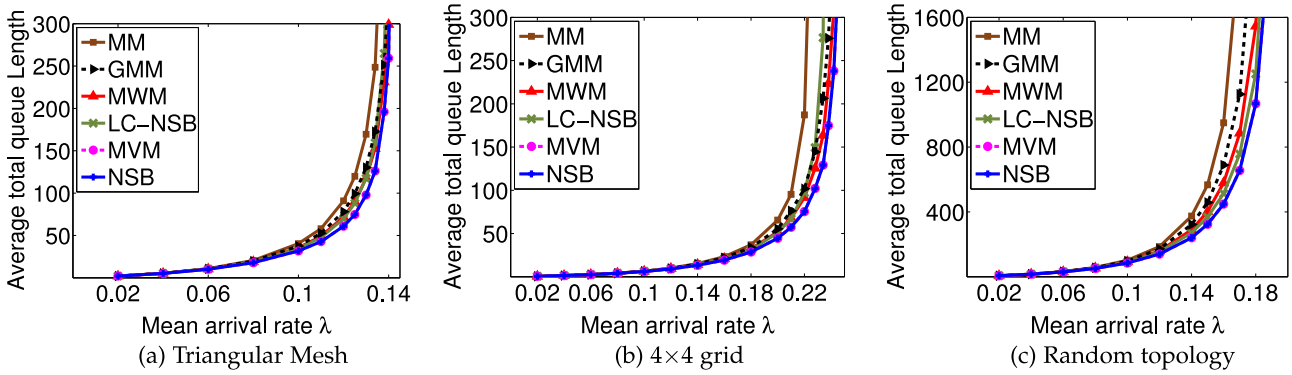


Fig. 8. Simulation results for Zipf arrivals with a support of $[0, 1, \ldots, 999]$.

distribution. The power exponent of the Zipf distribution is determined based on the mean arrival rate λ. The simulation results are presented in Fig. 8. The overall observations are again similar to that for the previous arrival patterns.

Finally, it is remarkable that in all simulation settings we consider, our proposed node-based algorithm NSB empirically achieves the best delay performance. When the traffic load is high, NSB even results in a significant reduction (10-30 percent) in the average delay performance compared to the link-based algorithms such as MWM (e.g., in Fig. 8c for a random topology with Zipf arrivals, the delay reduction is about 30 percent when $\lambda = 0.18$). Although NSB ties with another node-based algorithm MVM for the empirical delay performance, as we discussed in the introduction, the throughput performance of MVM is not well understood yet.

5.2 Evacuation Time Performance

In this section, we evaluate the evacuation time performance of our proposed algorithms. As we discussed in the introduction, the minimum evacuation time problem is equivalent to the classic multigraph edge coloring problem, for which there are common benchmarks. Therefore, we run simulations for six DIMACS benchmark instances [34] for the graph coloring problem. In addition, we also run simulations for three regular multigraphs, three random multigraphs, and one special graph (Fig. 3 with $N = 100$).

TABLE 3
Evacuation Time Performance for Six DIMACS Benchmark Graphs [34], Three Regular Multigraphs, and One Special Graph (Fig. 3 with $N = 100$)

Graph	MWM	GMM	MVM	NSB	LC-NSB	Δ
dsjc125.1	23	23	23	23	23	23
dsjc125.5	76	75	75	75	75	75
dsjc125.9	140	120	120	120	120	120
dsjc250.1	38	40	38	38	38	38
dsjc250.5	147	153	147	147	147	147
dsjc250.9	234	234	234	234	234	234
regm50.20	20	25	20	20	20	20
regm50.50	50	55	51	51	51	50
regm50.80	80	84	80	80	80	80
rand100.50	366	366	366	366	366	366
rand100.100	813	813	813	813	813	813
rand100.250	2161	2161	2161	2161	2161	2161
Fig. 3	199	199	101	101	101	101

In the table, Δ denotes the maximum node degree, dsjcX.Y denotes the label of the DIMACS benchmark graphs, regmX.Y denotes a regular mutigraph with X nodes and node degree Y, and randX.Y denotes a random topology in Fig. 5c with X nodes and the number of multi-edges at each link being uniformly distributed over the interval [0,Y].

The evacuation time performance for each of the considered algorithms under each graph is presented in Table 3. The simulation results show that all the algorithms have very similar evacuation time performance for the considered benchmark graphs and regular multigraphs, although they have different theoretical guarantees. For the special graph in Fig. 3, we can observe that the node-based algorithms exhibit a much better evacuation time performance compared to the link-based algorithms (e.g., MWM and GMM). Specifically, the link-based algorithms require about twice as much time as that of the node-based algorithms to evacuate all initial packets in the network.

6 Conclusion

In this paper, we studied the link scheduling problem for multi-hop wireless networks 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 important future direction is to consider more general models (which, e.g., allow for multi-hop traffic, general interference models, and time-varying channels). In such scenarios, it becomes much more challenging to provide provably good evacuation time performance.

7 Proofs

7.1 Proof of Proposition 1

We first restate a useful result of [35] in Lemma 6, which will be used in the proof of Proposition 1. Throughout the paper, we assume that the multigraph is loopless (i.e., there is no edge connecting a node to itself) unless explicitly mentioned.

Lemma 6 (Theorem 1 of [35]). *Let G be a loopless multigraph with maximum degree Δ. Let G_Δ denote the subgraph of G induced[4] by all the nodes having maximum degree. If G_Δ is bipartite, then there exists a matching over G that matches every node of maximum degree.*

4. An induced subgraph of a graph is formed from a subset of the nodes of the graph and all of the edges whose endpoints are both in this subset.

Now, we are ready to prove Proposition 1.

Proof of Proposition 1. First, note that in any time-slot, the network together with the present packets can be represented as a loopless multigraph. Recall that $G(k)$ denotes the multigraph at the beginning of time-slot k and $M(k)$ denotes the matching found by the NSB algorithm in time-slot k. Also, recall that 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 equal to $\Delta(k)$. Now, consider any frame k' consisting of three consecutive time-slots $\{p, p+1,\ p+2\}$, where $p = 3k'$. Suppose that the maximum node queue length is no smaller than two at the beginning of frame k', i.e., $\Delta(p) \geq 2$ at the beginning of time-slot p. Then, we want to show that under the NSB algorithm, 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 $p+1$), then the maximum degree must decrease by one in time-slot $p+2$, and becomes $\Delta(p) - 2$ at the end of time-slot $p+2$.

We start with step 1). It is a trivial case if the maximum degree decreases by one in time-slot p. Therefore, suppose the maximum degree does not decrease in time-slot p. Then, it suffices to show that all the nodes having maximum degree $\Delta(p)$ in $G(p+1)$ 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 having maximum degree must form an independent set over $G(p+1)$ at the beginning of time-slot $p+1$. We prove this by contradiction. Note that if there is only one node having maximum degree at the beginning of time-slot $p+1$, then it is trivial that the subgraph induced by this single node must consist of this node itself only and thus forms an independent set. So we consider the case where there are at least two nodes having maximum degree at the beginning of time-slot $p+1$. Suppose node i and node j are two adjacent nodes having maximum degree $\Delta(p)$ at the beginning of time-slot $p+1$. Then, none of the edges incident to either i or j was in matching $M(p)$. This implies that the edge between i and j can 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 having maximum degree must form an independent set at the beginning of time-slot $p+1$. Clearly, the subgraph induced by all the nodes having maximum degree forms an independent set and thus has no edges. In this case, it is trivial that this induced subgraph is bipartite. Then, by Lemma 6, there exists a matching over $G(p+1)$ that matches all the nodes having maximum degree in time-slot $p+1$. Note that $M(p+1)$ is an MVM over $G(p+1)$ with the assigned node weights (as in Eq. (8)) under the NSB algorithm. It is also easy to see that all the nodes with maximum degree $\Delta(p)$ are among the ones with the heaviest weight, as they have a weight of $2\Delta(p)$ and the weight of all the other nodes is less than $2\Delta(p)$. Hence, it implies from Lemma 1 that matching $M(p+1)$ also matches all the nodes having maximum degree, i.e., the maximum degree decreases by one in time-slot $p+1$. This completes the proof of step 1).

Now, we prove step 2). Clearly, the maximum degree becomes $\Delta(p) - 1$ at the beginning of time-slot $p+2$. Recall that $\mathcal{C}(p+2)$ denotes the set of critical nodes. We want to show that all the nodes in $\mathcal{C}(p+2)$ will be matched in time-slot $p+2$. We first show that all the nodes in $\mathcal{C}(p+2)$ are among the ones with the heaviest weights at the beginning of time-slot $p+2$. This is true due to the following. It is easy to see that for any node $i \in \mathcal{C}(p+2)$, it was matched at most once in time-slots p and $p+1$. Hence, according to the weight assignments in Eq. (8), node i has a weight of $2(\Delta(p) - 1)$, while all the nodes in $V \backslash \mathcal{C}(p+2)$ must have a degree less than $\Delta(p) - 1$ and thus have a weight less than $2(\Delta(p) - 1)$. Therefore, all the nodes in $\mathcal{C}(p+2)$ are among the ones with the heaviest weights.

Let $G_{\mathcal{C}(p+2)}$ denote the subgraph of $G(p+2)$ induced by all the nodes in $\mathcal{C}(p+2)$. If $G_{\mathcal{C}(p+2)}$ is bipartite, then again by Lemmas 1 and 6, following the same argument as in step 1), we can show that matching $M(p+2)$ matches all the nodes in $G_{\mathcal{C}(p+2)}$ in time-slot $p+2$. Therefore, it remains to show that $G_{\mathcal{C}(p+2)}$ is bipartite. We prove this by contradiction. Suppose $G_{\mathcal{C}(p+2)}$ 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 $\mathcal{C}(p+2)$ (i.e., their degree is $\Delta(p) - 1$ in time-slot $p+2$), then they both have maximum degree $\Delta(p)$ at the beginning of time-slot $p+1$. However, given that i and j are adjacent in $G(p+2)$ (and are thus adjacent in $G(p+1)$ as well), this contradicts what we have shown earlier – the nodes of degree $\Delta(p)$ in $G(p+1)$ 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 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 $G_{\mathcal{C}(p+2)}$ must be bipartite. This completes the proof of step 2), as well as the proof of Proposition 1. □

7.2 Proof of Lemma 3

Proof. Recall that $\mathcal{C}$ is the set of critical nodes in the fluid limits at scaled time t (Eq. (18)). We want to show that under the NSB algorithm, all the nodes in $\mathcal{C}$ will be scheduled at least twice within each frame of interval T.

First, recall that j is large enough such that $|Q_i(x_{r_j}\tau)/x_{r_j} - q_i(\tau)| < \beta/2$ 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*) $Q_i(k) \in x_{r_j}(q_{\max}(t) - \beta, q_{\max}(t) + \beta)$ for all $i \in \mathcal{C}$;

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

On account of condition (C1*) and Eq. (19), all the nodes in $\mathcal{C}$ are heavy nodes in all the time-slots of T, i.e., $Q_i(k) \geq (n-1)/n \cdot \Delta(k)$ 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 mapped to 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 any frame k' of interval T consisting of three consecutive time-slots $\{p, p+1, p+2\}$, where $p = 3k'$. 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 $\mathcal{C}'$ forms an independent set and thus has no edges. In this case, it is trivial that this induced subgraph is bipartite. Note that conditions (C1*) and (C2*) still hold even without accounting for the new packet arrivals. Then, by Lemma 4, 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 $2x_{r_j}(q_{\max}(t) - \beta)$, while any node in $V \backslash \mathcal{C}'$ cannot have a weight larger than $\max\{2x_{r_j}(q_{\max}(t) - 2\beta), x_{r_j}(q_{\max}(t) + \beta)\}$.

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 $2x_{r_j}(q_{\max}(t) - \beta)$, while any node in $V \backslash \mathcal{C}''$ cannot have a weight larger than $\max\{2x_{r_j}(q_{\max}(t) - 2\beta), x_{r_j}(q_{\max}(t) + \beta)\}$. 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 4 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 3. □

ACKNOWLEDGMENTS

This work was supported by the NSF under Grant CNS-1651947. A preliminary version of this work was presented at the IEEE INFOCOM, San Francisco, CA, April 10 – 15, 2016 [1].

REFERENCES

[1] B. Ji, G. R. Gupta, and Y. Sang, "Node-based service-balanced scheduling for provably guaranteed throughput and evacuation time performance," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

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

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

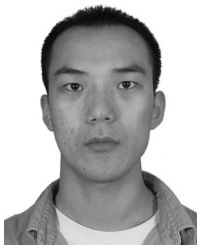
[4] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 910–917, Sep. 1988.

[5] S. Sarkar and L. Tassiulas, "End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks," *IEEE Trans. Autom. Control*, vol. 50, no. 9, pp. 1246–1259, Sep. 2005.

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

[7] C. Joo, X. Lin, and N. Shroff, "Greedy maximal matching: Performance limits for arbitrary network graphs under the node-exclusive interference model," *IEEE Trans. Autom. Control*, vol. 54, no. 12, pp. 2734–2744, Dec. 2009.

[8] G. R. Gupta, "Delay efficient control policies for wireless networks," PhD thesis, Purdue University, West Lafayette, IN, USA, 2009.
[9] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in wirelessHART networks," in *Proc. Eur. Control Conf.*, 2009, pp. 4320–4325.
[10] R. Singh and P. R. Kumar, "Throughput optimal decentralized scheduling of multi-hop networks with end-to-end deadline constraints: Unreliable links," arXiv:1606.01608, 2016, https://arxiv.org/abs/1606.01608
[11] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[12] I. Holyer, "The NP-completeness of edge-coloring," *SIAM J. Comput.*, vol. 10, no. 4, pp. 718–720, 1981.
[13] M. Stiebitz, D. Scheide, B. Toft, and L. M. Favrholdt, *Graph Edge Coloring: Vizing's Theorem and Goldberg's Conjecture*. Hoboken, Germany: Wiley, 2012.
[14] X. Wu, R. Srikant, and J. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 595–605, Jun. 2007.
[15] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proc. IEEE INFOCOM*, 1998, pp. 792–799.
[16] V. Tabatabaee and L. Tassiulas, "MNCM: A critical node matching approach to scheduling for input buffered switches with no speedup," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 294–304, Feb. 2009.
[17] B. Ji and J. Wu, "Node-based scheduling with provable evacuation time," in *Proc. 49th Annu. Conf. Inf. Sci. Syst.*, Mar. 2015, pp. 1–6.
[18] J. Kahn, "Asymptotics of the chromatic index for multigraphs," *J. Combinatorial Theory, Series B*, vol. 68, no. 2, pp. 233–254, 1996.
[19] L. Georgiadis, G. S. Paschos, L. Libman, and L. Tassiulas, "Minimal evacuation times and stability," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 931–945, Jun. 2015.
[20] L. Tassiulas and J. Joung, "Performance measures and scheduling policies in ring networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 5, pp. 576–584, Oct. 1995.
[21] L. Tassiulas, "Cut-through switching, pipelining, and scheduling for network evacuation," *IEEE/ACM Trans. Netw.*, vol. 7, no. 1, pp. 88–97, Feb. 1999.
[22] C. Joo and N. B. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1481–1493, Oct. 2009.
[23] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 709–720, Jun. 2011.
[24] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 825–836, Jun. 2012.
[25] B. Li, A. Eryilmaz, and R. Srikant, "Emulating round-robin in wireless networks," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2017, pp. 21:1–21:10.
[26] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying service rates," *Probability Eng. Inf. Sci.*, vol. 18, no. 02, pp. 191–217, 2004.
[27] T. H. Spencer and E. W. Mayr, "Node weighted matching," in *Automata, Languages and Programming*. Berlin, Germany: Springer, 1984, pp. 454–464.
[28] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," in *Proc. Annals Appl. Probability*, 1995, pp. 49–77.
[29] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, 2000, pp. 556–564.
[30] R. P. Anstee, "Simplified existence theorems for (g, f)-factors," *Discrete Appl. Math.*, vol. 27, no. 1, pp. 29–38, 1990.
[31] C.-C. Huang and T. Kavitha, "Efficient algorithms for maximum weight matchings in general graphs with small edge weights," in *Proc. 23rd Annu. ACM-SIAM Symp. Discrete Algorithms*, 2012, pp. 1400–1412.
[32] S. Pettie, "A simple reduction from maximum weight matching to maximum cardinality matching," *Inf. Process. Lett.*, vol. 112, no. 23, pp. 893–898, 2012.
[33] A. Feldmann, N. Kammenhuber, O. Maennel, B. Maggs, R. De Prisco, and R. Sundaram, "A methodology for estimating interdomain web traffic demand," in *Proc. 4th ACM SIGCOMM Conf. Internet Meas.*, 2004, pp. 322–335.
[34] DIMACS Graphs. [Online]. Available: http://www.info.univ-angers.fr/pub/porumbel/graphs/
[35] R. P. Anstee and J. R. Griggs, "An application of matching theory of edge-colourings," *Discrete Math.*, vol. 156, no. 1, pp. 253–256, 1996.
[36] S. I. Resnick, *A Probability Path*. Berlin, Germany: Springer, 2013.
[37] D. König, "Über graphen und ihre anwendung auf determinantentheorie und mengenlehre," *Mathematische Annalen*, vol. 77, no. 4, pp. 453–465, 1916.
[38] A. Kapoor and R. Rizzi, "Edge-coloring bipartite graphs," *J. Algorithms*, vol. 34, no. 2, pp. 390–396, 2000.



Yu Sang received the BE degree in electronic information engineering from the University of Science and Technology of China, in 2014. He is currently working toward the PhD degree in the CIS Department at Temple University. His research interests are in wireless networks and cloud computing systems.

Gagan R. Gupta received the bachelor's of technology degree in computer science and engineering from the Indian Institute of Technology, Delhi, India, in 2005. He received the MS degree in computer science from the University of Wisconsin, Madison, in 2006. He received the PhD degree from Purdue University, West Lafayette, Indiana, in 2009 for his dissertation titled "Delay efficient control policies for wireless networks." His research interests include performance modeling and optimization of communication networks and parallel computing. He is currently working in the telecommunications industry.

Bo Ji (S'11-M'12) received the BE and ME degrees in information science and electronic engineering from Zhejiang University, China, in 2004 and 2006, respectively. He received the PhD degree in electrical and computer engineering from The Ohio State University, in 2012. He is currently an assistant professor of the CIS Department at Temple University, Philadelphia, Pennsylvania, and is also a faculty member of the Center for Networked Computing (CNC). Prior to joining Temple University, he was a senior member of technical staff with AT&T Labs, San Ramon, California, from January 2013 to June 2014. His research interests include the modeling, analysis, control, and optimization of complex networked systems, such as communication networks, information-update systems, cloud/data-center networks, and cyber-physical systems. He received an US National Science Foundation (NSF) CAREER Award and NSF CISE Research Initiation Initiative (CRII) Award both in 2017. He is a member of the IEEE.