Throughput-Optimal Scheduling in Multihop Wireless Networks Without Per-Flow Information

Bo Ji, *Student Member, IEEE*, Changhee Joo, *Member, IEEE*, and Ness B. Shroff, *Fellow, IEEE*

***Abstract*—In this paper, we consider the problem of link scheduling in multihop wireless networks under general interference constraints. Our goal is to design scheduling schemes that do not use per-flow or per-destination information, maintain a single data queue for each link, and exploit only local information, while guaranteeing throughput optimality. Although the celebrated back-pressure algorithm maximizes throughput, it requires per-flow or per-destination information. It is usually difficult to obtain and maintain this type of information, especially in large networks, where there are numerous flows. Also, the back-pressure algorithm maintains a complex data structure at each node, keeps exchanging queue-length information among neighboring nodes, and commonly results in poor delay performance. In this paper, we propose scheduling schemes that can circumvent these drawbacks and guarantee throughput optimality. These schemes use either the readily available hop-count information or only the local information for each link. We rigorously analyze the performance of the proposed schemes using fluid limit techniques via an inductive argument and show that they are throughput-optimal. We also conduct simulations to validate our theoretical results in various settings and show that the proposed schemes can substantially improve the delay performance in most scenarios.**

***Index Terms*—Fluid limit tecniques, multihop wireless networks, per-hop/per-link queues, scheduling, throughput-optimal, without per-flow information.**

I. Introduction

LINK scheduling is a critical resource allocation functionality in multihop wireless networks, and also perhaps the most challenging. The seminal work of [1] introduces a joint adaptive routing and scheduling algorithm, called back-pressure, that has been shown to be throughput-optimal, i.e., it can stabilize the network under any feasible load. This paper focuses on the settings with fixed routes, where the back-pressure algorithm becomes a scheduling algorithm consisting of two components: flow scheduling and link scheduling. The back-pressure algorithm calculates the weight of a link as the product of the link capacity and the maximum "back-pressure" (i.e., the queue length difference between the queues at the transmitting nodes of this link and the next hop link for each flow) among all the flows passing through the link, and solves a MaxWeight problem to activate a set of noninterfering links that have the largest weight sum. The flow with the maximum queue length difference at a link is chosen to transmit packets when the link is activated.

The back-pressure algorithm, although throughput-optimal, needs to solve a MaxWeight problem, which requires centralized operations and is NP-hard in general [2]. To this end, simple scheduling algorithms based on carrier sensing multiple access (CSMA) [3]–[5] are developed to achieve the optimal throughput in a distributed manner for single-hop traffic and are later extended to the case of multihop traffic [3] leveraging the basic idea of back-pressure.

However, the back-pressure-type of scheduling algorithms (including CSMA for multihop traffic) have the following shortcomings: 1) require per-flow or per-destination information, which is usually difficult to obtain and maintain, especially in large networks where there are numerous flows; 2) need to maintain separate queues for each flow or destination at each node; 3) rely on extensive exchange of queue length information among neighboring nodes to calculate link weights, which becomes the major obstacle to their distributed implementation; and 4) may result in poor overall delay performance, as the queue length needs to build up (creating the back-pressure) from a flow destination to its source, which leads to large queues along the route a flow takes [6], [7]. An important question is whether one can circumvent the above drawbacks of the back-pressure-type of algorithms and design throughput-optimal scheduling algorithms that do not require per-flow or per-destination information, maintain a small number of data queues (ideally, a single data queue for each link), exploit only local information when making scheduling decisions, and potentially have good delay performance.

There have been some recent studies (e.g., [6] and [8]–[10]) in this direction. A cluster-based back-pressure algorithm that can reduce the number of queues is proposed in [9], where nodes (or routers) are grouped into clusters and each node needs only to maintain separate queues for destinations within its cluster. In [6], the authors propose a back-pressure policy making scheduling decisions in a shadow layer (where counters are used as per-flow shadow queues). Their scheme only needs to maintain a single *first-in–first-out* (FIFO) queue instead of per-flow queues for each link and shows dramatic improvement in the delay performance. However, their shadow algorithm still requires per-flow information and constant exchange of shadow queue length information among neighboring nodes. The work

Manuscript received July 05, 2011; revised March 17, 2012 and May 31, 2012; accepted June 07, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Bonald. This work was supported in part by the ARO under MURI Award W911NF-08-1-0238; the NSF under Awards 1012700-CNS, 0721236-CNS, 0721434-CNS, and 1065136-CNS; and the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science, and Technology under Grant No. 2012-0003227. A preliminary version of this work was presented at the IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Princeton, NJ, May 9–13, 2011.

B. Ji is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: ji@ece.osu.edu).

C. Joo is with the School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan 689-798, Korea (e-mail: cjoo@unist.ac.kr).

N. B. Shroff is with the Departments of Electrical and Computer Engineering and Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: shroff@ece.osu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2205017

in [8] proposes to exploit the local queue-length information to design throughput-optimal scheduling algorithms. Their approach combined with CSMA algorithms can achieve fully distributed scheduling without any information exchange. Their scheme is based on a two-stage queue structure, where each node maintains two types of data queues: per-flow queues and per-link queues. The two-stage queue structure imposes additional complexity and is similar to queues with regulators [11], which have been empirically noted to have very large delays. In [10], the authors propose a back-pressure algorithm that integrates the shortest path routing to minimize the average number of hops between each source and destination pair. However, their scheme further increases the number of queues by maintaining a separate queue $\{i, d, k\}$ at each node i for the packets that will be delivered to destination node d within k hops.

Although these algorithms partly alleviate the effect of the aforementioned disadvantages of the traditional back-pressure algorithms, to the best of our knowledge, no work has addressed all the aforementioned four issues. In particular, a critical drawback of the earlier mentioned works is that they require *per-flow or per-destination information* to guarantee throughput optimality. In this paper, we propose a class of throughput-optimal schemes that can remove this per-flow or per-destination information requirement, maintain a single data queue for each link, and remove information exchange. As a by-product, these proposed schemes also improve the delay performance in a variety of scenarios.

The main contributions of our paper are as follows.

First, we propose a scheduling scheme with *per-hop* queues to address the four key issues mentioned earlier. The proposed scheme maintains multiple FIFO queues $Q_{l,k}$ at the transmitting node of each link l. Specifically, any packet whose transmission over link l is the kth hop forwarding from its source node is stored at queue $Q_{l,k}$. This hop-count information is much easier to obtain and maintain compared to per-flow or per-destination information. For example, hop-count information can be obtained using *time-to-live* (TTL) information in packet headers. Moreover, as mentioned earlier, while the number of flows in a large network is very large, the number of hops is often much smaller. In the Internet, the longest route a flow takes typically has tens of hops,[1] while there are billions of users or nodes [14], and thus the number of flows could be extremely large. A shadow algorithm similar to [6] is adopted in our framework, where a shadow queue is associated with each data queue. We consider the MaxWeight algorithm based on shadow queue lengths and show that this per-Hop-Queue-based MaxWeight Scheduler (HQ-MWS) is throughput-optimal using fluid limit techniques via a hop-by-hop inductive argument. For illustration, in this paper, we focus on the centralized MaxWeight-type of policies. However, one can readily extend our approach to a large class of scheduling policies (where fluid limit techniques can be used). For example, combining our approach with the CSMA-based algorithms of [3]–[5], one can completely remove the requirement of queue-length information exchange and develop fully distributed scheduling schemes, under which no information exchange is required. *To the best of our knowledge, this is the first work that develops throughput-optimal scheduling schemes without per-flow or per-destination information in wireless networks with multihop traffic.* In addition, we believe that using this type of per-hop queue structure to study the problem of link scheduling is of independent interest.

[1]In the Routing Information Protocol (RIP) [12], the longest route is limited to 15 hops. In general, an upper bound on the length of a route is 255 hops in the Internet, as specified by TTL in the Internet Protocol (IP) [13].

Second, we have also developed schemes with per-link queues (i.e., a single data queue per link) instead of per-hop queues, extending the idea to per-Link-Queue-based MaxWeight Scheduler (LQ-MWS). We propose two schemes based on LQ-MWS using different queueing disciplines. We first combine it with the *priority* queueing discipline (called PLQ-MWS), where a higher priority is given to the packet that traverses a smaller number of hops, and then prove throughput optimality of PLQ-MWS. It is of independent interest that this type of hop-count-based priority discipline enforces stability. This, however, requires that nodes sort packets according to their hop-count information. We then remove this restriction by combining LQ-MWS with the FIFO queueing discipline (called FLQ-MWS) and prove throughput optimality of FLQ-MWS in networks where flows do not form loops.

Finally, we show through simulations that the proposed schemes can significantly improve the delay performance in most scenarios. In addition, the schemes with per-link queues (PLQ-MWS and FLQ-MWS) perform well in a wider variety of scenarios, which implies that maintaining per-link queues not only simplifies the data structure, but also can contribute to scheduling efficiency and delay performance.

The remainder of the paper is organized as follows. In Section II, we present a detailed description of our system model. In Section III, we prove throughput optimality of HQ-MWS using fluid limit techniques via a hop-by-hop inductive argument. We extend our ideas to show throughput optimality of PLQ-MWS and FLQ-MWS in Section IV. Furthermore, we evaluate different scheduling schemes through simulations in Section V. Finally, we conclude our paper in Section VI.

II. System Model

We consider a multihop wireless network described by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of links. Nodes are wireless transmitters/receivers, and links are wireless channels between two nodes if they can directly communicate with each other. Let $b(l)$ and $e(l)$ denote the transmitting node and receiving node of link $l = (b(l), e(l)) \in \mathcal{E}$, respectively. Note that we distinguish links (i, j) and (j, i). We assume a time-slotted system with a single frequency channel. Let c_l denote the link capacity of link l, i.e., link l can transmit at most c_l packets during a time-slot if none of the links that interfere with l is transmitting at the same time. We assume unit capacity links, i.e., $c_l = 1$ for all $l \in \mathcal{E}$. A flow is a stream of packets from a source node to a destination node. Packets are injected at the source and traverse multiple links to the destination via multihop communications. Let $\mathcal{S}$ denote the set of flows in the network. We assume that each flow s has a single, fixed, and loop-free route that is denoted by $\mathcal{L}(s) = \left(l_1^s, \ldots, l_{|\mathcal{L}(s)|}^s\right)$, where the route of flow s has $|\mathcal{L}(s)|$ hop length from the source to the destination, l_k^s denotes the kth hop link on the route of flow s, and $|\cdot|$ denotes the cardinality of a set. Let $L^{\max} \triangleq \max_s |\mathcal{L}(s)| < \infty$ denote the length of the longest route over all flows. Let $H_{l,k}^s \in \{0, 1\}$ be 1 if link l is the kth hop link on the route of flow s, and 0 otherwise. Note that the assumption of single route and unit capacity is only for ease of exposition, and one can readily extend the results to more general scenarios with *multiple fixed routes and heterogeneous link capacities*, applying the techniques used in this paper. We also restrict our attention to those links that have flows passing through them. Hence, without loss of generality, we assume that $\sum_s \sum_{k=1}^{|\mathcal{L}(s)|} H_{l,k}^s \geq 1$, for all $l \in \mathcal{E}$.

The interference set of link l is defined as $I(l) \triangleq \{j \in \mathcal{E} \mid$ link j interferes with link $l\}$. We consider a general interference model, where the interference is symmetric, i.e., for any $l, j \in \mathcal{E}$, if $l \in I(j)$, then $j \in I(l)$. A *schedule* is a set of (active or inactive) links, and can be represented by a vector $M \in \{0,1\}^{|\mathcal{E}|}$, where component M_l is set to 1 if link l is active, and 0 if it is inactive. A schedule M is said to be *feasible* if no two links of M interfere with each other, i.e., $l \notin I(j)$ for all l, j with $M_l = 1$ and $M_j = 1$. Let $\mathcal{M}$ denote the set of all feasible schedules over $\mathcal{E}$, and let $\mathrm{Co}(\mathcal{M})$ denote its convex hull.

Let $F_s(t)$ denote the cumulative number of packet arrivals at the source node of flow s up to time-slot t. We assume that packets are of unit length. We assume that each arrival process $F_s(t) - F_s(t-1)$ is an irreducible positive recurrent Markov chain with countable state space and satisfies the Strong Law of Large Numbers (SLLN): That is, with probability one

$$\lim_{t\to\infty} \frac{F_s(t)}{t} = \lambda_s \tag{1}$$

for each flow $s \in \mathcal{S}$, where λ_s is the mean arrival rate of flow s. We let $\lambda \triangleq [\lambda_s]$ denote the arrival rate vector. Also, we assume that the arrival processes are mutually independent across flows. (This assumption can be relaxed as in [15].)

As in [15], a stochastic queueing network is said to be *stable* if it can be described as a discrete-time countable Markov chain and the Markov chain is *stable* in the following sense: The set of positive recurrent states is nonempty, and it contains a finite subset such that, with probability one, this subset is reached within finite time from any initial state. When all the states communicate, stability is equivalent to the Markov chain being positive recurrent [16]. We define the *throughput region* of a scheduling policy as the set of arrival rate vectors for which the network is stable under this policy. Furthermore, we define the *optimal throughput region* (or *stability region*) as the union of the throughput regions of all possible scheduling policies, including the offline policies [1]. We denote by Λ^* the optimal throughput region, whereby Λ^* can be represented as

$$\Lambda^* \triangleq \{\lambda \mid \text{ for some } \phi \in \mathrm{Co}(\mathcal{M}), \sum_s \sum_k H_{l,k}^s \lambda_s \leq \phi_l \text{ for all links } l \in \varepsilon\}. \tag{2}$$

An arrival rate vector is strictly inside Λ^* if the inequalities above are all strict.

Throughout the paper, we let $(z)^+ \triangleq \max(z, 0)$ denote the larger value between z and 0.

III. Scheduling With Per-Hop Queues

In this section, we propose scheduling policies with per-hop queues and shadow algorithm. We will later extend our ideas to developing schemes with per-link queues in Section IV. We describe our scheduling schemes using the centralized MaxWeight algorithm for ease of presentation. Our approach combined with the CSMA algorithms can be extended to develop fully distributed scheduling algorithms. (Please refer to our online technical report [17].)

A. Queue Structure and Scheduling Algorithm

We start with the description of queue structure, and then specify our scheduling scheme based on per-hop queues and a shadow algorithm. We assume that, at the transmitting node of each link l, a single FIFO data queue $Q_{l,k}$ is maintained for packets whose kth hop is link l, where $1 \leq k \leq L^{\max}$. Such queues are called *per-hop* queues. For notational convenience, we also use $Q_{l,k}(t)$ to denote the queue length of $Q_{l,k}$ at time-slot t. Let $\Pi_{l,k}(t)$ denote the service of $Q_{l,k}$ at time-slot t, which takes a value of c_l (i.e., 1 in our setting) if queue $Q_{l,k}$ is active, or 0 otherwise. Let $D_{l,k}(t)$ denote the cumulative number of packet departures from queue $Q_{l,k}$ up to time-slot t, and let $\Psi_{l,k}(t) \triangleq D_{l,k}(t) - D_{l,k}(t-1)$ be the number of packet departures from queue $Q_{l,k}$ at time-slot t. Since a queue may be empty when it is scheduled, we have $\Psi_{l,k}(t) \leq \Pi_{l,k}(t)$ for all time-slots $t \geq 0$. Let $U_{s,k}(t)$ denote the cumulative number of packets transmitted from the $(k-1)$st hop to the kth hop for flow s up to time-slot t for $1 \leq k \leq \mathcal{L}(s)$, where we set $U_{s,1}(t) = F_s(t)$. Let $A_{l,k}(t)$ be the cumulative number of aggregate packet arrivals (including both exogenous arrivals and arrivals from the previous hops) at queue $Q_{l,k}$ up to time-slot t. Then, we have $A_{l,k}(t) = \sum_s H_{l,k}^s U_{s,k}(t)$, and in particular, $A_{l,1}(t) = \sum_s H_{l,1}^s F_s(t)$. Let $P_{l,k}(t) \triangleq A_{l,k}(t) - A_{l,k}(t-1)$ denote the number of arrivals for queue $Q_{l,k}$ at time-slot t. We adopt the convention that $A_{l,k}(0) = 0$ and $D_{l,k}(0) = 0$ for all $l \in \mathcal{E}$ and $1 \leq k \leq L^{\max}$. The queue length evolves as

$$Q_{l,k}(t) = Q_{l,k}(0) + A_{l,k}(t) - D_{l,k}(t). \tag{3}$$

For each data queue $Q_{l,k}$, we maintain a shadow queue $\hat{Q}_{l,k}$, and let $\hat{Q}_{l,k}(t)$ denote its queue length at time-slot t. The arrival and departure processes of the shadow queues are controlled as follows. We denote by $\hat{A}_{l,k}(t)$ and $\hat{D}_{l,k}(t)$ its cumulative amount of arrivals and departures up to time-slot t, respectively. Also, let $\hat{\Pi}_{l,k}(t)$, $\hat{P}_{l,k}(t) \triangleq \hat{A}_{l,k}(t) - \hat{A}_{l,k}(t-1)$ and $\hat{\Psi}_{l,k}(t) \triangleq \hat{D}_{l,k}(t) - \hat{D}_{l,k}(t-1)$ denote the amount of service, arrivals, and departures of queue $\hat{Q}_{l,k}$ at time-slot t, respectively. Likewise, we have $\hat{\Psi}_{l,k}(t) \leq \hat{\Pi}_{l,k}(t)$ for $t \geq 0$. We set by convention that $\hat{A}_{l,k}(0) = 0$ and $\hat{D}_{l,k}(0) = 0$ for all queues $\hat{Q}_{l,k}$. The arrivals for shadow queue $\hat{Q}_{l,k}$ are set to $(1+\epsilon)$ times the average amount of packet arrivals at data queue $Q_{l,k}$ up to time-slot t, i.e.,

$$\hat{P}_{l,k}(t) = (1+\epsilon)\frac{A_{l,k}(t)}{t} \tag{4}$$

where $\epsilon > 0$ is a sufficiently small positive number such that $(1+\epsilon)\lambda$ is also strictly inside Λ^* given that λ is strictly inside Λ^*. Then, the shadow queue length evolves as

$$\hat{Q}_{l,k}(t) = \hat{Q}_{l,k}(0) + \hat{A}_{l,k}(t) - \hat{D}_{l,k}(t). \tag{5}$$

Using these shadow queues, we determine the service of both data queues and shadow queues using the following MaxWeight algorithm.

1) Per-Hop-Queue-Based MaxWeight Scheduler (HQ-MWS): At each time-slot t, the scheduler serves data queues $Q_{l,k^*(l)}$ for $l \in M^*$, where

$$k^*(l) \in \mathrm{argmax}_k \hat{Q}_{l,k}(t), \text{ for each link } l \in \mathcal{E} \tag{6}$$

$$M^* \in \operatorname{argmax}_{M \in \mathcal{M}} \sum_{l \in \mathcal{E}} \hat{Q}_{l,k^*(l)}(t) \cdot M_l. \tag{7}$$

In other words, we set the service of data queue as $\Pi_{l,k}(t) = 1$ if $l \in M^*$ and $k = k^*(l)$, and $\Pi_{l,k}(t) = 0$ otherwise. We also set the service of shadow queues as $\hat{\Pi}_{l,k}(t) = \Pi_{l,k}(t)$ for all l and k.

Remark: The algorithm needs to solve a MaxWeight problem based on the shadow queue lengths, and ties can be broken arbitrarily if there is more than one queue having the largest shadow queue length at a link or there is more than one schedule having the largest weight sum. Note that we have $\Pi_{l,k}(t) = \hat{\Pi}_{l,k}(t)$ under this scheduling scheme, for all links $l \in \mathcal{E}$ and $1 \leq k \leq L^{\max}$ and for all time-slots $t \geq 0$. Once a schedule M^* is selected, data queues $Q_{l,k^*(l)}$ for links l with $M^*_l = 1$ are activated to transmit packets if they are nonempty, and shadow queues $\hat{Q}_{l,k^*(l)}$ "transmit" shadow packets as well. Note that shadow queues are just counters. The arrival and departure process of a shadow queue is simply an operation of addition and subtraction, respectively.

B. Throughput Optimality

We present the main result of this section as follows.

Proposition 1: HQ-MWS is throughput-optimal, i.e., the network is stable under HQ-MWS for any arrival rate vector λ strictly inside Λ^*.

We prove the stability of the network in the sense that the underlying Markov chain (whose state accounts for both data queues and shadow queues; see Appendix A for the detailed state description) is stable under HQ-MWS, using fluid limit techniques [15], [18]. We provide the proof of Proposition 1 in Appendix A and discuss the outline of the proof as follows.

Note that the shadow queues serve only single-hop traffic, i.e., after packets in the shadow queues are served, they leave the system without being transmitted to another shadow queue. We also emphasize that the single-hop shadow traffic gets smoothed under the arrival process of (4), and in the fluid limits (which will be formally established in Appendix A), after a finite time, the instantaneous shadow arrival rate is strictly inside the optimal throughput region Λ^* with small enough $\epsilon > 0$. Then, using the standard Lyapunov approach, we can show the stability for the subsystem consisting of shadow queues.

Now, we consider the data queues in the fluid limits starting from the first-hop data queue for each link $l \in \mathcal{E}$. Since the arrival process of data queue $Q_{l,1}$ satisfies the SLLN, the instantaneous arrival of shadow queue $\hat{Q}_{l,1}$ will be equal to $(1+\epsilon)\sum_s H^s_{l,1}\lambda_s$. This implies that the service rate of shadow queue $\hat{Q}_{l,1}$ is no smaller than $(1+\epsilon)\sum_s H^s_{l,1}\lambda_s$ due to the stability of shadow queues. Then, the service rate of data queue $Q_{l,1}$ is also no smaller than $(1+\epsilon)\sum_s H^s_{l,1}\lambda_s$ because $\Pi_{l,k}(t) = \hat{\Pi}_{l,k}(t)$ under HQ-MWS. Since the arrival rate of data queue $Q_{l,1}$ is $\sum_s H^s_{l,1}\lambda_s$, the service rate is strictly greater than the arrival rate for $Q_{l,1}$, establishing its stability. Using this as an induction base, we can show the stability of data queues via a hop-by-hop inductive argument. This immediately implies that the fluid limit model of the joint system is stable under HQ-MWS.

Although our proposed scheme shares similarities with [6] and [8], it has important differences. First, in [6], per-flow information is still required by their shadow algorithm. The shadow packets are injected into the network at the sources and are then "transmitted" to the destinations via multihop communications. Their scheme strongly relies on the information exchange of shadow queue lengths to calculate the link weights. In contrast, we take a different approach for constructing the instantaneous arrivals at each shadow queue according to (4) that is based on the average amount of packet arrivals at the corresponding data queue. This method of injecting shadow packets allows us to decompose multihop traffic into single-hop traffic for shadow queues and exploit only local information when making scheduling decisions. Second, although the basic idea behind the shadow arrival process of (4) is similar to the service process of the per-flow queues in [8], the scheme in [8] requires per-flow information and relies on a two-stage queue architecture that consists of both per-flow and per-link data queues. In contrast, our scheme needs only per-hop (and not per-flow) information, i.e., the number of hops each packet has traversed, completely removing per-flow information and per-flow queues. This simplification of required information and data structure is critical due to the fact that the maximum number of hops in a network is usually much smaller than the number of flows in a large network. For example, in the Internet, the longest route a flow takes typically has tens of hops, while there are billions of nodes, and thus the number of flows could be extremely large.

Note that the hop-count in our approach is counted from the source. Such per-hop information is easy to obtain (e.g., from *time-to-live* or *TTL* information in the Internet and ad hoc networks). At each link, packets with the same hop-count (from the source of each packet to the link) are kept at the same queue, regardless of sources, destinations, and flows, which significantly reduces the number of queues. In Section IV, we extend our approach to the schemes with per-link queues and further remove even the requirement of per-hop information.

IV. Scheduling With Per-Link Queues

In the previous section, we show that per-hop-queue-based MaxWeight scheduler (HQ-MWS) achieves optimal throughput performance. In this section, we extend our ideas to developing schemes with *per-link* queues. To elaborate, we show that per-link-queue-based MaxWeight scheduler, when associated with *priority* or *FIFO* queueing discipline, can also achieve throughput optimality.

A. MaxWeight Algorithm With Per-Link Queues

We consider a network where each link l has a single data queue Q_l. Let $Q_l(t)$, $A_l(t)$, $D_l(t)$, $\Pi_l(t)$, $\Psi_l(t)$ and $P_l(t)$ denote the queue length, cumulative arrival, cumulative departure, service, departure and arrival at the data queue Q_l, respectively. Also, we maintain a shadow queue $\hat{Q}_l$ associated with each Q_l, and let $\hat{Q}_l(t)$, $\hat{A}_l(t)$, $\hat{D}_l(t)$, $\hat{\Pi}_l(t)$, $\hat{\Psi}_l(t)$ and $\hat{P}_l(t)$ denote the queue length, cumulative arrival, accumulative departure, service, departure and arrival at the shadow queue $\hat{Q}_l$, respectively. Similar to (4) for per-hop shadow queues, we control the arrivals to the shadow queue $\hat{Q}_l$ as

$$\hat{P}_l(t) = (1+\epsilon)\frac{A_l(t)}{t} \tag{8}$$

where $\epsilon > 0$ is a sufficiently small positive number.

Next, we specify the MaxWeight algorithm with per-link queues as follows.

1) Per-Link-Queue-Based MaxWeight Scheduler (LQ-MWS): At each time-slot t, the scheduler serves links in M^* (i.e., $\Pi_l(t) = 1$ for $l \in M^*$, and $\Pi_l(t) = 0$ otherwise), where

$$M^* \in \text{argmax}_{M \in \mathcal{M}} \sum_{l \in \mathcal{E}} \hat{Q}_l(t) \cdot M_l.$$

Also, we set the service of shadow queues as $\hat{\Pi}_l(t) = \Pi_l(t)$ for all l.

Similar as in HQ-MWS, the shadow traffic under LQ-MWS gets smoothed due to the shadow arrival assignment of (8), and the instantaneous arrival rate of shadow queues can be shown to be strictly inside the optimal throughput region Λ^*. Hence, we show in Lemma 16 (see Appendix B) that the fluid limit model for the subsystem consisting of shadow queues is stable under LQ-MWS, using the standard Lyapunov approach and following the same line of analysis for HQ-MWS.

B. LQ-MWS With Priority Discipline

We develop a scheduling scheme by combining LQ-MWS with priority queueing discipline, called *PLQ-MWS*. Regarding priority of packets at each per-link queue, we define *hop-class* as follows: A packet has hop-class k if the link where the packet is located is the kth hop from the source of the packet. When a link is activated to transmit packets, packets with a smaller hop-class will be transmitted earlier; and packets with the same hop-class will be transmitted in a FIFO fashion.

Proposition 2: PLQ-MWS is throughput-optimal.

We provide the outline of the proof and refer to our online technical report [17] for the detailed proof. Basically, we follow the line of analysis for HQ-MWS using fluid limit techniques and induction method. Since a link transmits packets according to their priorities (i.e., hop-classes or hop-count from their respective source nodes), we can view packets with hop-class k at link l as in a subqueue $Q_{l,k}$ (similar to the per-hop queues under HQ-MWS). Now, we consider the data queues in the fluid limits. Since the exogenous arrival process satisfies the SLLN, the instantaneous arrival to shadow queue $\hat{Q}_l$ will be at least $(1+\epsilon)\sum_s H^s_{l,1}\lambda_s$ for each link $l \in \mathcal{E}$. This implies that the service rate of shadow queue $\hat{Q}_l$ is no smaller than $(1+\epsilon)\sum_s H^s_{l,1}\lambda_s$ due to the stability of the shadow queues (see Lemma 16 in Appendix B). Then, the service rate of subqueue $Q_{l,1}$ is also no smaller than $(1+\epsilon)\sum_s H^s_{l,1}\lambda_s$ because: 1) $\Pi_l(t) = \hat{\Pi}_l(t)$ under PLQ-MWS; and 2) the highest priority is given to subqueue $Q_{l,1}$ when link l is activated to transmit. Since the arrival rate of subqueue $Q_{l,1}$ is $\sum_s H^s_{l,1}\lambda_s$, the service rate is strictly greater than the arrival rate for subqueue $Q_{l,1}$, establishing its stability. Similarly, we can show that the hop-class-j subqueues are stable for all $j \le k+1$, given the stability of the hop-class-j' subqueues for all $j' \le k$. Therefore, we can show the stability of the data queues via a hop-by-hop inductive argument. This immediately implies that the fluid limit model of the joint system is stable under PLQ-MWS.

We emphasize that a "bad" priority discipline may cause instability (even in wireline networks). See [19] and [20] for two simple counterexamples showing that in a two-station network, a static priority discipline that gives a higher priority to customers with a larger hop-count may result in instability. (Interested readers are also referred to [16, Ch. 3] for a good summary of the instability results.) The key intuition of these counterexamples is that by giving a higher priority to packets with a larger hop-count in one station, the priority discipline may impede forwarding packets with a smaller hop-count to the next-hop station, which in turn starves the next-hop station. On the other hand, PLQ-MWS successfully eliminates this type of inefficiency by giving a higher priority to the packets with a smaller hop-count and continues to push the packets to the following hops.

Note that PLQ-MWS is different from HQ-MWS, although they appear to be similar. HQ-MWS makes scheduling decisions based on the queue length of each per-hop shadow queue. This may result in a waste of service if a per-hop queue is activated but does not have enough packets to transmit, even though the other per-hop queues of the same link have packets. In contrast, PLQ-MWS makes decisions based on the queue length of each per-link shadow queue and allows a link to transmit packets of multiple hop-classes, avoiding such an inefficiency. The performance difference due to this phenomenon will be illustrated through simulations in Section V. Furthermore, the implementation of PLQ-MWS is easier than HQ-MWS since PLQ-MWS needs to maintain only one single shadow queue per link.

Another aspect of PLQ-MWS we would like to discuss is about the *hop-count-based priority discipline* in the context of multiclass queueing networks (or wireline networks). In operations research, stability of multiclass queueing networks has been extensively studied in the literature (e.g., see [16] and the references therein). To the best of our knowledge, however, there is very limited work on the topic of "priority enforces stability" [21]–[23]. In [21] and [22], the authors obtained sufficient conditions (based on linear or piecewise linear Lyapunov functions) for the stability of a multiclass fluid network and/or queueing network under priority disciplines. However, to verify these sufficient conditions relies on verifying the feasibility of a set of inequalities, which in general can be very difficult. The most related work to ours is [23]. There, the authors showed that under the condition of "Acyclic Class Transfer," where customers can switch classes unless there is a loop in class transfers, a simple priority discipline stabilizes the network under the usual traffic condition (i.e., the normalized load is less than one). Their priority discipline gives a higher priority to customers that are closer to their respective sources.

Interestingly, our hop-count-based priority discipline (for wireline networks) is similar to the discipline proposed in [23]. However, there is a major difference in that while [23] studies stability of wireline networks (without link interferences) under the usual traffic condition, we consider stability of wireless networks with interference constraints that impose the (link) scheduling problem, which is much more challenging. In wireless networks, the service rate of each link depends on the underlying scheduling scheme, rather than being fixed as in wireline networks. Hence, the difficulty is to establish the usual traffic condition by designing appropriate wireless scheduling schemes. In this paper, we develop PLQ-MQS scheme and show that the usual traffic condition and then stability can be established via a hop-by-hop inductive argument under the PLQ-MWS scheme.

C. LQ-MWS With FIFO Discipline

In this section, we develop a scheduling scheme, called *FLQ-MWS*, by combining the LQ-MWS algorithm developed in Section IV-A with *FIFO* queueing discipline (instead of priority queueing discipline) and show that this scheme is throughput-optimal if flows do not form loops. We emphasize that FLQ-MWS requires neither per-flow information nor hop-count information.

To begin with, we define a positive integer $r(l)$ as the rank of link $l \in \mathcal{E}$ and call $R(\mathcal{E}) = (r(l), l \in \mathcal{E})$ a ranking of $\mathcal{E}$. Recall that $\mathcal{L}(s)$ denotes the loop-free route of flow s. In the following, we prove a key property of the network where flows do not form loops, which will be used to prove the main results in this section.

Lemma 3: Consider a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of flows $\mathcal{S}$, where the flows do not form loops. There exists a ranking $R(\mathcal{E})$ such that the following two statements hold.

1) For any flow $s \in \mathcal{S}$, the ranks are monotonically increasing when one traverses the links of flow s from l_1^s to $l_{|\mathcal{L}(s)|}^s$, i.e., $r\left(l_i^s\right) < r\left(l_{i+1}^s\right)$ for all $1 \leq i < |\mathcal{L}(s)|$.
2) The packet arrivals at a link are either exogenous or forwarded from links with a smaller rank.

We provide the proof of Lemma 3 in Appendix C. Note that such a ranking with the monotone property exists because the flows do not form a loop. In contrast, it is clear that if flows form a loop, then such a ranking does not exist. Two examples of the networks where flows do not form loops are provided in Fig. 4(b) and (c) in Appendix C, and an example of the network where flows do form a loop is provided in Fig. 4(a). Note that the ranking is only for the purpose of analysis and plays a key role in proving the system stability under FLQ-MWS, while it will not be used in the actual link scheduling algorithm.

Now, we give the main results of this section in the following proposition.

Proposition 4: FLQ-MWS is throughput-optimal in networks where *flows do not form loops*.

We omit the detailed proof and refer to our online technical report [17]. In the following, we provide the outline of the proof. Motivated by Lemma 3, we extend our analysis for HQ-MWS (or PLQ-MWS). Compared to the PLQ-MWS algorithm, there are differences only in the operations with data queues, and the underlying LQ-MWS algorithm remains the same. Thus, the shadow queues will exhibit similar behaviors, and the fluid limit model for the subsystem of shadow queues is stable under FLQ-MWS (see Lemma 16 in Appendix B). Also, note that Lemma 3 implies that given the qualified ranking (without loss of generality, assuming that the smallest rank is 1), the packet arrivals at links with rank 1 are all exogenous, then following a similar argument in the proof of Proposition 1, we can prove the stability of the corresponding data queues by showing that the instantaneous arrival rate is less than the instantaneous service rate. Since Lemma 3 also implies that the packet arrivals at links with rank 2 are either exogenous or from links with rank 1, we can similarly show the stability of links with rank 2. Repeating the above argument, we can prove the stability of all data queues by induction, which completes the proof of Proposition 4.

Corollary 5: FLQ-MWS is throughput-optimal in tree networks.

The above corollary follows immediately from Proposition 4 because a tree network itself does not contain a cycle of links and flows are all loop-free.

V. Numerical Results

In this section, we evaluate different scheduling schemes through simulations. We compare scheduling performance of HQ-MWS, PLQ-MWS, FLQ-MWS with the original back-pressure (BP) algorithm under the *node-exclusive*[2] interference model. Note that we focus on the node-exclusive interference model only for the purpose of illustration. Our scheduling schemes can be applied to general interference constraints as specified in Section II. We will first focus on a simple linear network topology to illustrate the advantages of the proposed schemes and further validate our theoretical results in a larger and more realistic grid network topology. The impacts of the parameter ϵ on the scheduling performance are also explored and discussed in our online technical report [17].

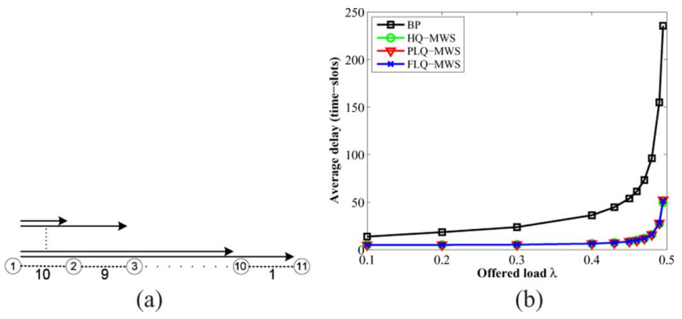
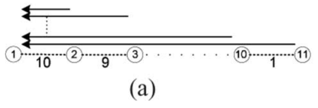


Fig. 1. Performance of BP, HQ-MWS, PLQ-MWS, and FLQ-MWS in a linear network topology ($\epsilon = 0.005$). (a) Linear network topology with 10 links. (b) Average delay.

First, we evaluate and compare the scheduling performance of HQ-MWS, PLQ-MWS, FLQ-MWS, and the back-pressure algorithm in a simple linear network that consists of 11 nodes and 10 links as shown in Fig. 1(a), where nodes are represented by circles and links are represented by dashed lines with link capacity, respectively. We establish 10 flows that are represented by arrows, where each flow i is from node 1 to node $i+1$ via all the nodes in between. We consider uniform traffic where all flows have packet arrivals at each time-slot following Poisson distribution with the same mean rate $\lambda > 0$. We run our simulations with changing traffic load λ. Clearly, in this scenario, any traffic load with $\lambda < 0.5$ is feasible. We use $\epsilon = 0.005$ for HQ-MWS, PLQ-MWS, and FLQ-MWS. We evaluate the performance by measuring average packet delays (in unit of time-slot) over all the delivered packets (that reach their respective destination nodes) in the network.

Fig. 1(b) plots the average delays under different offered loads to examine the performance limits of different scheduling schemes. Each result represents a simulation run that lasts for 10^7 time-slots. Since the optimal throughput region Λ^* is defined as the set of arrival rate vectors under which queue lengths and thus delays remain finite, we can consider the traffic load, under which the average delay increases rapidly, as the boundary of the optimal throughput region. Fig. 1(b) shows that all schemes achieve the same boundary (i.e., $\lambda < 0.5$), which supports our theoretical results on throughput optimality. Moreover, all the three proposed schemes achieve substantially better delay performance than the back-pressure algorithm. This is because under the back-pressure algorithm, the queue lengths have to build up along the route a flow takes from the destination to the source, and in general, earlier hop link has a larger queue length. This leads to poor delay performance

[2]It is also called the *primary* or *1-hop* interference model, where two links sharing a common node cannot be activated simultaneously. It has been known as a good representation for Bluetooth or FH-CDMA networks [2].



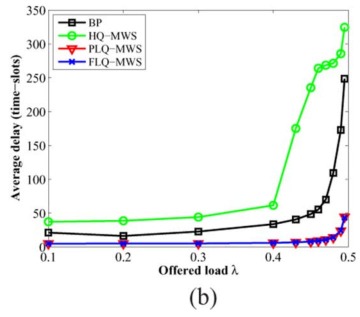
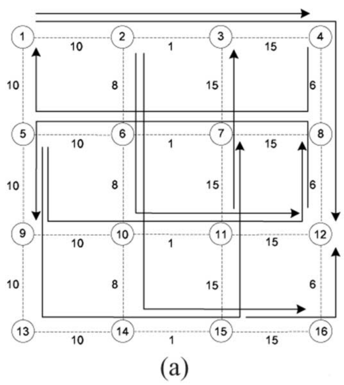


Fig. 2. Performance of BP, HQ-MWS, PLQ-MWS, and FLQ-MWS in a linear network topology ($\epsilon = 0.005$). (a) Linear network topology with 10 links. (b) Average delay.

especially when the route of a flow is lengthy, which is the case in Fig. 1(a). Note that in this specific scenario, there is only one per-hop queue at each link under HQ-MWS. Hence, HQ-MWS is equivalent to PLQ-MWS and FLQ-MWS in this scenario, which explains why the three proposed schemes perform the same as in Fig. 1(b).

Second, we evaluate the performance of the proposed schemes in the same linear network as in the previous case while reversing the direction of each flow. The new topology is illustrated in Fig. 2(a). In this scenario, the number of per-hop queues HQ-MWS maintains for each link is the same as the number of flows passing through that link. Hence, HQ-MWS is expected to operate differently from PLQ-MWS and FLQ-MWS and achieves different (and potentially poorer) delay performance. All the other simulation settings are kept the same as in the previous case. Fig. 2(b) shows that all schemes achieve the same boundary (i.e., $\lambda < 0.5$) in this scenario, which again supports our theoretical results on throughput performance. However, we observe that HQ-MWS has the worst delay performance, while PLQ-MWS and FLQ-MWS achieve substantially better performance. This is because PLQ-MWS and FLQ-MWS transmit packets more efficiently and do not waste service as long as there are enough packets at the activated link, while the back-pressure algorithm and HQ-MWS maintain multiple queues for each link and may waste service if the activated queue has less packets than the link capacity. HQ-MWS has larger delays than the back-pressure algorithm because the scheduling decisions of HQ-MWS are based on the shadow queue lengths rather than the actual queue lengths: A queue with very small (or even zero) queue length could be activated. This introduces another type of inefficiency in HQ-MWS. Note that PLQ-MWS and FLQ-MWS also make scheduling decisions based on the shadow queue lengths. However, their performance improvement from a single queue per link dominates delay increases from the inefficiency. These observations imply that maintaining per-link queues not only simplifies the data structure, but also improves scheduling efficiency and reduces delays.

Finally, we evaluate the performance of all the proposed schemes in a larger grid network with 16 nodes and 24 links as shown in Fig. 3(a), where the capacity of each link has been shown beside the link and carefully assigned to avoid traffic symmetry. Similar types of grid networks have been adopted in the literature (e.g., [4], [6], and [24]) to numerically evaluate scheduling performance. We establish 10 multihop flows that are represented by arrows in Fig. 3(a). Again, we consider



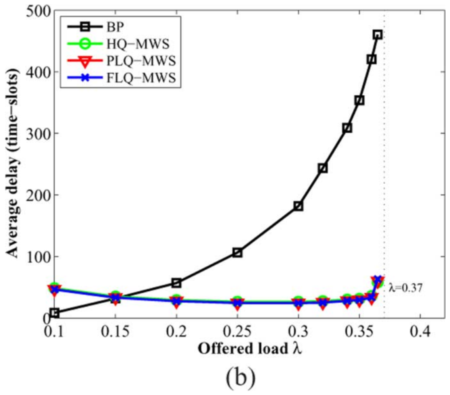


Fig. 3. Performance of all the proposed scheduling schemes in a grid network with 16 nodes and 24 links. In Fig. 3(b), the vertical dotted line $\lambda = 0.37$ denotes an upper bound for the feasible values of λ. (a) Grid network topology. (b) Average delay for MWS schemes with $\epsilon = 0.05$.

uniform traffic where each flow has independent packet arrivals at each time-slot following Poisson distribution with the same mean rate $\lambda > 0$. In this scenario, we can calculate an upper bound of $1/(4/8 + 2/10 + 2) = 10/27 \approx 0.37$ for the feasible value of λ by looking at the flows passing through node 6, which is the bottleneck in the network.

We choose $\epsilon = 0.05$ for HQ-MWS, PLQ-MWS, and FLQ-MWS. Under each scheduling scheme along with the back-pressure algorithm, we measure average packet delays under different offered loads to examine their performance limits. Fig. 3(b) shows that the proposed schemes have higher packet delays than the back-pressure algorithm when traffic load is light (e.g., $\lambda < 0.15$). This is due to the aforementioned inefficiency under the proposed schemes: Since the scheduling decisions are based on the shadow queue lengths rather than the actual queue lengths, queues with very small (or even zero) queue length can be activated. However, the effect tends to decrease with heavier traffic load since the queue lengths are likely to be large. The results also show that the proposed schemes consistently outperform the back-pressure algorithm when $\lambda > 0.15$. Note that with $\epsilon = 0.05$, the shadow traffic rate vector is outside the optimal throughput region when $\lambda > 0.37/(1 + 0.05) \approx 0.35$, however, interestingly, the schedules chosen based on the shadow queue lengths can still stabilize the data queues even if $0.35 < \lambda < 0.37$ (which is still feasible). Note that in the above scenario, FLQ-MWS does not guarantee throughput optimality since flows $(5 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 8)$ and $(12 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 9)$ form a loop. However, the results in Fig. 3(b) suggest that all the schemes, including

FLQ-MWS, empirically achieve the optimal throughput performance. This opens up an interesting question about throughput performance of FLQ-MWS in general settings.

VI. Conclusion

In this paper, we developed scheduling policies with per-hop or per-link queues and a shadow algorithm to achieve the overall goal of removing per-flow or per-destination information requirement, simplifying queue structure, exploiting only local information, and potentially reducing delay. We showed throughput optimality of the proposed schemes that use only the readily available hop-count information, using fluid limit techniques via an inductive argument. We further simplified the solution using FIFO queueing discipline with per-link queues and showed that this is also throughput-optimal in networks without flow-loops. The problem of proving throughput optimality in general networks with algorithms (like FLQ-MWS) that use only per-link information remains an important open and challenging problem. Furthermore, it is also worthwhile to investigate the problem with dynamic routing and see if per-flow and per-destination information can be removed even when routes are not fixed.

Appendix A
Proof of Proposition 1

To begin with, let $Q(t) \triangleq [Q_{l,k}(t)]$ and $\hat{Q}(t) \triangleq [\hat{Q}_{l,k}(t)]$ denote the queue length vector and the shadow queue length vector at time-slot t, respectively. We use $\|\cdot\|$ to denote the L_1-norm of a vector, e.g., $\|Q(t)\| = \sum_{l\in\mathcal{E}}\sum_{k=1}^{L^{\max}} Q_{l,k}(t)$. We let $m_{l,k}(i)$ be the index of the flow to which the ith packet of queue $Q_{l,k}$ belongs. In particular, $m_{l,k}(1)$ indicates the index of the flow to which the head-of-line packet of queue $Q_{l,k}$ belongs. We define the state of queue $Q_{l,k}$ at time-slot t as $\mathcal{Q}_{l,k}(t) = [m_{l,k}(1),\ldots,m_{l,k}(Q_{l,k}(t))]$ in an increasing order of the arriving time, or an empty sequence if $Q_{l,k}(t) = 0$. Then, we denote its vector by $\mathcal{Q}(t) \triangleq [\mathcal{Q}_{l,k}(t)]$. Define $\mathbb{Z}_{\mathcal{S}} \triangleq \{1,2,\ldots,|\mathcal{S}|\}$, and let $\mathbb{Z}_{\mathcal{S}}^{\infty}$ be the set of finitely terminated sequences taking values in $\mathbb{Z}_{\mathcal{S}}$. It is evident that $\mathcal{Q}_{l,k}(t) \in \mathbb{Z}_{\mathcal{S}}^{\infty}$, and hence $\mathcal{Q}(t) \in (\mathbb{Z}_{\mathcal{S}}^{\infty})^{|\mathcal{E}|\times L^{\max}}$. We define $\mathcal{X}(t) \triangleq \left(\mathcal{Q}(t), \hat{Q}(t), \frac{1}{t+1}A(t)\right)$, and then $\mathcal{X} = (\mathcal{X}(t), t \geq 0)$ is the process describing the behavior of the underlying system. Note that in the third term of $\mathcal{X}(t)$, we use $\frac{1}{t+1}A(t)$ instead of $\frac{1}{t}A(t)$ so that it is well defined when $t = 0$. Clearly, the evolution of $\mathcal{X}$ forms a countable Markov chain under HQ-MWS. We abuse the notation of L_1-norm by writing the norm of $\mathcal{X}(t)$ as $\|\mathcal{X}(t)\| \triangleq \|Q(t)\| + \lceil\|\hat{Q}(t)\|\rceil + \left\lceil \frac{1}{t+1}\|A(t)\|\right\rceil$. Let $\mathcal{X}^{(x)}$ denote a process $\mathcal{X}$ with an initial condition such that

$$\|\mathcal{X}^{(x)}(0)\| = x. \tag{9}$$

The following Lemma was derived in [19] for continuous-time countable Markov chains, and it follows from more general results in [25] for discrete-time countable Markov chains.

Lemma 6 (Theorem 4 of [15]): Suppose that there exist a $\xi > 0$ and a finite integer $T > 0$ such that for any sequence of processes $\{\frac{1}{x}\mathcal{X}^{(x)}(xT), x = 1,2,\ldots\}$, we have

$$\limsup_{x\to\infty} E\left[\frac{1}{x}\|\mathcal{X}^{(x)}(xT)\|\right] \leq 1-\xi. \tag{10}$$

Then, the Markov chain $\mathcal{X}$ is stable.

Lemma 6 implies the stability of the network. A stability criterion of type (10) leads to a fluid limit approach [18] to the stability problem of queueing systems. We start our analysis by establishing the *fluid limit model* as in [15] and [18]. We define another process $\mathcal{Y} \triangleq (F, U, Q, \Pi, \Psi, A, D, P, \hat{Q}, \hat{\Pi}, \hat{\Psi}, \hat{A}, \hat{D}, \hat{P})$, where the tuple denotes a list of vector processes. Clearly, a sample path of $\mathcal{Y}^{(x)}$ uniquely defines the sample path of $\mathcal{X}^{(x)}$. Then, we extend the definition of $\mathcal{Y}$ to each continuous time $t \geq 0$ as $\mathcal{Y}^{(x)}(t) \triangleq \mathcal{Y}^{(x)}(\lfloor t\rfloor)$.

Recall that a sequence of functions $f_n(\cdot)$ is said to converge to a function $f(\cdot)$ *uniformly over compact (u.o.c.)* intervals if for all $t \geq 0$, $\lim_{n\to\infty}\sup_{0\leq t'\leq t}|f_n(t') - f(t')| = 0$. Next, we consider a sequence of processes $\left\{\frac{1}{x_n}\mathcal{Y}^{(x_n)}(x_n\cdot)\right\}$ that is scaled both in time and space. Then, using the techniques of [18, Theorem 4.1] or [15, Lemma 1], we can show the convergence properties of the sequences in the following lemma.

Lemma 7: With probability one, for any sequence of processes $\left\{\frac{1}{x_n}\mathcal{Y}^{(x_n)}(x_n\cdot)\right\}$, where $\{x_n\}$ is a sequence of positive integers with $x_n \to \infty$, there exists a subsequence $\{x_{n_j}\}$ with $x_{n_j} \to \infty$ as $j \to \infty$ such that the following *u.o.c. convergences* hold:

$$\frac{1}{x_{n_j}}F_s^{(x_{n_j})}(x_{n_j}t) \to f_s(t) \tag{11}$$

$$\frac{1}{x_{n_j}}U_{s,k}^{(x_{n_j})}(x_{n_j}t) \to u_{s,k}(t) \tag{12}$$

$$\frac{1}{x_{n_j}}A_{l,k}^{(x_{n_j})}(x_{n_j}t) \to a_{l,k}(t) \tag{13}$$

$$\frac{1}{x_{n_j}}\hat{A}_{l,k}^{(x_{n_j})}(x_{n_j}t) \to \hat{a}_{l,k}(t) \tag{14}$$

$$\frac{1}{x_{n_j}}Q_{l,k}^{(x_{n_j})}(x_{n_j}t) \to q_{l,k}(t) \tag{15}$$

$$\frac{1}{x_{n_j}}\hat{Q}_{l,k}^{(x_{n_j})}(x_{n_j}t) \to \hat{q}_{l,k}(t) \tag{16}$$

$$\frac{1}{x_{n_j}}D_{l,k}^{(x_{n_j})}(x_{n_j}t) \to d_{l,k}(t) \tag{17}$$

$$\frac{1}{x_{n_j}}\hat{D}_{l,k}^{(x_{n_j})}(x_{n_j}t) \to \hat{d}_{l,k}(t) \tag{18}$$

$$\frac{1}{x_{n_j}}\int_0^{x_{n_j}t}\Pi_{l,k}^{(x_{n_j})}(\tau)d\tau \to \int_0^t \pi_{l,k}(\tau)d\tau \tag{19}$$

$$\frac{1}{x_{n_j}}\int_0^{x_{n_j}t}\hat{\Pi}_{l,k}^{(x_{n_j})}(\tau)d\tau \to \int_0^t \hat{\pi}_{l,k}(\tau)d\tau \tag{20}$$

$$\frac{1}{x_{n_j}}\int_0^{x_{n_j}t}\Psi_{l,k}^{(x_{n_j})}(\tau)d\tau \to \int_0^t \psi_{l,k}(\tau)d\tau \tag{21}$$

$$\frac{1}{x_{n_j}}\int_0^{x_{n_j}t}\hat{\Psi}_{l,k}^{(x_{n_j})}(\tau)d\tau \to \int_0^t \hat{\psi}_{l,k}(\tau)d\tau \tag{22}$$

$$\frac{1}{x_{n_j}}\int_0^{x_{n_j}t}P_{l,k}^{(x_{n_j})}(\tau)d\tau \to \int_0^t p_{l,k}(\tau)d\tau \tag{23}$$

$$\frac{1}{x_{n_j}}\int_0^{x_{n_j}t}\hat{P}_{l,k}^{(x_{n_j})}(\tau)d\tau \to \int_0^t \hat{p}_{l,k}(\tau)d\tau \tag{24}$$

where the functions f_s, $u_{s,k}$, $a_{l,k}$, $d_{l,k}$, $q_{l,k}$, $\hat{a}_{l,k}$, $\hat{d}_{l,k}$, $\hat{q}_{l,k}$ are Lipschitz continuous in $[0,\infty)$.

Note that the proof of the above lemma is quite standard using the techniques developed in [15], [18], and [26]. We provide the proof in our online technical report [17] for completeness.

Any set of limiting functions $(f, u, q, \pi, \psi, a, d, p, \hat{q}, \hat{\pi}, \hat{\psi}, \hat{a}, \hat{d}, \hat{p})$ is called a *fluid limit*. The family of these fluid limits is associated with our original stochastic network. The scaled sequences $\left\{\frac{1}{x_n}\mathcal{Y}^{(x_n)}(x_n\cdot)\right\}$ and their limits are referred to as a *fluid limit model* [16]. Since some of the limiting functions, namely f_s, $u_{s,k}$, $a_{l,k}$, $d_{l,k}$, $q_{l,k}$, $\hat{a}_{l,k}$, $\hat{d}_{l,k}$, $\hat{q}_{l,k}$, are Lipschitz continuous in $[0,\infty)$, they are absolutely continuous. Therefore, these limiting functions are differentiable at almost all time $t \in [0,\infty)$, which we call *regular* time.

Next, we will present the *fluid model equations* of the system, i.e., (25)–(40). Fluid model equations can be thought of as belonging to a *fluid network*, which is the deterministic equivalence of the original stochastic network. Any set of functions satisfying the fluid model equations is called a *fluid model solution* of the system. We show in the following lemma that any fluid limit is a fluid model solution.

Lemma 8: Any fluid limit $(f, u, q, \pi, \psi, a, d, p, \hat{q}, \hat{\pi}, \hat{\psi}, \hat{a}, \hat{d}, \hat{p})$ satisfies the following equations:

$$f_s(t) = \lambda_s t \tag{25}$$

$$q_{l,k}(t) = q_{l,k}(0) + a_{l,k}(t) - d_{l,k}(t) \tag{26}$$

$$a_{l,k}(t) = \sum_s H^s_{l,k} u_{s,k}(t) \tag{27}$$

$$a_{l,k}(t) = \int_0^t p_{l,k}(\tau)d\tau \tag{28}$$

$$d_{l,k}(t) = \int_0^t \psi_{l,k}(\tau)d\tau \tag{29}$$

$$\psi_{l,k}(t) \le \pi_{l,k}(t) \tag{30}$$

$$\frac{d}{dt}q_{l,k}(t) = p_{l,k}(t) - \psi_{l,k}(t) \tag{31}$$

$$\frac{d}{dt}q_{l,k}(t) = \begin{cases} p_{l,k}(t) - \pi_{l,k}(t), & \text{if } q_{l,k}(t) > 0 \\ (p_{l,k}(t) - \pi_{l,k}(t))^+, & \text{otherwise} \end{cases} \tag{32}$$

$$\hat{q}_{l,k}(t) = \hat{q}_{l,k}(0) + \hat{a}_{l,k}(t) - \hat{d}_{l,k}(t) \tag{33}$$

$$\hat{a}_{l,k}(t) = \int_0^t \hat{p}_{l,k}(\tau)d\tau \tag{34}$$

$$\hat{d}_{l,k}(t) = \int_0^t \hat{\psi}_{l,k}(\tau)d\tau \tag{35}$$

$$\hat{\psi}_{l,k}(t) \le \hat{\pi}_{l,k}(t) \tag{36}$$

$$\frac{d}{dt}\hat{q}_{l,k}(t) = \hat{p}_{l,k}(t) - \hat{\psi}_{l,k}(t) \tag{37}$$

$$\frac{d}{dt}\hat{q}_{l,k}(t) = \begin{cases} \hat{p}_{l,k}(t) - \hat{\pi}_{l,k}(t), & \text{if } \hat{q}_{l,k}(t) > 0 \\ (\hat{p}_{l,k}(t) - \hat{\pi}_{l,k}(t))^+, & \text{otherwise} \end{cases} \tag{38}$$

$$\|q(0)\| + \|\hat{q}(0)\| \le 1 \tag{39}$$

$$\pi_{l,k}(t) = \hat{\pi}_{l,k}(t). \tag{40}$$

The proof of the above lemma is straightforward and is provided in our online technical report [17] for completeness.

Due to the result of Lemma 6, we want to show that the stability criterion of (10) holds. Note that from system causality, we have $a_{l,k}(t) \le t\sum_s H^s_{l,k}\lambda_s + \sum_s\sum_h q_{s,h}(0)$ for all link $l \in \mathcal{E}$ and all $1 \le k \le L^{\max}$, for all $t \ge 0$. Then, we have

$$\lim_{j\to\infty} \frac{1}{x_{n_j}} \|A^{(x_{n_j})}(x_{n_j}t)\| \le \sum_l\sum_k \left(t\sum_s H^s_{l,k}\lambda_s + \sum_s\sum_h q_{s,h}(0)\right)$$

almost surely, and thus

$$\lim_{j\to\infty} \frac{1}{x_{n_j}} \left\lceil \frac{1}{x_{n_j}t+1}\|A^{(x_{n_j})}(x_{n_j}t)\| \right\rceil = 0 \tag{41}$$

almost surely, for all $t \ge 0$. Therefore, it remains to be shown that the fluid limit model for the joint system of data queues and shadow queues is stable (Lemma 14). Then, by uniform integrability of the sequence $\left\{\frac{1}{x}\|\mathcal{X}^{(x)}(xT)\|, x = 1, 2, \ldots\right\}$, it implies that (10) holds. We divide the proof of Lemma 14 into two parts: 1) in Lemma 11, we show that the subsystem consisting of shadow queues is stable; 2) in Lemma 13, the subsystem consisting of data queues is stable. Before proving Lemmas 11 and 13, we state and prove Lemmas 9 and 12, which are used to prove Lemmas 11 and 13, respectively.

The following lemma shows that the instantaneous shadow arrival rate is bounded in the fluid limit and is used to show that the fluid limit model for the subsystem consisting of shadow queues is stable under HQ-MWS.

Lemma 9: For all (scaled) time $t > 0$, and for all links $l \in \mathcal{E}$ and $1 \le k \le L^{\max}$, with probability one, the following inequality holds:

$$\hat{p}_{l,k}(t) \le (1+\epsilon)\left(\sum_s H^s_{l,k}\lambda_s + \frac{1}{t}\right) \tag{42}$$

and in particular

$$\hat{p}_{l,1}(t) = (1+\epsilon)\sum_s H^s_{l,1}\lambda_s. \tag{43}$$

Proof: We start by stating the following lemma, which will be used to prove Lemma 9.

Lemma 10: If a sequence $\{F(n), n = 1, 2, \ldots\}$ satisfies $\lim_{n\to\infty} F(n) = f$, then the following holds:

$$\lim_{n\to\infty} \frac{\sum_{\tau=1}^n F(\tau)}{n} = f.$$

The proof of the above lemma is quite standard and is omitted in this paper. Interested readers can find the full proof in our online technical report [17].

Now, we prove Lemma 9. Note that we have

$$A_{l,k}(t) \le \sum_{s\in\mathcal{S}} H^s_{l,k}F_s(t) + \sum_{i\in\mathcal{E}}\sum_{h=1}^{L^{\max}} Q_{i,h}(0) \tag{44}$$

for any $t > 0$ and for any link $l \in \mathcal{E}$ and $1 \le k \le L^{\max}$ due to system causality.

Since the arrival processes satisfy SLLN of type (1), we obtain from Lemma 10 that with probability one

$$\lim_{n\to\infty} \frac{\sum_{\tau=1}^n \frac{F_s(\tau)}{\tau}}{n} = \lambda_s, \qquad \text{for all } s \in \mathcal{S}. \tag{45}$$

Note that we will omit the superscript (x_{n_j}) of the random variables (depending on the choice of the sequence $\{x_{n_j}\}$) throughout the rest of the proof for notational convenience (e.g., we use $A_{l,k}(t)$ to denote $A_{l,k}^{(x_{n_j})}(t)$). Then, for all regular time $t > 0$, all links $l \in \mathcal{E}$ and $1 \le k \le L^{\max}$, we have

$$\begin{aligned}
&\hat{p}_{l,k}(t)\\
&= \frac{d}{dt}\int_0^t \hat{p}_{l,k}(\tau)d\tau\\
&= \lim_{\delta\to 0}\frac{\int_0^{t+\delta}\hat{p}_{l,k}(\tau)d\tau - \int_0^t \hat{p}_{l,k}(\tau)d\tau}{\delta}\\
&\overset{(24)}{=} \lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\hat{P}_{l,k}(\tau)}{\delta x_{n_j}}\\
&\overset{(4)}{=} (1+\epsilon)\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{A_{l,k}(\tau)}{\tau}}{\delta x_{n_j}}\\
&\overset{(44)}{\le} (1+\epsilon)\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{\sum_s H^s_{l,k}F_s(\tau)+\sum_i\sum_h Q_{i,h}(0)}{\tau}}{\delta x_{n_j}}\\
&= (1+\epsilon)\sum_s H^s_{l,k}\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_{\tau=1}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{F_s(\tau)}{\tau}}{\lfloor (t+\delta)x_{n_j}\rfloor}\cdot\frac{\lfloor (t+\delta)x_{n_j}\rfloor}{\delta x_{n_j}}\\
&\quad - (1+\epsilon)\sum_s H^s_{l,k}\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_{\tau=1}^{\lceil tx_{n_j}\rceil-1}\frac{F_s(\tau)}{\tau}}{\lceil tx_{n_j}\rceil-1}\cdot\frac{\lceil tx_{n_j}\rceil-1}{\delta x_{n_j}}\\
&\quad + (1+\epsilon)\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_i\sum_h Q_{i,h}(0)}{\delta x_{n_j}}\cdot\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{1}{\tau}\\
&\le (1+\epsilon)\sum_s H^s_{l,k}\lambda_s\lim_{\delta\to 0}\left(\frac{t+\delta}{\delta}-\frac{t}{\delta}\right)+(1+\epsilon)\frac{1}{t}\\
&= (1+\epsilon)\left(\sum_s H^s_{l,k}\lambda_s+\frac{1}{t}\right)
\end{aligned}$$

where in the last inequality, the first term is from (45), and the second term is from the fact that: 1) $\|q(0)\|+\|\hat{q}(0)\| \le 1$ implies $\lim_{j\to\infty}\frac{\sum_j\sum_h Q_{j,h}(0)}{x_{n_j}} \le 1$; and 2) $\lim_{j\to\infty}\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{1}{\tau} = \log\frac{t+\delta}{t}$ [see our technical report [17] for the derivation of 2)]. Combining 1) and 2), we have

$$\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_i\sum_h Q_{i,h}(0)}{\delta x_{n_j}}\cdot\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{1}{\tau} \le \lim_{\delta\to 0}\left(\frac{1}{\delta}\cdot\log\frac{t+\delta}{t}\right) = \frac{1}{t}$$

where the equality is from the L'Hospital's Rule.

So far, we have shown (42). Note that when $k=1$, (44) reduces to $A_{l,1}(t) = \sum_{s\in\mathcal{S}} H^s_{l,1}F_s(t)$. Then, in the above derivation of $\hat{p}_{l,k}(t)$, the first inequality [which follows from (44)] becomes an equality, and the right-hand side of this inequality becomes

$$(1+\epsilon)\lim_{\delta\to 0}\lim_{j\to\infty}\frac{\sum_{\tau=\lceil tx_{n_j}\rceil}^{\lfloor (t+\delta)x_{n_j}\rfloor}\frac{\sum_s H^s_{l,1}F_s(\tau)}{\tau}}{\delta x_{n_j}}.$$

Hence, we obtain (43). ■

Remark: Lemma 9 holds when the exogenous arrival processes satisfy the SLLN, and the shadow arrivals are controlled as in (4). Note that Lemma 9 does not hold for data queues $Q_{l,k}$ since the data arrival processes do not satisfy (4) due to their dependency on the service of the previous hop queues. Lemma 9 is important to proving the stability of the shadow queues and implies that in the fluid limit model, the instantaneous arrival rate of shadow queues is strictly inside the optimal throughput region Λ^* after a finite time.

Then, in the following lemma, we show that the fluid limit model for the subsystem consisting of shadow queues is stable[3] under HQ-MWS.

Lemma 11: The fluid limit model for the subsystem of shadow queues $\hat{q}$ operating under HQ-MWS satisfies the following: For any $\zeta > 0$, there exists a finite $T_1 > 0$ such that for any fluid model solution with $\|\hat{q}(0)\| \le 1$, we have that with probability one

$$\|\hat{q}(t)\| \le \zeta, \qquad \text{for all } t \ge T_1$$

for any arrival rate vector strictly inside Λ^*.

Proof: Suppose λ is strictly inside Λ^*, we can find a small $\epsilon > 0$ such that $(1+\epsilon)\lambda$ is strictly inside Λ^*. Then, there exists a vector $\phi \in \mathrm{Co}(\mathcal{M})$ such that $(1+\epsilon)\lambda < \phi$, i.e., $(1+\epsilon)\sum_s\sum_k H^s_{l,k}\lambda_s < \phi_l$, for all $l \in \mathcal{E}$. Let β denote the smallest difference between the two vectors, which is defined as $\beta \triangleq \min_{l\in\mathcal{E}}\left(\phi_l - (1+\epsilon)\sum_s\sum_k H^s_{l,k}\lambda_s\right)$. Clearly, we have $\beta > 0$. Let T' be a finite time such that $T' > \frac{(1+\epsilon)L^{\max}}{\beta}$, then we have $(1+\epsilon)\left(\sum_s\sum_k H^s_{l,k}\lambda_s + \frac{L^{\max}}{T'}\right) < \phi_l$. Let $\phi_{l,k} \triangleq (1+\epsilon)\left(\sum_s H^s_{l,k}\lambda_s + \frac{1}{T'}\right) + \frac{\phi_l - (1+\epsilon)\left(\sum_k\sum_s H^s_{l,k}\lambda_s + \frac{L^{\max}}{T'}\right)}{L^{\max}}$. Then, we have

$$\sum_k \phi_{l,k} = \phi_l \tag{46}$$

and from (42), we have

$$\hat{p}_{l,k}(t) \le (1+\epsilon)\left(\sum_s H^s_{l,k}\lambda_s + \frac{1}{T'}\right) < \phi_{l,k} \tag{47}$$

for all regular time $t \ge T'$. This implies that the instantaneous arrival rate of shadow queues is strictly inside the optimal throughput region Λ^*.

We consider a quadratic-form Lyapunov function $\hat{V}(\hat{q}(t)) = \frac{1}{2}\sum_l\sum_k(\hat{q}_{l,k}(t))^2$. It is sufficient to show that for any $\zeta_1 > 0$, there exist $\zeta_2 > 0$ and a finite time $T^* > 0$ such that at any regular time $t \ge T^*$, $\hat{V}(\hat{q}(t)) \ge \zeta_1$ implies $\frac{D^+}{dt^+}\hat{V}(\hat{q}(t)) \le -\zeta_2$. Choose $T^* = T'$. Since $\hat{q}(t)$ is differentiable for any regular time $t \ge T^*$, we can obtain the derivative of $\hat{V}(\hat{q}(t))$ as

$$\begin{aligned}
\frac{D^+}{dt^+}\hat{V}(\hat{q}(t)) &= \sum_l\sum_k \hat{q}_{l,k}(t)\cdot(\hat{p}_{l,k}(t) - \hat{\pi}_{l,k}(t))\\
&= \sum_l\sum_k \hat{q}_{l,k}(t)\cdot(\hat{p}_{l,k}(t) - \phi_{l,k})\\
&\quad + \sum_l\sum_k \hat{q}_{l,k}(t)\cdot(\phi_{l,k} - \hat{\pi}_{l,k}(t))
\end{aligned} \tag{48}$$

where $\frac{D^+}{dt^+}\hat{V}(\hat{q}(t)) = \lim_{\delta\downarrow 0}\frac{\hat{V}(\hat{q}(t+\delta)) - \hat{V}(\hat{q}(t))}{\delta}$, and the first equality is from (38).

[3]Similar to [15], we consider a weaker criterion for the stability of the fluid limit model in Lemma 11, which can imply the stability of the original system from Lemma 6.

Let us choose $\zeta_3 > 0$ such that $\hat{V}(\hat{q}(t)) \geq \zeta_1$ implies $\max_{l\in\mathcal{E},1\leq k\leq L^{\max}} \hat{q}_{l,k}(t) \geq \zeta_3$. Then, in the final result of (48), we can conclude that the first term is bounded. That is

$$\begin{aligned}\sum_l \sum_k & \hat{q}_{l,k}(t)\cdot(\hat{p}_{l,k}(t)-\phi_{l,k}) \\ &\leq -\zeta_3 \min_{l,k}(\phi_{l,k}-\hat{p}_{l,k}(t)) \\ &\leq -\zeta_3 \min_{l,k}\left(\phi_{l,k}-(1+\epsilon)\left(\sum_s H^s_{l,k}\lambda_s+\frac{1}{T'}\right)\right) \\ &\triangleq -\zeta_2 < 0\end{aligned}$$

where the second inequality is from (47). For the second term, since HQ-MWS chooses schedules that maximize the shadow queue length weighted rate, the service rate satisfies that

$$\hat{\pi}(t) \in \operatorname{argmax}_{\phi\in Co(\mathcal{M})} \sum_l \hat{q}_{l,k^*(l)}(t)\cdot\phi_l \tag{49}$$

where: 1) $\hat{q}_{l,k^*(l)}(t) = \max_k \hat{q}_{l,k}(t)$; and 2) $\hat{\pi}_l(t) = \sum_k \hat{\pi}_{l,k}(t)$ with $\hat{\pi}_{l,k}(t) = 0$ when $\hat{q}_{l,k}(t) < \hat{q}_{l,k^*(l)}(t)$. This implies that $\sum_l\sum_k \hat{q}_{l,k}(t)\cdot\phi_{l,k} \leq \sum_l\sum_k \hat{q}_{l,k^*}(t)\cdot\phi_{l,k} = \sum_l \hat{q}_{l,k^*}(t)\cdot\phi_l \leq \sum_l \hat{q}_{l,k^*}(t)\cdot\hat{\pi}_l(t) = \sum_l\sum_k \hat{q}_{l,k}(t)\cdot\hat{\pi}_{l,k}(t)$, for all $\phi \in Co(\mathcal{M})$, where the first equality and the second inequality are from (46) and (49), respectively. Then, we obtain that the second term of (48) is nonpositive. This shows that $\hat{V}(\hat{q}(t)) \geq \zeta_1$ implies $\frac{D^+}{dt^+}\hat{V}(\hat{q}(t)) \leq -\zeta_2$ for all regular time $t \geq T^*$. Hence, it immediately follows that for any $\zeta > 0$, there exists a finite $T_1 \geq T^* > 0$ such that $\|\hat{q}(t)\| \leq \zeta$, for all $t \geq T_1$. ■

We next present Lemma 12 that is used to show that the subsystem consisting of data queues is stable under HQ-MWS in the fluid limit model.

Lemma 12: If data queues $q_{l,j}$ are stable for all $l \in \mathcal{E}$ and for all $j \leq k$, then there exists a finite $T_1^k > 0$ such that for all regular time $t \geq T_1^k$ and for all $l \in \mathcal{E}$, we have that with probability one

$$\hat{p}_{l,k+1}(t) \geq (1+\epsilon)\sum_s H^s_{l,k+1}\lambda_s.$$

The proof follows a similar argument used in the proof for Lemma 9 and is provided in our technical report [17].

In the following lemma, using a hop-by-hop inductive argument, we show that the fluid model for the subsystem of data queues is stable.

Lemma 13: The fluid limit model of the subsystem of data queues q operating under HQ-MWS is stable, i.e., there exists a finite $T_2 > 0$ such that, for any fluid model solution with $\|q(0)\| \leq 1$, we have

$$\|q(t)\| = 0, \qquad \text{for all } t \geq T_2$$

for any arrival rate vector strictly inside Λ^*.

Proof: We prove the stability of data queues by induction. Suppose λ is strictly inside Λ^*, and the subsystem of shadow queues $\hat{q}$ is stable from Lemma 11. Let us choose sufficiently small $\zeta > 0$ such that $\zeta < \epsilon \min_s \lambda_s$, then there exists a finite time $T_1 > 0$ such that we have $\|\hat{q}(t)\| \leq \zeta$ for any regular time $t \geq T_1$. Thus, we have $\hat{\psi}_{l,k}(t) \geq \hat{p}_{l,k}(t) - \zeta$ from (37), for all $t \geq T_1$. Hence, for all data queues and all regular time $t \geq T_1$, we have

$$\pi_{l,k}(t) = \hat{\pi}_{l,k}(t) \geq \hat{p}_{l,k}(t) - \zeta \tag{50}$$

from (36) and (40).

Now we show by induction that all data queues are stable in the fluid limit model.

Base Case: First, note that $\pi_{l,1}(t) \geq (1+\epsilon)\sum_s H^s_{l,1}\lambda_s - \zeta$ from (43) and (50). Consider a subsystem that contains only queue $q_{l,1}$. From $p_{l,1}(t) = \sum_s H^s_{l,1}\lambda_s$ and (32), we have $\frac{d}{dt}q_{l,1}(t) = p_{l,1}(t) - \pi_{l,1}(t) \leq -\epsilon\sum_s H^s_{l,1}\lambda_s + \zeta < 0$, if $q_{l,1}(t) > 0$. This implies that the subsystem that contains only $q_{l,1}$ is stable, for all $l \in \mathcal{E}$.

Induction Step: Next, we show that if $q_{l,j}$ is stable for all $l \in \mathcal{E}$ and all $j \leq k$, then each queue $q_{l,k+1}$ is also stable for all $l \in \mathcal{E}$, where $1 \leq k < L^{\max}$.

Since $q_{l,j}(t)$ is stable for all $l \in \mathcal{E}$ and all $j \leq k$, i.e., there exists a finite $T_1^k > 0$ such that $q_{l,j}(t) = 0$ for all regular time $t \geq T_1^k$, then $u_{s,k+1}(t) = u_{s,k}(t) + q_{s,k}(0) = \cdots = u_{s,1}(t) + \sum_{h<k} q_{s,h}(0) = \lambda_s t + \sum_{h<k} q_{s,h}(0)$ for all $s \in \mathcal{S}$ and for all regular time $t \geq T_1^k$. Thus, we have $a_{l,k+1}(t) = t\sum_s H^s_{l,k+1}\lambda_s + \sum_s H^s_{l,k+1}\sum_{h<k} q_{s,h}(0)$ from (27), and $p_{l,k+1}(t) = \sum_s H^s_{l,k+1}\lambda_s$ from (28) by taking derivative, for all $l \in \mathcal{E}$ and all regular time $t \geq T_1^k$. Then, note that we have $\hat{p}_{l,k+1}(t) \geq (1+\epsilon)\sum_s H^s_{l,k+1}\lambda_s$ from Lemma 12. Hence, we have $\pi_{l,k+1}(t) \geq (1+\epsilon)\sum_s H^l_{s,k+1}\lambda_s - \zeta$ from (50). Therefore, we have $\frac{d}{dt}q_{l,k+1}(t) = p_{l,k+1}(t) - \pi_{l,k+1}(t) \leq -\epsilon\sum_s H^l_{s,k+1}\lambda_s + \zeta < 0$, if $q_{l,k+1}(t) > 0$. This implies that $q_{l,k+1}$ is stable for all $l \in \mathcal{E}$.

Therefore, the result follows by induction. ■

The following lemma says that the fluid limit model of joint data queues and shadow queues is stable, which follows immediately from Lemmas 11 and 13.

Lemma 14: The fluid limit model of the joint system of data queues q and shadow queues $\hat{q}$ operating under HQ-MWS satisfies the following: For any $\zeta > 0$, there exists a finite $T_2 > 0$ such that for any fluid model solution with $\|q(0)\| + \|\hat{q}(0)\| \leq 1$, we have that with probability one

$$\|q(t)\| + \|\hat{q}(t)\| \leq \zeta, \qquad \text{for all } t \geq T_2$$

for any arrival rate vector strictly inside Λ^*.

Now, consider any fixed sequence of processes $\left\{\frac{1}{x}\mathcal{X}^{(x)}(xt), x = 1, 2, \ldots\right\}$ (for simplicity also denoted by $\{x\}$). By Lemmas 7 and 14, we have that for any fixed $\xi_1 > 0$, we can always choose a large enough integer $T > 0$ such that for any subsequence $\{x_n\}$ of $\{x\}$, there exists a further (sub)subsequence $\{x_{n_j}\}$ such that

$$\begin{aligned}\lim_{j\to\infty} \frac{1}{x_{n_j}} \left(\left\|Q^{(x_{n_j})}(x_{n_j}T)\right\| + \left\lceil\left\|\hat{Q}^{(x_{n_j})}(x_{n_j}T)\right\|\right\rceil\right) \\ = \|q(T)\| + \|\hat{q}(T)\| \leq \xi_1\end{aligned}$$

almost surely. This, along with (41), implies that

$$\lim_{j\to\infty} \frac{1}{x_{n_j}} \left\|\mathcal{X}^{(x_{n_j})}(x_{n_j}T)\right\| \leq \xi_1$$

almost surely, which in turn implies (for small enough ζ_1) that

$$\limsup_{x\to\infty} \frac{1}{x}\|\mathcal{X}^{(x)}(xT)\| \leq \xi_1 \triangleq 1 - \xi < 1 \tag{51}$$

almost surely. This is because there must exist a subsequence of $\{x\}$ that converges to the same limit as $\limsup_{x\to\infty} \frac{1}{x}\|\mathcal{X}^{(x)}(xT)\|$.

We can show that the sequence $\left\{\frac{1}{x}\|\mathcal{X}^{(x)}(xT)\|, x = 1, 2, \ldots\right\}$ is uniformly integrable. The proof of showing uniform integrability is very standard and is provided in our online technical report [17]. Then, the almost surely convergence in (51) along with uniform integrability implies the following convergence in the mean:

$$\limsup_{x\to\infty} E\left[\frac{1}{x}\|\mathcal{X}^{(x)}(xT)\|\right] \le 1-\xi.$$

Since the above convergence holds for any sequence of processes $\left\{\frac{1}{x}\mathcal{X}^{(x)}(xT), x=1,2,\ldots\right\}$, the condition of type (10) in Lemma 6 is satisfied. This completes the proof of Proposition 1.

Appendix B
Stability of the Shadow Queues Under LQ-MWS

Similarly to (11)–(24), we can establish the fluid limits of the system: $(f,u,q,\pi,\psi,a,d,p,\hat{q},\hat{\pi},\hat{\psi},\hat{a},\hat{d},\hat{p})$.

We present a lemma similar to Lemma 9. This will be used to show that the fluid limit model for the subsystem consisting of shadow queues is stable under LQ-MWS. We omit its proof since it follows the same line of analysis for the proof of Lemma 9.

Lemma 15: For all (scaled) time $t>0$ and for all links $l\in\mathcal{E}$, we have that with probability one

$$\hat{p}_l(t) \le (1+\epsilon)\left(\sum_s\sum_k H^s_{l,k}\lambda_s + \frac{1}{t}\right). \qquad (52)$$

Now, we can show that the fluid limit model for the subsystem of shadow queues $\hat{q}$ is stable under LQ-MWS.

Lemma 16: The fluid limit model for the subsystem of shadow queues $\hat{q}$ operating under LQ-MWS satisfies the following: For any $\zeta>0$, there exists a finite $T_3>0$ such that for any fluid model solution with $\|\hat{q}(0)\|\le 1$, we have that with probability one

$$\|\hat{q}(t)\| \le \zeta, \qquad \text{for all } t\ge T_3$$

for any arrival rate vector strictly inside Λ^*.

The proof is similar to that of Lemma 11 and is thus omitted.

Appendix C
Proof of Lemma 3

Recall that $\mathcal{L}(s)$ denotes the loop-free route of the flow s. We prove Lemma 3 in a constructive way, i.e., for a network where flows do not form loops, we will give an algorithm that generates a ranking such that the following statements in Lemma 3 hold: 1) for any flow $s\in\mathcal{S}$, the ranks are monotonically increasing when one traverses the links on the route of the flow s from l^s_1 to $l^s_{|\mathcal{L}(s)|}$, i.e., $r(l^s_i) < r(l^s_{i+1})$ for all $1\le i<|\mathcal{L}(s)|$; and 2) the packet arrivals at a link are either exogenous or forwarded from links with a smaller rank.

We start with some useful definitions.

Definition 1: Two flows $s_1, s_2\in\mathcal{S}$ are *connected* if they have common (directed) links on their routes, i.e., $\mathcal{L}(s_1)\bigcap\mathcal{L}(s_2)\neq\emptyset$, and *disconnected* otherwise. A sequence of flows $(\tau_1,\ldots,\tau_n)$ is a *communicating sequence* if every two adjacent flows τ_i and τ_{i+1} are connected with each other. Two flows s_1 and s_2 *communicate* if there exists a communicating sequence between s_1 and s_2.

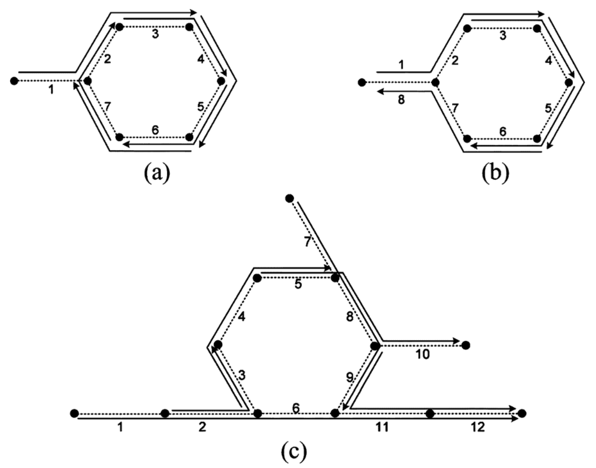


Fig. 4. Examples of different types of components. Links and flows are denoted by dashed lines with numbers and solid lines with arrows, respectively. Note that links without data flows are omitted (not numbered), and two numbers labeled beside a dashed line stand for two links with opposite directions, e.g., links 1 and 8 in (b). (a) Component containing a flow-loop (2, 3, 4, 5, 6, 7), and the component is not a flow-tree. (b) Flow-tree with one flow-path: (1, 2, 3, 4, 5, 6, 7, 8). (c) Flow-tree and with five flow-paths: $P_1=(1,2,3,4,5,8,10)$, $P_2=(1,2,6,11,12)$, $P_3=(7,8,10)$, $P_4=(7,8,9,11,12)$, and $P_5=(1,2,3,4,5,8,9,11,12)$.

Definition 2: Let $\mathcal{S}(l)\subseteq\mathcal{S}$ denote the set of flows passing through link l, and let $\mathcal{S}(\mathcal{Z})\triangleq\bigcup_{l\in\mathcal{Z}}\mathcal{S}(l)$ denote the set of flows passing through a set of links $\mathcal{Z}\subseteq\mathcal{E}$. A nonempty set of links $\mathcal{Z}$ is called a *component* if the following conditions are satisfied.

1) $\mathcal{Z}=\bigcup_{s\in\mathcal{S}(\mathcal{Z})}\mathcal{L}(s)$.
2) Either $|\mathcal{S}(\mathcal{Z})|=1$, or any two flows s_1, $s_2\in\mathcal{S}(\mathcal{Z})$ communicate.

Definition 3: Consider a component $\mathcal{Z}$; a sequence[4] of flows $(s_1,s_2,\ldots,s_N)\subseteq\mathcal{S}(\mathcal{Z})$, where $N\ge 2$, is said to form a *flow-loop*, if one can find two links $l^{s_n}_{i_n}$ and $l^{s_n}_{j_n}$ for each $n=1,2,\ldots,N$, satisfying:

1) $i_n<j_n$ for each $1\le n\le N$;
2) $\begin{cases} l^{s_n}_{j_n}=l^{s_{n+1}}_{i_{n+1}}, & \text{for each } n<N \\ l^{s_N}_{j_N}=l^{s_1}_{i_1}. \end{cases}$

An example of a component that contains a flow-loop is presented in Fig. 4(a), where the network consists of seven links and six flows. The routes of the flows are as follows: $\mathcal{L}(s_1)=(1,2,3)$, $\mathcal{L}(s_2)=(3,4)$, $\mathcal{L}(s_3)=(4,5)$, $\mathcal{L}(s_4)=(5,6)$, $\mathcal{L}(s_5)=(6,7)$, $\mathcal{L}(s_6)=(7,2)$.

Definition 4: A component $\mathcal{Z}$ is called a *flow-tree* if $\mathcal{Z}$ does not contain any flow-loops.

Definition 5: Consider a component $\mathcal{Z}$; a link $l\in\mathcal{Z}$ is called a *starting link* if there exists a flow $s'\in\mathcal{S}(\mathcal{Z})$ such that $H^{s'}_{l,1}=1$ and $H^s_{l,k}=0$ for all other $s\in\mathcal{S}(\mathcal{Z})$ and all $k\ge 2$, i.e., a starting link has only exogenous arrivals. Similarly, a link $l\in\mathcal{Z}$ is called an *ending link* if there exists a flow $s''\in\mathcal{S}(\mathcal{Z})$ such that $H^{s''}_{l,|\mathcal{L}(s'')|}=1$ and $H^s_{l,k}=0$ for all other $s\in\mathcal{S}(\mathcal{Z})$ and all $k<|\mathcal{L}(s)|$, i.e., an ending link transmits only packets that will leave the system immediately. A path $P=(l_{P,1},l_{P,2},\ldots,l_{P,\mathrm{len}(P)})$, where $\mathrm{len}(P)$ denotes the length of path P and $l_{P,i}$ denotes

[4]By slightly abusing the notation, we also use $(s_1,s_2,\ldots,s_N)$ to denote the set of unique elements of the sequence.

the ith hop link of P, is called a *flow-path*, if the following conditions are satisfied.

1) Links $l_{P,1}$ and $l_{P,\text{len}(P)}$ are the only starting and ending link on the path P, respectively.
2) Either $\text{len}(P) = 1$, or for each $1 \leq i < \text{len}(P)$, there exists a flow s such that, $l_{P,i} \in \mathcal{L}(s)$ and $l_{P,i+1} \in \mathcal{L}(s)$, i.e., two adjacent links $l_{P,i}$ and $l_{P,i+1}$ are on the route of some flow.

In general, a flow-tree consists of multiple (possibly overlapped) flow-paths. An illustration of flow-loop, flow-path, and flow-tree is presented in Fig. 4. It is clear from Definition 3 that, if there exists a flow-loop in a component, this component must contain a cycle of links, while the opposite is not necessarily true. For example, the components in Fig. 4(b) and (c) both contain a cycle, while neither of them contains a flow-loop.

Now, we describe Algorithm 1, which is used to generate a ranking for a network without flow-loops such that the monotone property in Lemma 3 holds.

Algorithm 1: Rank Assignment

```
 1: procedure ASSIGNRANK($\mathcal{T}$)
 2:     $r(l) \leftarrow -1$ for all $l \in \mathcal{T}$
 3:     $\mathcal{P}' \leftarrow \mathcal{P}(\mathcal{T})$
 4:     while $\mathcal{P}' \neq \emptyset$ do
 5:         pick a flow-path $P \in \mathcal{P}'$
 6:         $\text{count} \leftarrow 1$
 7:         for $1 \leq i \leq \text{len}(P)$ do
 8:             if $r(l_{P,i}) = -1$ then
 9:                 $r(l_{P,i}) \leftarrow \text{count}$
10:             else if $r(l_{P,i}) \geq \text{count}$ then
11:                 $\text{count} \leftarrow r(l_{P,i})$
12:             else
13:                 for all $l \in \Gamma_k(l_{P,i})$ do
14:                     $r(l) \leftarrow r(l) + (\text{count} - r(l_{P,i}))$
15:                 end for
16:                 $r(l_{P,i}) \leftarrow \text{count}$
17:             end if
18:             $\text{count} \leftarrow \text{count} + 1$
19:         end for
20:         $\mathcal{P}' \leftarrow \mathcal{P}' \backslash \{P\}$
21:     end while
22: end procedure
```

Let $\mathcal{E}(P)$ denote the set of links belonging to flow-path P. Let $\mathcal{T}$ denote a flow-tree, and let $\mathcal{P}(\mathcal{T})$ denote the set of all flow-paths in $\mathcal{T}$, i.e., $\mathcal{P}(\mathcal{T}) \triangleq \{P \text{ is a flow path} \mid \mathcal{E}(P) \subseteq \mathcal{T}\}$. Let $P_k(\mathcal{T})$ denote the flow-path chosen in the kth while-loop when running Algorithm 1 for $\mathcal{T}$, and let $\mathcal{P}_k(\mathcal{T}) \triangleq \bigcup_{j<k} P_k(\mathcal{T})$. Let $r(l)$ denote the rank of link $l \in \mathcal{T}$, and let $\mathcal{P}(l)$ denote the set of flow-paths passing through link l, i.e., $\mathcal{P}(l) \triangleq \{P \in \mathcal{P}(\mathcal{T}) \mid l \in \mathcal{E}(P)\}$. Let $\Gamma_k(l) \triangleq \{l' \in \bigcup_{P \in \mathcal{P}(l) \bigcap \mathcal{P}_k(\mathcal{T})} \mathcal{E}(P) \mid r(l') > r(l)\}$ denote the set of links that belong to the flow-paths of $\mathcal{P}(l) \bigcap \mathcal{P}_k(\mathcal{T})$ (i.e., flow-paths that pass through link l and are chosen in the jth while-loop for $j < k$) and have a rank greater than $r(l)$.

The details of ranking are provided in Algorithm 1. In line 2, we do initialization by setting the rank of all links of $\mathcal{T}$ to -1. In lines 4–21, we pick a flow-path $P \in \mathcal{P}'$ and assign a rank to each link of P starting from link $l_{P,1}$. We may update a link's rank if we already assigned a rank to that link. The set of flow-paths $\mathcal{P}'$ is updated in line 20. The while-loop continues until $\mathcal{P}'$ becomes empty. We set $\text{count} = 1$ in line 6 and assign a rank to links $l_{P,i}$ for each $1 \leq i \leq \text{len}(P)$. For each link $l_{P,i}$, we consider the following three cases: 1) $r(l_{P,i}) = -1$; 2) $r(l_{P,i}) \geq \text{count}$; 3) $0 < r(l_{P,i}) < \text{count}$.

Case 1) Link $l_{P,i}$ has not been assigned a rank yet. We set $r(l_{P,i}) = \text{count}$ in line 9.

Case 2) Link $l_{P,i}$ already has a rank that is no smaller than the current count. In this case, the rank does not need an update, and we set $\text{count} = r(l_{P,i})$ in line 11.

Case 3) Link $l_{P,i}$ already has a rank that is smaller than the current count. In this case, we update the rank of some other links as well as that of link $l_{P,i}$. Specifically, for all the links $l \in \Gamma_k(l_{P,i})$, i.e., links that belong to the flow-paths in $\mathcal{P}(l) \bigcap \mathcal{P}_k(\mathcal{T})$ and have a rank greater than $r(l_{P,i})$, we increase their ranks by $\text{count} - r(l_{P,i})$ in lines 13–15. Then, we update the rank of link $l_{P,i}$ by setting it to count in line 16.

After considering all three cases, we increase the value of count by 1 in line 18.

TABLE I
EVOLUTION OF THE RANKING FOR THE FLOW-TREE IN FIG. 4(C)

Iteration k	Ranking of links $1-12$
0	(-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1)
1	(1,2,3,4,5,-1,-1,6,-1,7,-1,-1)
2	(1,2,3,4,5,3,-1,6,-1,7,4,5)
3	(1,2,3,4,5,3,1,6,-1,7,4,5)
4	(1,2,3,4,5,3,1,6,7,7,8,9)
5	(1,2,3,4,5,3,1,6,7,7,8,9)

The intention of this ranking is to assign a rank to each link such that the ranks are monotonically increasing when one traverses any flow-path from its starting link. Algorithm 1 may give different ranking to a given flow-tree depending on the order of choosing flow-paths. We give two examples for illustration as follows. In Fig. 4(b), one (and the unique one in this case) example of the ranking for the flow-tree is (1, 2, 3, 4, 5, 6, 7, 8) for links 1–8. In Fig. 4(c), one example of the ranking for the flow-tree is (1, 2, 3, 4, 5, 3, 1, 6, 7, 7, 8, 9) for links 1–12. The evolution of the ranking for the flow-tree in Fig. 4(c) is presented in Table I, where flow-path P_i is chosen in the ith while-loop, for $i = 1, 2, 3, 4, 5$.

Since we assume $\sum_s \sum_{k=1}^{|\mathcal{L}(s)|} H_{l,k}^s \geq 1$ for all $l \in \mathcal{E}$, a network graph $\mathcal{G}$ can be decomposed into multiple disjoint components. Clearly, a network with no flow-loops is equivalent to that all the components of the network are flow-trees. Without loss of generality, in the rest of the proof, we assume that the network we consider consists of one single component, which is a flow-tree under the condition of Lemma 3. The same argument applies to the case with multiple disjoint components. We claim the following lemma and provide its proof in our online technical report [17].

Lemma 17: Algorithm 1 assigns a rank to each link of a flow-tree $\mathcal{T}$ such that for any flow-path $P \in \mathcal{P}(\mathcal{T})$, the ranks are monotonically increasing when one traverses the links of P from $l_{P,1}$ to $l_{P,len(P)}$, i.e., $r(l_{P,i}) < r(l_{P,i+1})$ for all $1 \leq i < len(P)$ and for any $P \in \mathcal{P}(\mathcal{T})$.

Now, consider any flow $s \in \mathcal{S}$. The statement 1) holds trivially for the case of $|\mathcal{L}(s)| = 1$. Hence, we assume that $|\mathcal{L}(s)| > 1$. It is clear that for any $1 \leq i < |\mathcal{L}(s)|$, the links l_i^s and

l^s_{i+1} must belong to some flow-path $P \in \mathcal{P}(\mathcal{E})$, where $\mathcal{E}$ is assumed to be a flow-tree. Therefore, the statement 1) follows from Lemma 17.

Note that the packet arrivals at a link are either exogenous or from the previous hop on the route of some flow passing through it. Owing to the monotonically increasing rank assignment, it is clear that these previous hop links have a smaller rank. Hence, the statement 2) immediately follows from statement 1). This completes the proof of Lemma 3.

Acknowledgment

The authors would like to thank the associate editor and the reviewers for their valuable comments.

References

[1] 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.

[2] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proc. MobiCom*, 2006, pp. 227–238.

[3] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[4] 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," 2009 [Online]. Available: http://arxiv.org/PS_cache/arxiv/pdf/0901/0901.2333v4.pdf

[5] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: An efficient randomized protocol for contention resolution," in *Proc. ACM SIGMETRICS*, 2009, pp. 133–144.

[6] L. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1597–1609, Dec. 2011.

[7] A. Stolyar, "Large number of queues in tandem: Scaling properties under back-pressure algorithm," *Queueing Syst.*, vol. 67, no. 2, pp. 111–126, 2011.

[8] S. Liu, E. Ekici, and L. Ying, "Scheduling in multihop wireless networks without back-pressure," in *Proc. Allerton*, 2010, pp. 686–690.

[9] L. Ying, R. Srikant, and D. Towsley, "Cluster-based back-pressure routing algorithm," in *Proc. IEEE INFOCOM*, 2008, pp. 484–492.

[10] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over multihop wireless networks," in *Proc. IEEE INFOCOM*, 2009, pp. 1674–1682.

[11] X. Wu, R. Srikant, and J. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Trans. Mobile Comput.*, pp. 595–605, 2007.

[12] IETF, "Rfc 2453," 1998.

[13] IETF, "Rfc 791," 1981.

[14] Miniwatts Marketing Group, Bogota, Colombia, "Internet world stats," 2011 [Online]. Available: http://www.internetworldstats.com/stats.htm

[15] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting*, Scheduling in a Queuing System With Asynchronously Varying Service Rates*. Cambridge, U.K.: Cambridge Univ. Press, 2004, vol. 18.

[16] M. Bramson, "Stability of queueing networks," *Probab. Surveys*, vol. 5, no. 1, pp. 169–345, 2008.

[17] B. Ji, C. Joo, and N. B. Shroff, "Throughput-optimal scheduling in multi-hop wireless networks without per-flow information," 2012 [Online]. Available: http://arxiv.org/abs/1101.4211

[18] J. Dai, "On positive harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *Ann. Appl. Probab.*, pp. 49–77, 1995.

[19] A. Rybko and A. Stolyar, "Ergodicity of stochastic processes describing the operation of open queueing networks," *Problems Inf. Transmission*, vol. 28, pp. 199–220, 1992.

[20] S. Lu and P. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. Autom. Control*, vol. 36, no. 12, pp. 1406–1416, Dec. 1991.

[21] H. Chen and H. Zhang, "Stability of multiclass queueing networks under priority service disciplines," *Oper. Res.*, vol. 48, no. 1, pp. 26–37, 2000.

[22] H. Chen and H. Ye, "Piecewise linear Lyapunov function for the stability of multiclass priority fluid networks," *IEEE Trans. Autom. Control*, vol. 47, no. 4, pp. 564–575, Apr. 2002.

[23] H. Chen and D. Yao, "Stable priority disciplines for multiclass networks," in *Stochastic Networks: Stability and Rare Events*, P. Glasserman, K. Sigman, and D. Yao, Eds. New York: Springer-Verlag, 1996, ch. 2, pp. 27–39.

[24] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multi-hop wireless networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2579–2587.

[25] V. Malyshev and M. Menshikov, "Ergodicity, continuity and analyticity of countable Markov chains," *Trans. Moscow Math. Soc.*, vol. 39, pp. 3–48, 1979.

[26] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, 2000, pp. 556–564.

Bo Ji (S'11) received the B.E. and M.E. degrees in information science and electronic engineering from Zhejiang University, Hangzhou, China, in 2004 and 2006, respectively, and is currently pursuing the Ph.D. degree in electrical and computer engineering at The Ohio State University, Columbus.

His research interests include resource allocation, optimization, cross-layer network control, and low-complexity scheduling algorithm design.

Changhee Joo (M'09) received the Ph.D. degree in electrical and computer engineering from Seoul National University, Seoul, Korea, in 2005.

He worked at the Center of Wireless Systems and Applications, Purdue University, West Lafayette, IN, and at The Ohio State University, Columbus. In 2010, he joined the Korea University of Technology and Education, Cheonan, Korea, as a faculty member. He currently works at the Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea. His research interests include communication network systems, network performance optimization, wireless sensor networks, and network controls.

Dr. Joo is a recipient of the IEEE INFOCOM 2008 Best Paper Award.

Ness B. Shroff (S'91–M'93–SM'01–F'07) received the Ph.D. degree in electrical engineering from Columbia University, New York, NY, in 1994.

He joined Purdue University, West Lafayette, IN, immediately thereafter as an Assistant Professor. At Purdue, he became Professor of the School of Electrical and Computer Engineering (ECE) in 2003, and Director of the Center for Wireless Systems and Applications (CWSA) in 2004. In July 2007, he joined the ECE and Computer Science and Engineering (CSE) departments at The Ohio State University, where he holds the Ohio Eminent Scholar Chaired Professorship of Networking and Communications. His research interests are in fundamental problems in communication, sensing, social, and cyberphysical networks.

Dr. Shroff is an NSF CAREER awardee, and a number of his conference and journals papers have received best paper awards.