A Low-Complexity Congestion Control and Scheduling Algorithm for Multihop Wireless Networks with Order-Optimal Per-Flow Delay

Po-Kai Huang, Xiaojun Lin, and Chih-Chun Wang
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN, USA
Email: {huang113, linx, chihw}@purdue.edu

***Abstract*—We consider the problem of designing a joint congestion control and scheduling algorithm for multihop wireless networks. The goal is to maximize the total utility and achieve low end-to-end delay simultaneously. Assume that there are m flows inside the network, and each flow has a fixed route with H_m hops. Further, the network operates under the one-hop interference constraint. We develop a new congestion control and scheduling algorithm that combines a window-based flow control algorithm and a new distributed rate-based scheduling algorithm. For any $\epsilon, \epsilon_m \in (0,1)$, by appropriately choosing the number of backoff mini-slots for the scheduling algorithm and the window-size of flow m, our proposed algorithm can guarantee that each flow m achieves throughput no smaller than $r_m(1-\epsilon)(1-\epsilon_m)$, where the total utility of the rate allocation vector $\vec{r} = [r_m]$ is no smaller than the total utility of any rate vector within half of the capacity region. Furthermore, the end-to-end delay of flow m can be upper bounded by $H_m/(r_m(1-\epsilon)\epsilon_m)$. Since a flow-$m$ packet requires at least H_m time slots to reach the destination, the order of the per-flow delay upper bound is optimal with respect to the number of hops. To the best of our knowledge, this is the first fully-distributed joint congestion-control and scheduling algorithm that can guarantee order-optimal per-flow end-to-end delay and utilize close-to-half of the system capacity under the one-hop interference constraint. The throughput and delay bounds are proved by a novel stochastic dominance approach, which could be of independent value. Our algorithm can be easily implemented in practice with a low per-node complexity that does not increase with the network size.**

I. Introduction

The joint congestion control and scheduling problem in multihop wireless networks has been extensively studied in the literature [1], [2]. Often, each user is associated with a non-decreasing and concave utility function of its rate, and a cross-layer utility maximization problem is formulated to maximize the total system utility subject to the constraint that the rate vector can be supported by some scheduling algorithm. One optimal solution to this problem is known to be the max weight back-pressure scheduling algorithm combined with a congestion control component at the source [1], [2]. Further, significant progresses have been made in designing distributed scheduling algorithms with provable throughput and lower complexity than the back-pressure algorithm [3]–[10]. However, most of the existing works on joint congestion control and scheduling have only considered the throughput performance metric and not accounted for delay performance issues. Although for flows with congestion control (e.g., file transfer) the throughput is often the most critical performance metric, packet delay is very important as well because practical congestion control protocols need to set retransmission time-out values based on the packet delay, and such parameters could significantly impact the speed of recovery when packet loss occurs. Packet delay is also important for multimedia traffic, some of which have been carried on congestion-controlled sessions.

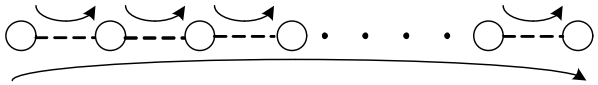


Fig. 1. A wireless network with linear topology.

There are two major issues on the delay-performance of the back-pressure algorithm. Firstly, for long flows, the end-to-end delay may grow quadratically with the number of hops. The reason can be best explained by the following example [11]. Consider a long flow traverses a fixed route with H hops. For each link that the long flow traverses, there is a short flow that competes with the long flow as shown in Fig. 1. Under the back-pressure algorithm, for the long flow to be scheduled on a link, the queue difference of the long flow must be larger than the queue length q of the competing short flow. Therefore, when the joint congestion and scheduling algorithm converges, the queue length of the long flow at each hop must be around $Hq, (H-1)q, \cdots, q$, which leads to the total end-to-end backlog of order $\mathcal{O}(H^2)$. It then follows from Little's law that the end-to-end delay will also be of the order $\mathcal{O}(H^2)$. Secondly, under the back-pressure algorithm it is difficult to control the end-to-end delay of each flow. The main parameter to tune a joint congestion control and scheduling algorithm based on the back-pressure algorithm is the step size in the queue update. A larger step size may lead to smaller queue length; however, a smaller step size is needed to ensure that the joint congestion control and scheduling algorithm converges to close-to-optimal system throughput. Although one may use the step sizes to tune the throughput-delay tradeoff, a change of the step size on one node will likely affect all flows passing through the node. Hence, it is difficult to tune the throughput-delay tradeoff on a per-flow basis.

In this paper, we will provide a new class of joint congestion control and scheduling algorithms[1] that can achieve both provable throughput and provable per-flow delay. Consider m flows in a multihop network under the one-hop interference model, and each flow m is given a fixed route with H_m hops. Our algorithm consists of three main components: window-based flow control, virtual-rate computation, and scheduling. The main ideas are to tightly control the number of packets inside the network and to schedule the packets by a rate-based scheduling algorithm rather than a queue-length-based algorithm. The key difficulty in analyzing the end-to-end throughput and delay for window-based flow control is that *the services at different links are correlated*. Hence, a Markov chain analysis will no longer provide a closed-form solution. We employ a novel stochastic dominance technique to circumvent this difficulty and derive closed-form bounds on the per-flow throughput and delay. Specifically, for any $\epsilon, \epsilon_m \in (0,1)$, by appropriately choosing the number of backoff mini-slots for the scheduling algorithm and the window size of flow m, our algorithm can guarantee that each flow m will achieve a throughput no less than $r_m(1-\epsilon)(1-\epsilon_m)$, where the total utility of the rate allocation vector $\vec{r} = [r_m]$ is no smaller than the total utility of any rate vector within half of the system capacity region. Further, the end-to-end expected delay of flow m can be upper bounded by $H_m/(r_m(1-\epsilon)\epsilon_m)$. Simulation results show that when $H_m \geq 2$, with a reasonable choice of $\epsilon = 0.22$ and $\epsilon_m = 0.33$ the actual throughput is typically larger than r_m. Therefore, our scheme can utilize a provable fraction of the total system utility with per-flow expected delay that increases linearly with the number of hops. Since a flow-m packet requires at least H_m time slots to reach the destination, the order of the per-flow delay upper-bound is optimal with respect to the number of hops. Our proposed algorithm is fully-distributed and can be easily implemented in practice. Further, the delay-throughput tradeoff of each flow can be individually controlled. To the best of our knowledge, this is the first fully-distributed cross-layer control solution that can both guarantee order-optimal per-flow delay and utilize close-to-half of the system capacity under the one-hop interference constraint.

Recently, there have been a number of papers that quantify the delay performance of wireless networks with or without congestion control [11], [13]–[22]. In [11], [13], the authors propose methods to reduce the delay of the back-pressure algorithm. The algorithm proposed in [11] is a shadow back-pressure algorithm, which maintains a single FIFO queue at each link and uses multiple shadow queues to schedule the transmissions. This method decouples the control information from the real queues and hence reduces the delay. In our simulation, this algorithm seems to achieve linear delay after the algorithm converges. However, at the transient period, the real queues will still follow the shadow queues, which leads to a large queue backlog (see Fig. 4(c) in this paper and figures in [11]). In [13], the authors propose another mechanism to decouple the control signal from the real queues by injecting dummy packets into the queues. However, until the algorithm converges, it will be difficult to know the correct number of dummy packets. Hence, the backlog at the transient period is still difficult to control. In contrast, our proposed algorithm tightly controls the end-to-end backlog at all times.

Our result is also different from other works in providing a linear-order per-flow delay bound. First, [14], [15] only prove delay bounds for single-hop flows rather than multihop flows. Second, [16]–[18] consider the delay among all the flows rather than the per-flow delay. Similarly, the results in [19] can be used to construct a bound on the delay averaged over all flows. However, it is still not a per-flow delay bound. Third, a per-flow delay bound is provided in [20], but the bound scales with the size of the network. Fourth, a single flow end-to-end delay analysis is given in [21] based on an approximation of the departure process for each hop. However, it is unclear how to extend the analysis to multiple flows.

Our result is perhaps most comparable to that in [22], where the authors provide a per-flow delay bound that scales with the number of hops without considering congestion control. However, the algorithm in [22] has a factor 5 loss of throughput under the one-hop interference constraint, and the algorithm is much more complicated, e.g., the per-node complexity is $\mathcal{O}(N)$, where N is the number of nodes. In contrast, our algorithm only requires $\mathcal{O}(1)$ complexity per-node and can utilitze close-to-half of the capacity.

Our contributions can be summarized as follows.

- We provide a new joint congestion control and scheduling algorithm that can utilize close to half of the system capacity, i.e., a factor 2 loss of throughput, and guarantee a per-flow expected delay upper bound that increases linearly with the number of hops.
- The congestion control algorithm is based on window flow control. For each flow, this method deterministically bounds the end-to-end backlog within the network and prevents buffer overflows. Further, each flow's throughput-delay tradeoff can be individually controlled.
- Our algorithm is fully distributed and can be easily implemented in practice with a low per-node complexity that does not increase with the network size.
- We use a novel stochastic dominance method to analyze the end-to-end delay. This method is new and can be applied to general interference constraints.

The remainder of this paper is organized as follows. The system model is presented in Section II. In Section III, we propose the joint congestion control and scheduling algorithm and present the main analytical results on per-flow throughput and delay. Section IV is dedicated to the proof of a key proposition by a novel stochastic dominance method. Implementation issues are discussed in Section V, and simulation results are reported in Section VI. Then we conclude.

[1]A related delay bound can also be shown for the scheduling algorithm without congestion control, which will appear as a one-page abstract in [12]. However, the delay for joint congestion control and scheduling in this paper is more difficult due to the closed-loop feedback.

II. SYSTEM MODEL

We model a wireless network by a graph $G = (V, E)$, where V is the set of nodes, and E is the set of links. Each link $\ell \in E$ consists of a transmitter node $b(\ell)$ and a receiver node $d(\ell)$. Two nodes are *one-hop* neighbors if they are the end-points of a common link. Two links are one-hop neighbors if they share a common node. For each node v, let $N(v)$ denote the set of links that connect to the one-hop neighbors of node v.

We assume a time-slotted wireless system, where packet transmissions occur within time slots of unit length. The capacity of any link is normalized to 1. We say two links interfere with each other, if they can not transmit data at the same time slot. For ease of exposition, we assume that a link will interfere with all its one-hop neighboring links (i.e., one-hop interference constraint). This constraint has been used to model wireless networks in [5], [6], [23], [24]. (We note that our approach can be extended to a more general interference model, where an interference set for each link is given, and a link interferes with any other link in its interference set.) In our system, there are M flows, and each flow is associated with a source node, a destination node, and a fixed route between them. The routes are given by the matrix $[L_m^\ell]$, where $L_m^\ell = 1$ if flow m passes through link ℓ, and $L_m^\ell = 0$ otherwise. We assume that each flow always has packets to transmit. The congestion control algorithm will then determine the rate with which packets are injected into the network [1]. Each flow is associated with a utility function $U_m(R_m)$ [25], which reflects the "satisfactory level" of user m with injection rate R_m. We assume that $U_m(\cdot)$ is strictly concave, non-decreasing, and continuously differentiable. The capacity region Ω of a wireless network is the set of all rate vectors $\vec{R} = [R_m]$ such that there exists a network control policy to stabilize the network. We then model the joint congestion control and scheduling problem as:

$$\max_{R_m \geq 0} \sum_m U_m(R_m), \ \vec{R} \in \Omega. \tag{1}$$

The exact capacity region Ω is often difficult to characterize. It is also well known that, under the one-hop interference model, $\Psi_0/2 \subseteq \Omega \subseteq \Psi_0$, where

$$\Psi_0 = \left\{ \vec{R} \left| \sum_{\ell \in N(v)} \sum_m L_m^\ell R_m \leq 1, \text{ for all nodes } v \right. \right\}. \tag{2}$$

Note that equation (2) simply states that, for each node v, the total load must be no larger than 1. This is because a wireless node can only communicate with one other node at a time slot. In Section III, we will describe how we utilize the relationship between Ω and Ψ_0 to approximately solve problem (1). Since we assume infinite backlog, the delay of a packet is computed from the time it is injected to the network to the time it reaches the destination. We are interested in the per-flow average delay.

III. JOINT CONGESTION CONTROL AND SCHEDULING ALGORITHM

As we discussed in Section I, there are many approaches available in the literature to solve problem (1), and most of them do not consider delay performance. A typical optimal solution can be obtained by a duality approach which results into the back-pressure algorithm and a congestion-control component at the source node [1], [2]. Further, a considerable amount of effort has focused on developing low-complexity and distributed scheduling algorithms that can replace the centralized back-pressure algorithm and yet still achieve provable good throughput performance [3]–[10]. Like the back-pressure algorithm, these low-complexity scheduling algorithms are usually also queue-length-based. The drawback of these approaches, however, is that the end-to-end delay of the resulting queue-length-based scheduling algorithm is very difficult to quantify, and there are evidence that, under certain cases, the back-pressure can have poor delay performance [11], [26]. In this paper, we will use a window-based flow control algorithm and a rate-based scheduling algorithm that are very different from back-pressure. Our solution strategy is to first approximately solve problem (1) and compute the decision vector $\vec{r} = [r_m]$. However, the decision variables r_m are NOT directly used as the rates to inject flow-m packets. For this reason, we refer to these variables r_m as "virtual rates". We will use these virtual rates as the control variables in a new class of rate-based scheduling algorithms. The actual end-to-end throughput under our algorithm will be denoted as R_m. As readers will see, for each flow, this new joint congestion control and scheduling algorithm will guarantee both provable throughput (close to r_m) and provably-low delay. Also, they are fully distributed and easy-to-implement in real systems.

A. Virtual-Rate Computation

We first briefly describe how to approximately solve problem (1). Since the true capacity region Ω is of a complex form, instead of solving problem (1) directly, we solve the following optimization problem: (we will make precise the relationship between optimization problems (1) and (3) in Section III-D.)

$$\max_{r_m \geq 0} \sum_m U_m(r_m), \ \vec{r} \in \Psi_0/2. \tag{3}$$

Note that the optimization problem (3) is very similar to the standard convex-optimization problem in wireline network with linear constraints [27], [28]. Therefore, it is easy to apply the approaches in [27], [28] to problem (3). We will not elaborate on all the possible approaches to solve problem (3). Instead, we only present one well-known distributed solution. Specifically, associate a Lagrange multiplier (the dual variable) $\lambda_v \geq 0$ to each constraint in (3). Let $c_{vm} = 2$ if v is an intermediate node of flow m, $c_{vm} = 1$ if v is the source node or destination node of flow m, and $c_{vm} = 0$ otherwise. The objective function of the dual problem of (3) becomes:

$$D(\vec{\lambda}) := \max_{r_m \geq 0} \sum_m U_m(r_m) - \sum_v \lambda_v \left(\sum_m r_m c_{vm} - \frac{1}{2} \right).$$

We can then use the following gradient algorithm to minimize $D(\vec{\lambda})$ and compute the optimal virtual-rates.

Virtual-Rate Computation Algorithm: At each time t,

1) The source node of flow m updates r_m by equation:

$$r_m(t) = U_m'^{-1}(\sum_{v \in m} \lambda_v(t) c_{vm}),$$

where $v \in m$ indicates that node v is on flow m's route.

2) Each node updates the dual variables by equation:

$$\lambda_v(t+1) = \left[\lambda_v(t) + \gamma_v \left(\sum_{m: v \in m} r_m(t) c_{vm} - \frac{1}{2}\right)\right]^+,$$

where $\gamma_v > 0$ is the step size, and $[\cdot]^+$ denotes the projection to $[0, \infty)$.

Using similar techniques as [27], one can show that as long as γ_v are sufficiently small, the above algorithm will converge to the optimal solution of (3). Note that as in [27], this algorithm requires passing λ_v and r_m among nodes in the network. We will give a simple protocol to exchange such information in Section V. As we emphasized earlier, the variables r_m are "virtual rates", and they are not directly used to inject flow-m packets under our proposed algorithm. We choose not to directly use the virtual rates as the real injection rates due to the following reasons. First, optimization problems (1) and (3) are formulated as if the rates are immediately passed to all links at the same time. In reality, a packet must traverse the links in a hop-by-hop fashion, and a flow-control algorithm is needed to regulate this hop-by-hop packet flow. Second, the low-complexity virtual-rate computation algorithm did not produce the schedule for link transmission. We still need a scheduling algorithm to compute the schedule that can support the virtual rate vector $\vec{r} = [r_m]$.

Readers who are familiar with the literature will realize that the back-pressure algorithm can again be used to answer the above flow control and scheduling questions. However, we would then return to our starting point that the end-to-end delay of back-pressure is difficult to quantify and may be poor [11], [26]. Hence, in the sequel, we will use very different scheduling and flow-control components, for which we can quantify both the throughput and the end-to-end delay on a per-flow basis.

B. Scheduling Algorithm

We now present the scheduling algorithm, which is a modification of the low-complexity distributed scheduling algorithm in [10]. Each time slot consists of a scheduling slot and a transmission slot. The links that are to be scheduled are selected in the scheduling slot, and the selected links transmit their packets in the transmission slot. The scheduling slot is further divided into F mini-slots. Let $a_\ell(t) = \sum_m L_m^\ell r_m(t)$, which is the sum of the virtual rate over link ℓ, and let $x_\ell(t) = \max\left(\sum_{e \in N(b(\ell))} a_e(t), \sum_{e \in N(d(\ell))} a_e(t)\right)$.

Rate-based Scheduling Algorithm: At each time t,

1) Each link ℓ first computes $P_\ell = \log(F) a_\ell(t)/x_\ell(t)$.
2) Each link then randomly picks the number of backoff mini-slots (B) with distribution: $P\{B = F+1\} = e^{-P_\ell}$ and $P\{B = f\} = e^{-P_\ell \frac{f-1}{F}} - e^{-P_\ell \frac{f}{F}}, f = 1, \cdots, F$. If $F+1$ is picked by link ℓ, it will not attempt to transmit in this time slot.
3) When the backoff timer for a link expires, it begins transmission unless it has already heard a transmission from one of its interfering links. If two or more links that interfere begin transmissions simultaneously, a collision occurs, and both transmissions fail.
4) If a link begins transmission, it will randomly choose a passing flow m to serve with probability $r_m(t)/a_\ell(t)$.

Note that this scheduling algorithm only uses virtual rates to compute P_ℓ, which is different from the queue-length-based algorithm studied in [10]. For simplicity, our performance analysis will be based on this scheduling algorithm. On the other hand, note that this algorithm can be easily improved by letting each link attempt only if it has packets to transmit, and if it starts transmission, it will randomly serve a flow m with positive backlog (i.e., $Q_{m\ell}(t) > 0$) with probability $\frac{r_m(t)}{\sum_{\{m: Q_{m\ell}(t)>0\}} L_m^\ell r_m(t)}$, where $Q_{m\ell}(t)$ is the number of flow-m packets at link ℓ at time t. It is easy to see that this improved version will lead to higher throughput. Hence, the bound we derive in Section IV also holds for this improved version.

C. Window-based Flow Control Algorithm

Now, we describe the congestion control component. Our approach is to use window-based flow control. For each flow, we maintain a window W_m at the source node, and we only inject new packets to the queue at the source node when the total number of packets for this flow inside the network is smaller than the window size. This can be achieved by letting the destination node send an acknowledgement (ACK) back to the source node whenever it receives a packet. There are two advantages for this approach. First, for each flow, we can tightly control the maximum number of packets in each intermediate node along the route. This will prevent buffer overflows, which is an important issue as addressed in [19]. Second, as we will show in Section IV, each flow's tradeoff between throughput and delay can be individually controlled by the window size. Note that when we present the analysis in Section IV, we assume that there is a feedback channel from the destination node to the source node at each time slot. Through this feedback channel, the destination node can send the ACK to the source node, and the source node can then decide if it is possible to inject another packet at the next time slot. In reality, each ACK will also go through the entire route hop-by-hop in the reverse direction to reach the source node. In Section V, we will discuss how this can be achieved by piggy-backing the ACK after each packet transmission. As readers will see, this method can be analyzed with the same approach presented in Section IV, and this extra ACK delay does not change the delay order of our result.

D. Performance Analysis

In this subsection, we will present the main steps of the analysis and the bounds on the throughput and delay of

the above proposed scheme. We first present a relationship between optimization problems (1) and (3). Let $[r_m^*]$ be the optimal solution of (3), and let $[r_m'^*]$ be the optimal solution of the following optimization problem:

$$\max_{r_m \geq 0} \sum_m U_m(r_m),\ \vec{r} \in \Omega/2. \tag{4}$$

Lemma 1: $\sum_m U_m(r_m^*) \geq \sum_m U_m(r_m'^*)$.

Proof: Since r_m^* is the optimal solution of (3), the total utility of r_m^* will be larger than any rate vector in $\Psi_0/2$. Also, by $\Omega \subseteq \Psi_0$, we have that $\Omega/2 \subseteq \Psi_0/2$. As a result, $r_m'^*$ is a rate vector in $\Psi_0/2$, and $\sum_m U_m(r_m^*) \geq \sum_m U_m(r_m'^*)$. ■

In other words, if each flow achieves a throughput equal to the optimal virtual rate r_m^*, then the total system utility will be no less than the maximum utility within $\Omega/2$. Further, we can show the following property of the scheduling algorithm.

Lemma 2: For any $\epsilon \in (0,1)$, if flow m passes through link ℓ, we can choose F large enough such that the probability that link ℓ will schedule flow m at time t is no smaller than $r_m(t)(1-\epsilon)/(2x_\ell(t))$.

Proof: The proof of this lemma is similar to the proof of Lemma 1 in [10]. Let $\mathcal{E}_\ell$ be the union of the interference set of link ℓ. Also, in the following proof, we drop the index t from the notation $P_\ell(t)$ when there is no cause of confusion. Let E_ℓ be the event that link ℓ is scheduled. By equation (17) in [10], we have that

$$P(E_\ell) \geq (e^{\frac{P_\ell}{F}} - 1) \sum_{f=1}^{F} e^{-\frac{f}{F}\sum_{h\in\mathcal{E}_\ell} P_h}. \tag{5}$$

We first find an upper bound for the term $\sum_{h\in\mathcal{E}_\ell} P_h$ in equation (5). Let $\alpha = \log(F)$. At time t, we have that

$$\begin{aligned}
&\sum_{h\in\mathcal{E}_\ell} P_h = \alpha \sum_{h\in\mathcal{E}_\ell} \frac{a_h(t)}{\max\left(\sum_{e\in N(b(h))} a_e(t), \sum_{e\in N(d(h))} a_e(t)\right)} \\
&\leq \alpha \left(\sum_{h\in N(b(\ell))} \frac{a_h(t)}{\sum_{e\in N(b(\ell))} a_e(t)} + \sum_{h\in N(d(\ell))} \frac{a_h(t)}{\sum_{e\in N(d(\ell))} a_e(t)} \right) \\
&\leq 2\alpha.
\end{aligned} \tag{6}$$

From (5) and (6), we know that

$$\begin{aligned}
P(E_\ell) &\geq (e^{\frac{P_\ell}{F}} - 1) \sum_{f=1}^{F} e^{-2\alpha\frac{f}{F}} \geq \frac{P_\ell}{F} \sum_{f=1}^{F} e^{-2\alpha\frac{f}{F}} \\
&= \frac{P_\ell}{F} \frac{1-e^{-2\alpha}}{1-e^{-\frac{2\alpha}{F}}} e^{-\frac{2\alpha}{F}}
\end{aligned}$$

Since $\alpha = \log(F)$ and $F > 1$, we can see that $\frac{\alpha}{F}/(1-e^{-2\alpha/F}) \geq 1/2$, $e^{-2\alpha/F} \geq 1 - 2\log(F)/F$, and $1 - e^{-2\alpha} = 1 - 1/F^2$. Hence,

$$\begin{aligned}
P(E_\ell) &\geq \frac{a_\ell(t)}{2x_\ell(t)}\left(1 - \frac{2F\log(F)+1}{F^2}\right) \\
&> \frac{a_\ell(t)}{2x_\ell(t)}(1-\epsilon),
\end{aligned} \tag{7}$$

where F is chosen such that $\epsilon > \frac{2F\log(F)+1}{F^2}$. We can then conclude that

$$\begin{aligned}
&P(\text{link } \ell \text{ is scheduled to transmit a flow } m \text{ packet}) \\
=\ &P(\text{flow } m \text{ is scheduled}|E_\ell)P(E_\ell) \\
\geq\ &\frac{r_m(t)}{a_\ell(t)}\frac{a_\ell(t)}{2x_\ell(t)}(1-\epsilon) \\
=\ &\frac{r_m(t)}{2x_\ell(t)}(1-\epsilon).
\end{aligned}$$

■

This lemma implies that the scheduling algorithm will start serving flow-m packets even before the virtual-rate computation algorithm converges. After the virtual-rate computation algorithm converges[2], the value of $r_m(t)$ will be equal to the optimal solution r_m^*. Furthermore, we have from the constraints of optimization problem (3) that $x_\ell(t) \leq 1/2$. It then follows from Lemma 2 that every hop along the path of flow m will serve flow m with probability no smaller than $r_m^*(1-\epsilon)$ independent across time slots. This observation allows us to isolate flow m out of the network and view the flow-m packets as passing through a virtual tandem network of H_m queues. Intuitively, if the window size of flow m is large, the end-to-end throughput of flow m, i.e., R_m, will be at least $r_m^*(1-\epsilon)$; however, the end-to-end packet delay will also be large. If we reduce the window size, although the delay will decrease, the throughput of flow m will also decrease. Clearly, the key is then to analyze the throughput and delay as a function of the window size. The following proposition, which will be proved in Section IV, is the key result of the paper. Note that this analysis is difficult because Lemma 2 only provides a lower-bounded marginal probability for services. Further, the exact statistics of the correlation among links is hard to characterize because of the interference constraint.

Proposition 3: After the virtual-rate computation algorithm converges, for each flow m, our congestion control and scheduling algorithm can achieve average throughput $\frac{r_m^*(1-\epsilon)W_m}{W_m+H_m-1}$, where W_m is the window size of flow m, and ϵ is chosen as in Lemma 2. Moreover, the average delay is upper bounded by $\frac{W_m+H_m-1}{r_m^*(1-\epsilon)}$.

This proposition has the following two implications. First, for any $\epsilon_m \in (0,1)$, let W_m be the smallest positive integer such that $W_m > (H_m-1)(1-\epsilon_m)/\epsilon_m$. This implies that $W_m/(W_m+H_m-1) > (1-\epsilon_m)$. It then follows from Proposition 3 that the average throughput R_m will be lower bounded by $r_m^*(1-\epsilon)(1-\epsilon_m)$, which can be arbitrary close to r_m^*. Note that by Lemma 1, the total utility of the rate vector $\vec{r}* = [r_m^*]$ is no smaller than the total utility of any rate vector within half of the capacity region. Second, since W_m is the smallest positive integer such that $W_m > (H_m-1)(1-\epsilon_m)/\epsilon_m$, we have that $W_m \leq (H_m-1)(1-\epsilon_m)/\epsilon_m + 1$. Thus,

$$\frac{W_m+H_m-1}{r_m^*(1-\epsilon)} \leq \frac{H_m+\epsilon_m-1}{r_m^*(1-\epsilon)\epsilon_m} < \frac{H_m}{r_m^*(1-\epsilon)\epsilon_m}.$$

[2]We note that a comparable bound on the probability of scheduling flow m on link ℓ can also be obtained by assuming that $r_m(t)$ is within some small neighborhood fo r_m^*. For ease of exposition, we do not pursued this direction further in this paper.

It then follows from Proposition 3 that the delay will be upper bounded by $H_m/(r_m^*(1-\epsilon)\epsilon_m)$. As discussed in Section I, this implies that our per-flow delay upper-bound is order optimal.

IV. Proof of Proposition 3

Assume that the virtual-rate computation algorithm has converged at time t. Thus, $r_m(t) = r_m^*$ for the following time slots. This implies that, for a particular flow, its service at every hop is identically and independently distributed (i.i.d.) across time. Furthermore, for ease of presentation, we assume that there is a feedback channel from the destination node to the source node for the window flow control at each time slot as discussed in Section III-C. (This assumption will be removed in Section V.) Now, we focus on a particular flow m. The analysis for other flows is the same. To ease the notation, we drop the index m from the notations W_m and H_m. We can then model this flow as a H-hop closed tandem network. Label the link along the route from 1 to H, where 1 is the link closest to the source node. By the discussion after Lemma 2, we know that

$$\mu_\ell \geq \mu \triangleq r_m^*(1-\epsilon), \tag{8}$$

where μ_ℓ is the probability that link ℓ will serve a flow-m packet. Since we use window flow control, and flow m always has packets to transmit, the number of flow-m packets in the network will be W at each time slot. We can thus use the discrete time Markov Chain (MC) analysis to study the closed tandem network for flow m. Specifically, let $\vec{Q}(t) = (Q_1(t), \cdots, Q_H(t))$ be the system state, where $Q_i(t)$ is the number of flow-m packets at i^{th} hop at the beginning of time t. Furthermore, let $\vec{S}(t) = (S_1(t), \cdots, S_H(t))$ be the random schedule vector for flow m at time t, where $S_i(t) = \mathbf{1}_{\{\text{link } i \text{ is scheduled at time } t\}}$. Since $\vec{S}(t)$ is i.i.d. across time slots, the state at time $t+1$ will only depend on the current state $\vec{Q}(t)$ and the schedule $\vec{S}(t)$. It can be verified that this MC is ergodic.

Lemma 4: The MC induced by algorithm 2 is ergodic, i.e., irreducible, positive recurrent, and aperiodic.

Proof: It is easy to see that the Markov Chain is irreducible because, for each link ℓ that flow m passes by, there is a nonzero probability that only this link will serve flow m's packets. As a result, all states can communicate with each other. Since the Markov Chain only has a finite number of states, by Theorem 3.3 in [29], it is positive recurrent. Consider the state that the number of packets in the first queue is equal to W. Because there is a nonzero probability that only link 2 will be scheduled, there is a nonzero probability for this state to go back to itself in one step. Hence, this state is aperiodic. By Proposition 4.2.2 in [30], the Markov Chain is aperiodic. ■

Fig. 2 illustrates an example of the MC for a 3-hop closed tandem network with 5 packets. If the MC is in steady state, we can compute the actual throughput R_m as follows:

$$R_m = \mu_H[1 - P\{Q_H = 0\}]. \tag{9}$$

If we can compute the throughput, then the delay can be obtained by Little's law. Unfortunately, it appears impossible to directly solve the MC. The reason is because the services of different links are correlated. For example, link 1 and link 2 will never be scheduled together due to interference, and there is a chance that link 1 and link 3 will be served together. Further, the exact statistics of such correlation is hard to characterize. All that we know (from Lemma 2) is a lower-bounded marginal probability μ that link ℓ is activated to send a flow m packet. Hence, it is difficult to solve the MC directly. To circumvent this difficulty, we next use a novel stochastic dominance approach to derive the throughput lower-bound.

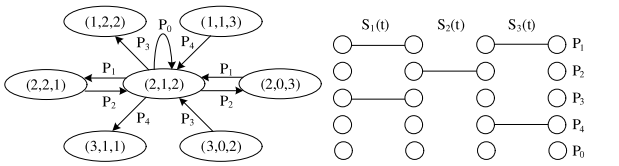


Fig. 2. Left: The incoming and outgoing transitions to and from state $(2,1,2)$. Right: The distribution of the random schedule vector $\vec{S}(t)$.

A. Overview of the Approach

We start with some definitions and assumptions. In the rest of this section, when we refer to a particular system, we mean a version of the H-hop closed tandem network with window flow control and window size W. For each system, the random schedule vector is always i.i.d. across time. Further, for different systems, the initial condition for the packet placement is the same. However, within a time slot, the distribution of the schedule vector is different depending on the system that we refer to. Since the only difference between two systems is the distribution of the schedule, we abuse the notation and denote a system by $\vec{S}(\cdot)$ when the corresponding random schedule vector is denoted by $\vec{S}(t)$. Furthermore, we denote $T(\vec{S}(\cdot))$ and $D(\vec{S}(\cdot))$ as the throughput and delay of system $\vec{S}(\cdot)$.

Consider a system $\vec{S}^{(1)}(\cdot)$. Let the probability distribution of $\vec{S}^{(1)}(t)$ be $P\{\vec{S}^{(1)}(t) = \vec{x}_i\} = p_i'$, $i = 0, \cdots, I$, where $\vec{x}_i = (x_{i1}, \cdots, x_{iH})$ is the i^{th} schedule vector, $x_{ij} = 1$ if link j is activated under the i^{th} schedule vector, and $x_{ij} = 0$ otherwise. We use the convention that $\vec{x}_0 = \vec{0}$. Let $A_\ell = \{i | x_{i\ell} = 1\}$. Notice that $\mu_\ell' \triangleq \sum_{i \in A_\ell} p_i'$ is the marginal probability that link ℓ is scheduled at one time slot. Assume that system $\vec{S}^{(1)}(\cdot)$ satisfies the following property. (We will discuss in Section IV-C how to treat the case when property (10) is not satisfied.)

$$\sum_\ell \mu_\ell' = \sum_\ell \sum_{i \in A_\ell} p_i' \leq 1, \text{and } \mu_\ell' \geq \mu'. \tag{10}$$

Recall that the key difficulty of analyzing the system is the correlation of the services among links. We now introduce a splitting procedure that convert a given system to another system where links are less likely to be scheduled together.

Construct system $\vec{S}^{(2)}(\cdot)$ as follows. First, pick a schedule vector of $\vec{S}^{(1)}(t)$ with positive probability such that more than two links are scheduled. Assume that this schedule vector is $\vec{x}_1$. Next, choose the smallest ℓ such that $x_{1\ell} = 1$. Let $\vec{e}_\ell$ be the schedule that only schedules link ℓ, and let $\vec{x}_1 - \vec{e}_\ell$ be

the schedule that removes link ℓ from $\vec{x}_1$. The distribution of $\vec{S}^{(2)}(t)$ is:

$$P\{\vec{S}^{(2)}(t)=\vec{x}_1-\vec{e}_\ell\}=p'_1,\quad P\{\vec{S}^{(2)}(t)=\vec{e}_\ell\}=p'_1,$$
$$P\{\vec{S}^{(2)}(t)=\vec{x}_0\}=p'_0-p'_1,\quad P\{\vec{S}^{(2)}(t)=\vec{x}_i\}=p'_i, i\geq 2.$$

Note that the schedule $\vec{x}_1$ is now splitted into two schedules $\vec{e}_\ell$ and $\vec{x}_1-\vec{e}_\ell$. Let $|\vec{x}_i|$ be the number of links scheduled by $\vec{x}_i$, and recall that $|\vec{x}_1|\geq 2$. We can then show that

$$\begin{aligned} p'_0-p'_1 &= 1-\textstyle\sum_{i\neq 0}p'_i-p'_1\geq 1-\sum_{i\neq 0}|\vec{x}_i|p'_i \\ &= 1-\textstyle\sum_\ell\sum_{i\in A_\ell}p'_i\geq 0, \end{aligned}$$

where the last inequality follows from the fact that $\vec{S}^{(1)}(\cdot)$ has property (10). Thus, the probability distribution of $\vec{S}^{(2)}(t)$ is valid. We call $\vec{S}^{(2)}(\cdot)$ a split version of $\vec{S}^{(1)}(\cdot)$. The key relationship between $\vec{S}^{(1)}(\cdot)$ and $\vec{S}^{(2)}(\cdot)$ is as follows.

Theorem 5: If we have an ergodic system $\vec{S}^{(1)}(\cdot)$ with property (10), and an ergodic system $\vec{S}^{(2)}(\cdot)$, which is the split version of system $\vec{S}^{(1)}(\cdot)$, then $T(\vec{S}^{(1)}(\cdot))\geq T(\vec{S}^{(2)}(\cdot))$. Moreover, $\vec{S}^{(2)}(\cdot)$ has property (10).

In other words, with the same window-based flow control, splitting will not increase the average throughput. To the best of our knowledge, this important relationship has not been reported in the literature. Clearly, if Theorem 5 holds, we can iteratively perform further splitting procedures on system $\vec{S}^{(2)}(\cdot)$. After finite number of iterations, we will reach a system $\vec{S}^{(3)}(\cdot)$ such that each schedule vector only schedules one link! The distribution of $\vec{S}^{(3)}(t)$ is $P\{\vec{S}^{(3)}(t)=\vec{e}_\ell\}=\mu_\ell, \ell=1,\cdots,H$ and $P\{\vec{S}^{(3)}(t)=\vec{x}_0\}=1-\sum_\ell\mu_\ell$. We have the following corollary.

Corollary 6: $T(\vec{S}^{(1)}(\cdot))\geq T(\vec{S}^{(3)}(\cdot))$.

Proof: If we can find a sequence of ergodic systems $\vec{S}_i(\cdot)$, $1\leq i\leq K$. $\vec{S}_1(\cdot)$ is a split version of $\vec{S}^{(1)}(\cdot)$, $\vec{S}_{i+1}(\cdot)$ is a split version of $\vec{S}_i(\cdot)$, and $\vec{S}_K(\cdot)=\vec{S}^{(3)}(\cdot)$. Then this corollary can be proved by Theorem 5. Recall that when we do the split procedure for a system, we pick a realization with the number of scheduled links (k) no smaller than 2 and replace it by two schedules with the number of scheduled links equal to $k-1$ and 1. Since the total number of schedules is finite, our split process will end up with a system $\vec{S}_K(\cdot)$ in a finite number of steps K when the schedule distribution only has realizations serving one link. Moreover, by Theorem 5, the splitting procedure will preserve the marginal probability of scheduling a link. It implies that, for system $\vec{S}_K(\cdot)$, $P\{\vec{S}_K(t)=\vec{e}_\ell\}=\mu'_\ell$. As a result, the distribution of $\vec{S}_K(t)$ is equal to the distribution of $\vec{S}^{(3)}(t)$, which means that system $\vec{S}_K(\cdot)=\vec{S}^{(3)}(\cdot)$. Finally, the ergodicity of system $\vec{S}_i(\cdot)$ can be proved by the same argument as Lemma 4. ∎

As we will see in Section IV-C, the lower bound of $T(\vec{S}^{(3)}(\cdot))$ can be more easily calculated.

B. Proof of Theorem 5

To prove Theorem 5, we use a specific stochastic ordering called supermodular ordering. We review the basic definitions used in our proof, and readers are referred to [31], [32] for other definitions and basic properties of stochastic ordering.

Definition 7: **(Supermodular Function)** A function $\phi(\vec{x}):\mathcal{R}^n\to\mathcal{R}$ is said to be supermodular if, for any n-dimensional vectors $\vec{x}_1,\vec{x}_2$, it satisfies that

$$\phi(\vec{x}_1)+\phi(\vec{x}_2)\leq\phi(\vec{x}_1\wedge\vec{x}_2)+\phi(\vec{x}_1\vee\vec{x}_2), \tag{11}$$

where $\wedge$ and $\vee$ mean componentwise minimum and maximum.

Definition 8: **(Supermodular Ordering)** Let $\mathcal{F}$ be the class of all supermodular functions from $\mathcal{R}^n$ into $\mathcal{R}$. For two n-dimensional random vectors $\vec{X}$ and $\vec{Y}$, $\vec{X}$ is said to be smaller than $\vec{Y}$ in the supermodular order (denoted by $\vec{X}\leq_{sm}\vec{Y}$) if $E[\phi(\vec{X})]\leq E[\phi(\vec{Y})]$, for all $\phi\in\mathcal{F}$.

An important relationship between $\vec{S}^{(1)}$ and its split version $\vec{S}^{(2)}$ is the following.

Lemma 9: If we have a system $\vec{S}^{(1)}(\cdot)$ with property (10), and another system $\vec{S}^{(2)}(\cdot)$, which is the split version of system $\vec{S}^{(1)}(\cdot)$, then $\vec{S}^{(1)}(t)\geq_{sm}\vec{S}^{(2)}(t)$.

Proof: Consider a supermodular function $\phi:\mathcal{R}^H\to\mathcal{R}$. Without loss of generality, assume that $P\{\vec{S}^{(1)}(t)=\vec{x}_i\}=p'_i, i=0,\cdots,I$, and the distribution of $\vec{S}^{(2)}(t)$ is:

$$P\{\vec{S}^{(2)}(t)=\vec{x}_1-\vec{e}_\ell\}=p'_1,\quad P\{\vec{S}^{(2)}(t)=\vec{e}_\ell\}=p'_1,$$
$$P\{\vec{S}^{(2)}(t)=\vec{x}_0\}=p'_0-p'_1,\quad P\{\vec{S}^{(2)}(t)=\vec{x}_i\}=p'_i, i\geq 2.$$

Recall that $\vec{x}_0=\vec{0}$. Then, we have that

$$\begin{aligned} & E[\phi(\vec{S}^{(1)}(t))]-E[\phi(\vec{S}^{(2)}(t))] \\ = & \textstyle\sum_i\phi(\vec{x}_i)p'_i-\phi(\vec{x}_1-\vec{e}_\ell)p'_1-\phi(\vec{e}_\ell)p'_1 \\ & -\phi(\vec{x}_0)(p'_0-p'_1)-\sum_{i\geq 2}\phi(\vec{x}_i)p'_i \\ = & \phi(\vec{x}_1)p'_1+\phi(\vec{x}_0)p'_1-\phi(\vec{x}_1-\vec{e}_\ell)p'_1-\phi(\vec{e}_\ell)p'_1 \\ \geq & 0, \end{aligned}$$

where the last equality follows from the definition of a supermodular function and the fact that $\vec{x}_1=\{\vec{x}_1-\vec{e}_\ell\}\vee\vec{e}_\ell$, and $\vec{x}_0=\{\vec{x}_1-\vec{e}_\ell\}\wedge\vec{e}_\ell$. ∎

Proof of Theorem 5: Fix a packet placement at time 0. Under window flow control with window size W, let f be a function that maps a given sequence of schedule vectors to the total number of packets leaving queue H at the end of time t. To prove Theorem 5, we first show that f is a supermodular function with respect to the schedule vector at time 1.[3] In other words, let $T_i(t)=f(\vec{z}_i,\vec{y}(2),\vec{y}(3),\cdots,\vec{y}(t))$, $i=1,\cdots,4$, where $\vec{z}_1=\vec{z}_3\vee\vec{z}_4$, $\vec{z}_2=\vec{z}_3\wedge\vec{z}_4$, and $\vec{y}(2),\vec{y}(3),\cdots,\vec{y}(t)$ are a sequence of deterministic schedules. We would like to show the following lemma.

Lemma 10: For any time t, $T_1(t)+T_2(t)\geq T_3(t)+T_4(t)$.

Proof: The proof is given in Appendix A. ∎

With this lemma, we have the following theorem.

Theorem 11:

$$\begin{aligned} & E[f(\vec{S}^{(2)}(1),\cdots,\vec{S}^{(2)}(k-1),\vec{S}^{(1)}(k),\cdots,\vec{S}^{(1)}(t))] \\ \geq & E[f(\vec{S}^{(2)}(1),\cdots,\vec{S}^{(2)}(k),\vec{S}^{(1)}(k+1),\cdots,\vec{S}^{(1)}(t))] \end{aligned}$$

, for any $k=1,2,\ldots,t$.

Proof of Theorem 11: To ease the presentation of the proof, denote $\vec{S}^{(i)}(t_1,t_2)$ as the sequence

[3]Readers why not show directly that f is a supermodular function with respect to the entire sequences. Unfortunately, we can construct counter examples to show that this is not true. One example will be given in Appendic C

$\vec{S}^{(i)}(t_1), \cdots, \vec{S}^{(i)}(t_2)$ for system $\vec{S}^{(i)}(\cdot)$. We first show the following inequality.

$$E[f(\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k,t))|\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k+1,t)] \geq E[f(\vec{S}^{(2)}(1,k), \vec{S}^{(1)}(k+1,t))|\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k+1,t)] \quad (12)$$

If (12) holds, take expectation of (12), we have that

$$E[f(\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k,t))] \geq E[f(\vec{S}^{(2)}(1,k), \vec{S}^{(1)}(k+1,t))], \quad (13)$$

which is exactly what we want to prove.

To show (12), assume that $\vec{S}^{(2)}(1,k-1)$ is equal to a sequence of fixed schedules $\vec{y}(1,k-1)$ and that $\vec{S}^{(1)}(k+1,t)$ is equal to a sequence of fixed schedules $\vec{y}(k+1,t)$. Assume that the distribution of $\vec{S}^{(1)}(k)$ is $P\{\vec{S}^{(1)}(k) = \vec{x}_i\} = p'_i$. Since system $\vec{S}^{(2)}(\cdot)$ is a split version of $\vec{S}^{(1)}(\cdot)$, the distribution of $\vec{S}^{(2)}(k)$ is:

$$P\{\vec{S}^{(2)}(k) = \vec{x}_1 - \vec{e}_\ell\} = p'_1, \quad P\{\vec{S}^{(2)}(k) = \vec{e}_\ell\} = p'_1,$$
$$P\{\vec{S}^{(2)}(k) = \vec{x}_0\} = p'_0 - p'_1, \quad P\{\vec{S}^{(2)}(k) = \vec{x}_i\} = p'_i, i \geq 2.$$

Denote $\phi(\vec{z}) = f(\vec{y}(1,k-1), \vec{z}, \vec{y}(k+1,t))$. We have that

$$\begin{aligned}
&E[f(\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k,t))|\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k+1,t)] \\
&- E[f(\vec{S}^{(2)}(1,k), \vec{S}^{(1)}(k+1,t))|\vec{S}^{(2)}(1,k-1), \vec{S}^{(1)}(k+1,t)] \\
&= \sum_i \phi(\vec{x}_i)p'_i - \phi(\vec{x}_1 - \vec{e}_\ell)p'_1 - \phi(\vec{e}_\ell)p'_1 \\
&\quad - \phi(\vec{x}_0)(p'_0 - p'_1) - \sum_{i\geq 2} \phi(\vec{x}_i)p'_i \\
&= \phi(\vec{x}_1)p'_1 + \phi(\vec{x}_0)p'_1 - \phi(\vec{x}_1 - \vec{e}_\ell)p'_1 - \phi(\vec{e}_\ell)p'_1 \geq 0.
\end{aligned}$$

The last inequality is an application of Lemma 10. Notice that $\vec{x}_0 = \{\vec{x}_1 - \vec{e}_\ell\} \wedge \vec{e}_\ell$, $\vec{x}_1 = \{\vec{x}_1 - \vec{e}_\ell\} \vee \vec{e}_\ell$, and $\vec{y}(1,k-1)$ will generate an arbitrary initial packet placement. Let $\phi(\vec{x}_1)$, $\phi(\vec{x}_0)$, $\phi(\vec{x}_1 - \vec{e}_\ell)$, and $\phi(\vec{e}_\ell)$ be the throughput of systems Y_1, Y_2, Y_3, and Y_4. Then we can apply Lemma 10 to have the last inequality. ■

This theorem can be used to show the following theorem.

Theorem 12:

$$\begin{aligned}
&E[f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))] \\
\geq\ &E[f(\vec{S}^{(2)}(1), \vec{S}^{(2)}(2), \cdots, \vec{S}^{(2)}(t))].
\end{aligned}$$

Proof of Theorem 12:

$$\begin{aligned}
&E[f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))] \\
\geq\ &E[f(\vec{S}^{(2)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))] \\
\geq\ &E[f(\vec{S}^{(2)}(1), \vec{S}^{(2)}(2), \vec{S}^{(1)}(3) \cdots, \vec{S}^{(1)}(t))] \\
\geq\ &\vdots \\
\geq\ &E[f(\vec{S}^{(2)}(1), \vec{S}^{(2)}(2), \cdots, \vec{S}^{(2)}(t))].
\end{aligned}$$

Each inequality is due to the application of Theorem 11 by changing a random schedule vector at a particular time t_1 from $\vec{S}^{(1)}(t_1)$ to $\vec{S}^{(2)}(t_1)$. ■

We continue the proof of Theorem 5. Our next step is to show that

$$\lim_{t\to\infty} f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))/t = T(\vec{S}^{(1)}(\cdot)). \quad (14)$$

Let $\mathcal{Q}$ be the set of all states. Furthermore, define function $g : \mathcal{Q}^2 \to \mathcal{R}$ as follows. $g(\vec{Q}(t), \vec{Q}(t+1)) = 1$ if there is a arriving packet to the destination node for the transition from state $\vec{Q}(t)$ to $\vec{Q}(t+1)$ and 0 otherwise. Therefore,

$$f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))/t = \frac{1}{t}\sum_{k=1}^{t} g(\vec{Q}(k), \vec{Q}(k+1)).$$

By the ergodic theorem [29, corollary 4.1],

$$\lim_{t\to\infty} \frac{1}{t}\sum_{k=1}^{t} g(\vec{Q}(k), \vec{Q}(k+1)) = \sum_{i_0,i_1} g(i_0,i_1)\pi(i_0)p_{i_0 i_1},$$

where $\pi(i_0)$ is the stationary distribution, and $p_{i_0 i_1}$ is the transition probability form state i_0 to i_1. It is not hard to verify that

$$\sum_{i_0,i_1} g(i_0,i_1)\pi(i_0)p_{i_0 i_1} = \mu'_H P(Q_H \neq 0) = T(\vec{S}^{(1)}(\cdot)).$$

Hence, we have proved (14). Similarly,

$$\lim_{t\to\infty} f(\vec{S}^{(2)}(1), \vec{S}^{(2)}(2), \cdots, \vec{S}^{(2)}(t))/t = T(\vec{S}^{(2)}(\cdot)). \quad (15)$$

We also know that $|f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t)/t| \leq 1$ because there are at most one outgoing packet at any given time. By bounded convergence theorem [33, Theorem 16.5],

$$\begin{aligned}
&E[\lim_{t\to\infty} f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))/t] \\
=\ &\lim_{t\to\infty} E[f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))/t].
\end{aligned} \quad (16)$$

This implies that

$$\lim_{t\to\infty} E[f(\vec{S}^{(1)}(1), \cdots, \vec{S}^{(1)}(t))]/t = T(\vec{S}^{(1)}).$$

With the same argument, we can also show that $\lim_{t\to\infty} E[f(\vec{S}^{(2)}(1), \cdots, \vec{S}^{(2)}(t))]/t = T(\vec{S}^{(2)})$. Now, we have from Theorem 12 that

$$\begin{aligned}
&E[f(\vec{S}^{(1)}(1), \vec{S}^{(1)}(2), \cdots, \vec{S}^{(1)}(t))/t] \\
\geq\ &E[f(\vec{S}^{(2)}(1), \vec{S}^{(2)}(2), \cdots, \vec{S}^{(2)}(t))/t].
\end{aligned} \quad (17)$$

Let t go to infinity. We can then conclude that $T(\vec{S}^{(1)}(\cdot)) \geq T(\vec{S}^{(2)}(\cdot))$ Because the marginal probability of scheduling link ℓ in $\vec{S}^{(1)}(\cdot)$ and $\vec{S}^{(2)}(\cdot)$ are the same for each ℓ, it is easy to see that $\vec{S}^{(2)}(\cdot)$ still has property (10). This ends the proof of Theorem 5. ■

C. Throughput Lower Bound and Delay Upper Bound

As discussed in Section IV-A, we can use Theorem 5 to show that $T(\vec{S}^{(1)}(\cdot)) \geq T(\vec{S}^{(3)}(\cdot))$. Consider another system $\vec{S}^{(4)}(\cdot)$. The distribution of $\vec{S}^{(4)}(t)$ is $P\{\vec{S}^{(4)}(t) = \vec{e}_\ell\} = \mu', \ell = 1, \cdots, H$ and $P\{\vec{S}^{(4)}(t) = \vec{x}_0\} = 1 - H\mu'$. It can be verified that $T(\vec{S}^{(3)}(\cdot)) \geq T(\vec{S}^{(4)}(\cdot))$.

Theorem 13: $T(\vec{S}^{(3)}(\cdot)) \geq T(\vec{S}^{(4)}(\cdot))$.

Proof: The proof is given in Appendix B. ■

This result is intuitive because, for every link ℓ, $P\{\vec{S}^{(3)}(t) = \vec{e}_\ell\} \geq P\{\vec{S}^{(4)}(t) = \vec{e}_\ell\}$. Thus, the throughput of system $\vec{S}^{(3)}(\cdot)$ should be no smaller than the throughput of system $\vec{S}^{(4)}(\cdot)$. The throughput of system $\vec{S}^{(4)}(\cdot)$ has a closed-form solution.

Lemma 14: $T(\vec{S}^{(4)}(\cdot)) = \mu' W/(W+H-1)$.

Proof: Consider the Markov Chain of system $\vec{S}^{(4)}(\cdot)$ and a particular state $\vec{n} = (n_1, \cdots, n_H)$, where n_i is the number of packets at queue i. Since the only possible schedule for system $\vec{S}^{(4)}(\cdot)$ is $\vec{e}_\ell$ for all link ℓ, there are the same number of incoming states and outgoing states for an arbitrary state $\vec{n}$. If we assume that $P(\vec{n}) = x$ for any state $\vec{n}$, then it will satisfy the global balance equation. Therefore, to obtain the solution of this stationary Markov Chain, we only need to normalize $\sum P(\vec{n}) = 1$. Hence,

$$x = \frac{1}{\text{number of possible states}} = \frac{1}{\binom{W+H-1}{W}}.$$

We can then show that

$$P(Q_H = 0) = x(\sum_{\vec{n}} \mathbf{1}_{n_H=0}) = \frac{\binom{W+H-2}{W}}{\binom{W+H-1}{W}} = \frac{H-1}{W+H-1},$$

and

$$T(\vec{S}^{(4)}(\cdot)) = \mu'[1 - P(Q_H = 0)] = \mu' \frac{W}{W+H-1}.$$

■

This lemma implies that we have a throughput lower bound for system $\vec{S}^{(3)}(\cdot)$. Note that the MC for system $\vec{S}^{(4)}(\cdot)$ is similar to that of a closed tandem $M/M/1$ queues with identical service rates [34].

Until this point, we have assumed that system $\vec{S}^{(1)}(\cdot)$ satisfies property (10). We now discuss how to treat the case when the original system $\vec{S}(\cdot)$ does not satisfy property (10). Suppose that the distribution of $\vec{S}(t)$ is $P\{\vec{S}(t) = \vec{x}_i\} = p_i, i = 0, \cdots, I$. Define system $\vec{S}^{(1)}(\cdot)$ as follows. For $i \neq 0$, let $p'_i = p_i/(\sum_\ell \mu_\ell)$, and $p'_0 = 1 - \sum_{i \neq 0} p'_i$. The distribution of $\vec{S}^{(1)}(t)$ is $P\{\vec{S}^{(1)}(t) = \vec{x}_i\} = p'_i, i = 0, \cdots, I$. Recall that $A_\ell = \{i | x_{i\ell} = 1\}$. We can then show that

$$\begin{aligned} \sum_\ell \mu'_\ell &= \sum_\ell \sum_{i \in A_\ell} p'_i = \sum_\ell \sum_{i \in A_\ell} p_i/(\sum_j \mu_j) \\ &= \sum_\ell \mu_\ell/(\sum_j \mu_j) \leq 1. \end{aligned}$$

We also have from (8) that $\mu'_\ell = \mu_\ell / \sum_i \mu_i \geq \mu / \sum_i \mu_i \triangleq \mu'$. Thus, system $\vec{S}^{(1)}(\cdot)$ has property (10). The relationship between $T(\vec{S}(\cdot))$ and $T(\vec{S}^{(1)}(\cdot))$ is given by the following lemma.

Lemma 15: $T(\vec{S}(\cdot)) = (\sum_\ell \mu_\ell) T(\vec{S}^{(1)}(\cdot))$.

Proof: By (9),

$$T(\vec{S}(\cdot)) = \mu_H[1 - P_S(Q_H = 0)]$$

and

$$T(\vec{S}^{(1)}(\cdot)) = \mu'_H[1 - P_{S^{(1)}}(Q_H = 0)]$$

If we can show that system $\vec{S}(\cdot)$ and system $\vec{S}^{(1)}(\cdot)$ have the same probability for queue H to be zero, this lemma can be proved by $\mu_H = \mu'_H(\sum_\ell \mu_\ell)$. To show that system $\vec{S}(\cdot)$ and system $\vec{S}^{(1)}(\cdot)$ have the same probability for queue H to be zero, notice that, by changing system $\vec{S}(\cdot)$ to system $\vec{S}^{(1)}(\cdot)$, for each state, we divide each incoming or outgoing transition probability by $\sum_\ell \mu_\ell$. Since the global balance equation will not change after the transition probability divided by a constant, system $\vec{S}(\cdot)$ and $\vec{S}^{(1)}(\cdot)$ will have the same steady state distribution, which implies that the probability for queue H to be zero is the same. ■

From this lemma and $T(\vec{S}^{(1)}(\cdot)) \geq T(\vec{S}^{(4)}(\cdot)) = \mu' W/(W+H-1)$, we have that

$$T(\vec{S}(\cdot)) = (\sum_\ell \mu_\ell) T(\vec{S}^{(1)}(\cdot)) \geq \frac{\mu W}{W+H-1}.$$

By Little's law, $W = T(\vec{S}(\cdot)) D(\vec{S}(\cdot))$. Thus,

$$D(\vec{S}(\cdot)) \leq (W+H-1)/\mu.$$

This finishes the proof of Proposition 3.

V. Implementation Issues

In this section, we discuss some practical issues for implementing our algorithm. We will address three components of our scheme: window-based flow control, virtual-rate computation, and scheduling. First, the window-based flow control requires a backward channel for communicating the ACKs. This backward channel can be easily implemented as follows. Immediately after a link transmits a flow m packet, the receiving node will respond with an acknowledgement, which piggy-backs an ACK for flow m that it has received from the destination in the past. With this mechanism, each link can be modelled as an upper queue for the forward direction and a lower queue for the backward direction. The window-based flow control for a given flow m can then be modelled as a $2H_m$-hop closed queueing network in Fig. 3. Note that both the upper queue and the lower queue will be served with probability bounded by Lemma 2. It is then easy to see that we can again use the technique of Section IV to derive the throughput and delay bounds. The only difference is that the number of hops is changed from H_m to $2H_m$, which does not affect our order-optimal delay result.

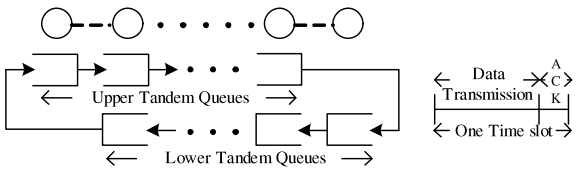


Fig. 3. Upper (resp. Lower) tandem queues store packets (resp. ACKs).

Second, in the virtual-rate computation algorithm, each node needs to collect the virtual-rate of each flow that passes through itself, and each source node needs to collect the sum of the dual variables along its route. In practice, such information exchange can be easily achieved by piggy-backing the virtual-rate information on each packet sent by the source node and piggy-backing the sum of the dual-variables on each ACK sent by the destination node. Note that although the virtual rates and dual variables are updated asynchronously, our window-based flow control algorithm guarantees an upper bound on the expected delay of such information exchange. Hence, we expect that the virtual-rate computation algorithm will still converges with suitable choices of the step sizes [27], [35].

Finally, in the scheduling algorithm, each transmitting node must collect the virtual-rates around the receiving node. Again, this information can be piggy-backed on data packets and ACKs. We note that the transmitter may now attempt with outdated information, but it will not affect our delay bound. This is because, after the virtual-rate computation algorithm converges, the virtual-rate will not change significantly.

Readers can see that under our proposed algorithm, each node only needs to perform a constant number of operations with a constant time of F mini-slots. This complexity is significantly lower than the algorithm in [22], which requires $\mathcal{O}(N)$ per-node operations.

VI. Simulation Results

We simulate our proposed algorithm using the linear topology in Fig. 1 with H links under the one-hop interference constraint. We use the improved version of our scheduling algorithm as discussed in Section III-B and set the number of backoff mini-slots $F = 32$. The window sizes of each short flow and the long flow are 2 and $2H$, respectively. The utility function is $H\log(\cdot)$ for the long flow and $5\log(\cdot)$ for each short flow. Hence, when we increase the number of hops, the optimal rate assignment for the flows will be the same. This will help us to observe the relationship between average delay and the number of hops at a fixed throughput.

We first compare the performance of our proposed algorithm with the standard back-pressure algorithm (for different step sizes) and the *shadow back-pressure* algorithm proposed in [11]. Fig. 4(a) shows that the average delay of our algorithm increases linearly with the number of hops. On the other hand, at all step sizes, the average delay of the back-pressure algorithm increases quadratically with the number of hops. Therefore, our algorithm outperforms the back-pressure algorithm in the delay performance when $H \geq 7$ even though the back-pressure also utilizes centralized computation. Moreover, our average delay curve is below the delay upper bound derived in Section IV. This verifies our delay analysis result.

In Fig. 4(b), we plot the corresponding long-flow throughput of our algorithm versus back-pressure and SBP. We can see that the throughput of our distributed algorithm is indeed more than half of the centralized and high-complexity back-pressure algorithm. Another interesting observation is that when the step size is large (BP-16), the throughput differs significantly from that at smaller step sizes. This indicates that the delay reduction (in Fig. 4(a)) of the back-pressure algorithm at such a large step size is at the cost of losing its optimal contorl capability. In contrast, the step size in our proposed algorithm does not directly affect the delay. Finally, note that although the average delay curve of shadow back-pressure algorithm also shows linear-scaling, it requires roughly 10000 time slots for the whole algorithm to converge, and the total queue length inside the network will first rise to a very large value as shown in Fig. 4(c). Therefore, the average delay of the first 1000 outgoing packets of the long flow is nearly 1000. In contrast, because of the window-based flow control, the total queue backlog of our algorithm is consistently the lowest throughout the simulation at all time (even at the transient period).

In Fig. 4(d), we demonstrate the per-flow controllability of our scheme. We first plot the throughput-delay curve for the long flow by letting the window size of each short flow be 2 and varying the window size of the long flow. As the window size increases, the average throughput of the long flow will approach to the limit, and the delay will increase linearly with window size. This curve shows that when the window size is equal to $2H$, we can achieve a good throughput-delay tradeoff. Next, we plot the throughput-delay curve for the long flow by letting the window size of the long flow be $2H$ and varying the window size of the short flows. As shown in Fig. 4(d), all points are concentrated around a small region, which implies that the performance of the long flow does not change when the window size of the short flows changes.

VII. Extensions to General Interference Constraints

In this section, we discuss how to extend our joint congestion and scheduling algorithm to the wireless networks under the general interference constraints, i.e., every link ℓ will interfere with a given set of other links. We will first present a joint congestion control and scheduling algorithm, which guarantees a lower bounded flow m scheduling probability for all the links that flow m passes by. Then the stochastic dominance method can again be applied.

Let $\mathcal{E}_\ell$ be the interference set of link ℓ. We adopt the convention that $\ell \in \mathcal{E}_\ell$, i.e., $\mathcal{E}_\ell = \{\ell\} \cup \{\ell' : \ell' \in E$ and ℓ' interferes with $\ell\}$. Assume that the interference relationship is symmetric, i.e., if $k \in \mathcal{E}_\ell$ then $\ell \in \mathcal{E}_k$. Now, adopt the definition of interference degree. The interference degree of a link ℓ is the maximum number of links within its interference range that can be activated simultaneously without interfering with each other. The interference degree $\mathcal{K}$ of a network is the maximum interference degree over all links.

Let Ω' be the capacity region for the network. We have that $\Psi'_0/\mathcal{K} \subseteq \Omega' \subseteq \Psi'_0$ [10], where

$$\Psi'_0 = \left\{ \vec{r} \left| \sum_{\ell \in \mathcal{E}_k} \sum_m L_m^\ell r_m \leq \mathcal{K}, \text{ for all links } k \right. \right\}.$$

A. Virtual-Rate Computation

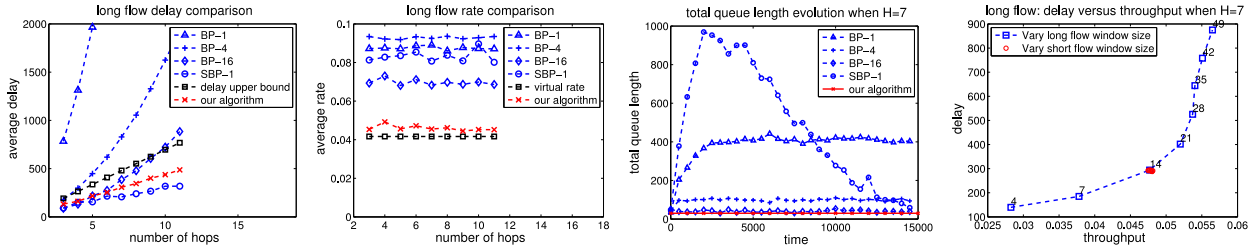
We first solve the following optimization problem.

$$\max_{r_m \geq 0} \sum_m U_m(r_m), \ \vec{r} \in \Psi'_0/\mathcal{K}. \tag{18}$$

Specifically, associate a dual variable $\lambda_k \geq 0$ to each constraint in problem (18). Let c_{km} be the number of flow m's passing links that interfere with link k. The objective function of the dual problem of (18) becomes:

$$D(\vec{\lambda}) := \max_{r_m \geq 0} \sum_m U_m(r_m) - \sum_k \lambda_k \left(\sum_m r_m c_{km} - 1 \right).$$

We can then use the following gradient algorithm to minimize $D(\vec{\lambda})$ and compute the optimal virtual-rates.



(a) (b) (c) (d)

Fig. 4. BP-α represents the back-pressure algorithm with step size α. SBP-α represents the shadow back-pressure algorithm with step size α. Numbers along the curve of 4(d) are the window size of the long flow.

Virtual-Rate Computation Algorithm: At each time t,

1) The source node of flow m updates r_m by equation:

$$r_m(t) = U_m'^{-1}(\sum_{k\in m} \lambda_k(t)c_{km}),$$

where $k \in m$ indicates that link k interferes with at least one of flow m's passing links.

2) Each node updates the dual variables by equation:

$$\lambda_k(t+1) = \left[\lambda_k(t) + \gamma_k\left(\sum_{m:k\in m} r_m(t)c_{km} - 1\right)\right]^+,$$

where $\gamma_k > 0$ is the step size, and $[\cdot]^+$ denotes the projection to $[0,\infty)$.

Since the constraints in problem (18) are still linear, using similar techniques as [27], one can show that as long as γ_k are sufficiently small, the above algorithm will converge to the optimal solution of (18).

B. Scheduling Algorithm

The scheduling algorithm is as follows. Let $a_\ell(t) = \sum_m L_m^\ell r_m(t)$ be the sum of the virtual rate over link ℓ, and let $x_\ell(t) = \max_{i\in\mathcal{E}_\ell}\left(\sum_{k\in\mathcal{E}_i} a_k(t)\right)$.

Rate-based Scheduling Algorithm: At each time slot t,

1) Each link ℓ first computes $P_\ell = \log(F)a_\ell(t)/x_\ell(t)$.
2) Each link then randomly picks a backoff time (B) with distribution: $P\{B = F+1\} = e^{-P_\ell}$ and $P\{B = f\} = e^{-P_\ell\frac{f-1}{F}} - e^{-P_\ell\frac{f}{F}}, f = 1,\cdots,F$. If $F+1$ is picked by link ℓ, it will not attempt to transmit in this time slot.
3) When the backoff timer for a link expires, it begins transmission unless it has already heard a transmission from one of its interfering links. If two or more links that interfere begin transmissions simultaneously, a collision occurs, and both transmissions fail.
4) When a link begins transmission, it will randomly choose a passing flow m to serve with probability $r_m(t)/a_\ell(t)$.

C. Performance Analysis

Let r_m^* be the optimal solution of (18), and let $r_m'^*$ be the optimal solution to the following optimization problem:

$$\max_{r_m\geq 0}\sum_m U_m(r_m),\ \vec{r}\in\Omega'/\mathcal{K}. \tag{19}$$

We have the following lemma.

Lemma 16: $\sum_m U_m(r_m^*) \geq \sum_m U_m(r_m'^*)$.

Proof: Since r_m^* is the optimal solution of (18), the total utility of r_m^* will be larger than any rate vector in $\Psi_0'/\mathcal{K}$. Also, by $\Omega' \subseteq \Psi_0'$, we have that $\Omega'/\mathcal{K} \subseteq \Psi_0'/\mathcal{K}$. Thus, $r_m'^*$ is a rate vector in $\Psi_0'/\mathcal{K}$, and $\sum_m U_m(r_m^*) \geq \sum_m U_m(r_m'^*)$. ■

Lemma 17: For any $\epsilon \in (0,1)$, if flow m passes through link ℓ, we can choose F large enough such that the probability that link ℓ will schedule flow m at time slot t is no smaller than $r_m(t)(1-\epsilon)/x_\ell(t)$.

Proof: The proof of this lemma is similar to the proof of Lemma 1 in [10]. In the following proof, we drop the index t from the notation $P_\ell(t)$ when there is no cause of confusion. Let E_ℓ be the event that link ℓ is scheduled. By equation (17) in [10], we have that

$$P(E_\ell) \geq (e^{\frac{P_\ell}{F}} - 1)\sum_{f=1}^{F} e^{-\frac{f}{F}\sum_{h\in\mathcal{E}_\ell} P_h}. \tag{20}$$

We first find an upper bound for the term $\sum_{h\in\mathcal{E}_\ell} P_h$ in equation (20). Let $\alpha = \log(F)$. At time t, we have that

$$\sum_{h\in\mathcal{E}_\ell} P_h = \alpha\sum_{h\in\mathcal{E}_\ell}\frac{a_h(t)}{\max\limits_{i\in\mathcal{E}_\ell}\left(\sum\limits_{k\in\mathcal{E}_i} a_k(t)\right)} \leq \alpha. \tag{21}$$

From (20) and (21), we know that

$$\begin{aligned} P(E_\ell) &\geq (e^{\frac{P_\ell}{F}} - 1)\sum_{f=1}^{F} e^{-\alpha\frac{f}{F}} \geq \frac{P_\ell}{F}\sum_{f=1}^{F} e^{-\alpha\frac{f}{F}} \\ &= \frac{P_\ell}{F}\frac{1-e^{-\alpha}}{1-e^{-\frac{\alpha}{F}}}e^{-\frac{\alpha}{F}}. \end{aligned}$$

Since $\alpha = \log(F)$ and $F > 1$, we can see that $\frac{\alpha}{F}/(1-e^{-\alpha/F}) \geq 1$, $e^{-\alpha/F} \geq 1-\log(F)/F$, and $1-e^{-\alpha} = 1-1/F$. Hence,

$$\begin{aligned} P(E_\ell) &\geq \frac{a_\ell(t)}{x_\ell(t)}(1-\frac{\log(F)+1}{F}) \\ &> \frac{a_\ell(t)}{x_\ell(t)}(1-\epsilon), \end{aligned} \tag{22}$$

where F is chosen such that $\epsilon > \frac{\log(F)+1}{F}$. We can then conclude that

$$\begin{aligned} & P(\text{ link } \ell \text{ is scheduled to transmit a flow } m \text{ packet}) \\ = \ & P(\text{flow } m \text{ is scheduled}|E_\ell)P(E_\ell) \\ \geq \ & \frac{r_m(t)}{a_\ell(t)}\frac{a_\ell(t)}{x_\ell(t)}(1-\epsilon) \\ = \ & \frac{r_m(t)}{x_\ell(t)}(1-\epsilon). \end{aligned}$$

■

After the virtual-rate computation algorithm converges, the value of $r_m(t)$ will be equal to the optimal solution r_m^*. Furthermore, we have from the constraints of optimization problem (18) that $x_\ell(t) \leq 1$. It then follows from Lemma 17 that every hop along the path of flow m will serve flow m with probability no smaller than $r_m^*(1-\epsilon)$, independent across time slots, where ϵ is a small positive constant that only depends on F. With the same stochastic dominance technique, it is easy to see that the bounds provided in Proposition 3 will still hold. Therefore, our algorithm can utilize the capacity with a factor $\mathcal{K}$ loss of throughput and have a per-flow delay bound that only increases linearly with the number of hops under the general interference constraints. Define the in-degree and out-degree of a node v as the number of links in E that ends in v and orginates from v. The directed degree of a link ℓ is the sum of the out-degree of $b(\ell)$ and in-degree of $d(\ell)$. The maximum directed link degree in G, $\triangle_G$, is the maximum directed degree of any link in E. The value of $\mathcal{K}$ can be upper bounded by $\max(\triangle_G - 2, 1)$ [8]. Note that in [22], under the two-hop interference constraint their algorithm requires a factor $4\triangle^2$ loss of throughput to achieve linear delay bound, where $\triangle$ is the maximum degree of the vertices. We know $\triangle_G \leq 2\triangle$. Thus, our throughput loss factor is again smaller.

VIII. Conclusion

In this paper, we propose a low-complexity and distributed algorithm for joint congestion control and scheduling in multihop wireless networks under the one-hop interference constraint. The main ideas of the proposed algorithm are to control the congestion with window-based flow control and to use both virtual-rate information and queue information (rather than just queue information) to perform scheduling. Our scheduling algorithm is fully distributed and only requires a constant time (independent of network size) to compute a schedule [10]. We prove that our congestion control and scheduling algorithm can utilize nearly half of the capacity region and provide a per-flow delay bound that increases linearly with the number of hops. Our analysis uses a novel stochastic dominance approach to compare the throughput of our system with another system, which we can compute the exact throughput. We then use Little's law to derive a per-flow delay upper-bound. The methodology in this paper can also be extended to more general interference models defined by interference sets (see [29] for details). In our future work, it will be interesting to know if it is possible to design solutions that can achieve the same delay order with an even smaller sacrifice of capacity region.

Appendix A
Proof of Lemma 10

We first give a series of definitions and lemmas required for Lemma 10. Since we only care about the total number of outgoing packets for each system, the service discipline of each queue is not very important. Assume that each queue will use FIFO discipline.

Definition 18: **(Packet Label)** For a closed tandem network with H hops, consider an arbitrary initial placements for the W packets in the system. We label packets from 1 to W according to their distance to the destination and their position in the current queue. A packet has smaller index if it is closer to the destination or it will be served earlier at the current queue.

An example of initial packet label for a closed tandem network with 3 hops and 5 packets is given in Fig. 5

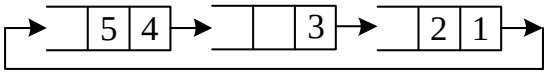


Fig. 5.

Note that our packet label ensures that the outgoing packet sequence will be $123\cdots W123\cdots W\cdots$ and so on.

Definition 19: **(Packet Round)** For a closed tandem network with H hops, define $R(i,t)$ as the number of times that packet i has been served by queue H at the end of time t.

By looking at the outgoing packet sequence, we know that the packet with a smaller index will always have more rounds, i.e.,

$$R(i,t) \geq R(j,t), \text{ if } i < j. \tag{23}$$

Also, the round difference between packets with different indices will be bounded by 1, i.e.,

$$R(j,t) + 1 \geq R(i,t), \text{ if } i < j. \tag{24}$$

Notice that at the end of time t, the total number of outgoing packets will be $\sum_j R(j,t)$. If the value of $R(i,t)$ is known for some packet i, we then have from (23) and (24) that

$$\begin{aligned} \textstyle\sum_j R(j,t) &\leq (i-1)(R(i,t)+1) + (W-i+1)R(i,t) \\ &= R(i,t)W + i - 1. \end{aligned} \tag{25}$$

Definition 20: **(Packet Position)** For a closed tandem network with H hops, index the queues as $0, \cdots, H-1$ starting from the source node. Define $P(i,t)$ as the index of the queue, which packet i stays at the end of time t.

Note that for packet position, we use the convention that if packet i stays at queue q, the packet position is $q-1$. In the following, when we refer to a particular queue, we mean this relabelled index. According to the definition, $0 \leq P(i,t) \leq H-1, \forall\, i,t$. Also, by our packet label and FIFO discipline, if we know the packet rounds of two packets, we can deduce the relationship of packet positions, and vice versa. To ease the presentation of this property, we have the following definition.

Definition 21: At time t, a packet i is said to be ahead of packet j if $P(i,t) > P(j,t)$ or $P(i,t) = P(j,t)$, but packet i will be served earlier than packet j by its current staying queue.

For example, in Fig. 5, packet 1 is ahead of packet 2 and packet 3. Then we have the following lemma.

Lemma 22: Assume that $i < j$. At time slot t,

(i) $R(i,t) = R(j,t)$ if and only if packet i is ahead of packet j.

(ii) $R(i,t) > R(j,t)$ if and only if packet j is ahead of packet i.

Proof: This result is an immediate consequence of our packet label and FIFO discipline. ∎

Definition 23: **(Packet Relative Position)** For a closed tandem network with H hops, define $RP(i,t)$ as the relative position of packet i at the end of time t. Its value is determined by the value of $P(i,t)$ and $R(i,t)$ by the following equality, $RP(i,t) = R(i,t)H + P(i,t)$.

A simple fact for the relative position of packet i is that

$$RP(i,t) \leq RP(i,t+1) \leq RP(i,t) + 1. \tag{26}$$

This is true because the relative position will increase by at most 1 at one time slot and never decrease. Unlike packet position, the order of relative packet positions between two packets never changes.

Lemma 24: If $i < j$, $RP(i,t) \geq RP(j,t)$, for all t.

Proof: We have from (23) and (24) that $R(i,t) = R(j,t)$ or $R(i,t) = R(j,t)+1$. If $R(i,t) = R(j,t)+1$, we can show that

$$RP(i,t) - RP(j,t) = H + P(i,t) - P(j,t) \geq 0.$$

On the other hand, if $R(i,t) = R(j,t)$, it follows from Lemma 22 (i) that $P(i,t) \geq P(j,t)$. Hence,

$$RP(i,t) - RP(j,t) = P(i,t) - P(i,t) \geq 0.$$

∎

The relationship of relative packet positions between two packets is also very similar to the relationship of packet positions. For example, the difference of packet positions is bounded by $H-1$. A similar result for packet relative position is presented as follows.

Lemma 25: $RP(W,t) \geq RP(1,t) - H$.

Proof: We have from (23) and (24) that $R(1,t) = R(W,t)$ or $R(1,t) = R(W,t) + 1$. If $R(1,t) = R(W,t)$, we can show that

$$RP(1,t) - RP(W,t) = P(1,t) - P(W,t) \leq H.$$

On the other hand, if $R(1,t) = R(W,t) + 1$, it follows from Lemma 22 (ii) that $P(1,t) \leq P(W,t)$. Hence,

$$RP(1,t) - RP(W,t) = H + P(1,t) - P(W,t) \leq H.$$

∎

Now, consider the following four deterministic systems Y_i, $i = 1, \cdots, 4$. They have the same initial packets placement. For system Y_i, from time 1 to some time t, it has a sequence of deterministic schedules, $\vec{z}_i, \vec{y}(2), \cdots, \vec{y}(t)$. It is easy to see that the only difference for these four deterministic systems is the schedule at the first time slot. Furthermore, $\vec{z}_1, \vec{z}_2, \vec{z}_3$, and $\vec{z}_4$ satisfy the following set of equalities.

$$\vec{z}_2 = \vec{z}_3 \wedge \vec{z}_4, \vec{z}_1 = \vec{z}_3 \vee \vec{z}_4. \tag{27}$$

Notice that $\vec{z}_2 \preccurlyeq \{\vec{z}_3, \vec{z}_4\} \preccurlyeq \vec{z}_1$, where $\preccurlyeq$ means componentwise smaller.

Let $T_k(t)$ be the total number of outgoing packets (i.e., the number of packets that leaves the last queue) at the end of time t in system Y_k. Let $P_k(i,t)$, $R_k(i,t)$, $RP_k(i,t)$ be the position, round, and relative position of packet i for system Y_k at the end of time t. The proof of Theorem 11 is closely connected with the throughput comparison of these four deterministic systems. Since, for a system, the number of outgoing packets can be computed by the packet rounds, the first step of the comparison is to find the relationship of packet rounds between different systems.

Lemma 26: For system Y_k, Y_w, $k, w = 1, \cdots, 4$, and packet i, j,

(i) if $RP_k(i,t) \geq RP_w(j,t) - nH$, then $R_k(i,t) \geq R_w(j,t) - n$.

(ii) if $RP_k(i,t) = RP_w(j,t) - nH$, then $R_k(i,t) = R_w(j,t) - n$ and $P_k(i,t) = P_w(j,t)$.

Proof: We prove (i) by contradiction. Assume that $R_w(j,t) - R_k(i,t) \geq n+1$. Thus,

$$\begin{aligned} & RP_w(j,t) - RP_k(i,t) \\ = \;& R_w(j,t)H + P_w(j,t) - R_k(i,t)H - P_k(i,t) \\ \geq \;& (n+1)H + P_w(j,t) - P_k(i,t) \\ \geq \;& (n+1)H + 0 - (H-1) \geq nH + 1. \end{aligned}$$

We have a contradiction.

Now we prove (ii). From $RP_k(i,t) = RP_w(j,t) - nH$, we know that

$$nH + [R_k(i,t) - R_w(j,t)]H = P_w(j,t) - P_k(i,t).$$

This implies that $P_w(j,t) - P_k(i,t)$ can be divided by H. Since $-(H-1) \leq P_w(j,t) - P_k(i,t) \leq H-1$, we can conclude that $P_w(j,t) - P_k(i,t) = 0$. This implies that $R_k(i,t) = R_w(j,t) - n$. ∎

Note that Lemma 26 is still true if $k = w$ or $i = j$. This lemma shows that if we can establish the order of relative packet positions between different systems, this order will be preserved for packet rounds. This implies that the problem of comparing throughput is intuitively the same as the problem of comparing relative positions, which is resolved by Lemmas 27, 29, and 31. The throughput comparison between the four systems is then presented in Lemmas 28, 30, and 32. We first present Lemma 27. This lemma is intuitive because if a system has more services than the other system, then the relative position of packet i should be no smaller than that of the other system.

Lemma 27: Consider system Y_k, Y_w. For any time t and packet i, if $\vec{z}_k \succcurlyeq \vec{z}_w$, then $RP_k(i,t) \geq RP_w(i,t)$

Proof: We prove this lemma by induction on time t. At time 0, $RP_k(i,0) = RP_w(i,0)$ because system Y_k and system Y_w have the same initial packet placements. At time 1, since the initial schedule $\vec{z}_w$ is componentwise smaller than $\vec{z}_k$, if packet i moves to the next queue in system Y_w, then packet i will also moves to the next queue in system Y_k. This implies that if $RP_w(i,1) = RP_w(i,0) + 1$, then $RP_k(i,1) = RP_k(i,0) + 1$. Thus, $RP_k(i,1) \geq RP_w(i,1)$.

Assume that $RP_k(i,t) \geq RP_w(i,t)$ for all packet i and $t \geq 1$. We would like to show that

$$RP_k(i,t+1) \geq RP_w(i,t+1). \tag{28}$$

Note that we can assume that

$$\begin{cases} RP_k(i,t) = RP_w(i,t), \\ RP_w(i,t+1) = RP_w(i,t) + 1. \end{cases} \tag{29}$$

Otherwise, (28) will hold by the facts that the relative position increases by at most 1 at one time slot, and the relative position is a non-decreasing function of time.

Consider the following two cases.

Case 1*:* $i = 1$. Since $RP_k(1,t) = RP_w(1,t)$, we have from Lemma 26 (ii) that

$$R_k(1,t) = R_w(1,t) \text{ and } P_k(1,t) = P_w(1,t). \tag{30}$$

Furthermore, we have from $RP_w(1,t+1) = RP_w(1,t) + 1$ that for system Y_w, packet 1 must have been at the front of queue $P_w(1,t)$ at the end of time t, and the queue is served at time $t+1$. Otherwise, it is impossible for packet 1 to move to the next queue at time $t+1$. Consider system Y_k. Suppose at time t packet 1 is also at the front of queue $P_k(1,t)$. Since $P_w(1,t)$ is served at time $t+1$, and system Y_w and Y_k have the same schedule after time 1, we have that $RP_k(1,t+1) = RP_k(1,t) + 1$. This finishes the proof. Suppose not, then, for system Y_k, packet W is at queue $P_k(1,t)$ at time t because the next packet in front of packet 1 must be packet W. This implies that packet W is ahead of packet 1 and $P_k(1,t) = P_k(W,t)$. We then have from Lemma 22 (ii) that $R_k(1,t) > R_k(W,t)$. Thus,

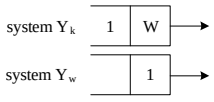


Fig. 6. For system Y_k, packet W may or may not be served at time $t+1$.

$$RP_k(1,t) = RP_k(W,t) + H. \tag{31}$$

From induction hypothesis, we also know that

$$RP_k(W,t) \geq RP_w(W,t). \tag{32}$$

It then follows from (31), (32), Lemma 25, and (29) that

$$\begin{aligned} RP_k(1,t) &= RP_k(W,t) + H \\ &\geq RP_w(W,t) + H \geq RP_w(1,t) = RP_k(1,t). \end{aligned}$$

Hence, $RP_w(W,t) + H = RP_w(1,t)$, and we have from Lemma 26 (ii) that $R_w(W,t) = R_w(1,t) - 1$ and $P_w(W,t) = P_w(1,t)$. From Lemma 22 (ii), it follows that, for system Y_w, packet W is ahead of packet 1. Since $P_w(W,t) = P_w(1,t)$, packet 1 will not be served at time $t+1$. This contradicts with (29).

Case 2*:* $i \neq 1$. The proof for this case is similar to Case 1. The only difference is that when we consider the packet ahead of packet i, its index will be smaller than i. With similar approach of Case 1, we have that

$$R_k(i,t) = R_w(i,t) \text{ and } P_k(i,t) = P_w(i,t). \tag{33}$$

Also, for system Y_w, packet i is at the front of queue $P_w(i,t)$ and served at time $t+1$. For system Y_k, packet $i-1$ is ahead of packet i, and $P_k(i-1,t) = P_k(i,t)$. We then have from Lemma 22 (i) that $R_k(i-1,t) = R_k(i,t)$. Thus,

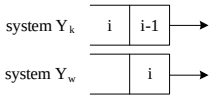


Fig. 7. For system Y_k, packet $i-1$ may or may not be served at time $t+1$.

$$RP_k(i,t) = RP_k(i-1,t). \tag{34}$$

From induction hypothesis, we also know that

$$RP_k(i-1,t) \geq RP_w(i-1,t). \tag{35}$$

It then follows from (34), (35), Lemma 24, and (33) that

$$\begin{aligned} RP_k(i,t) &= RP_k(i-1,t) \\ &\geq RP_w(i-1,t) \geq RP_w(i,t) = RP_k(i,t). \end{aligned}$$

Hence, $RP_w(i-1,t) = RP_w(i,t)$, and we have from Lemma 26 (ii) that $R_w(i-1,t) = R_w(i,t)$ and $P_w(i-1,t) = P_w(i,t)$. From Lemma 22 (i), it follows that, for system Y_w, packet $i-1$ is ahead of packet i. Since $P_w(i-1,t) = P_w(i,t)$, packet i will not be served at time $t+1$. This contradicts with (29). ■

Lemma 28: Consider system Y_k, Y_w. For any time t, if $\vec{z}_k \succcurlyeq \vec{z}_w$, then $T_k(t) \geq T_w(t)$.

Proof: By Lemma 27, $RP_k(i,t) \geq RP_w(i,t)$. Therefore, we have from Lemma 26 (i) that $R_k(i,t) \geq R_w(i,t)$. This implies that

$$T_k(t) = \sum_i R_k(i,t) \geq \sum_i R_w(i,t) = T_w(t).$$

■

Now, we show that, for the system with less service, the relative position of packet i (resp. W) can bound the relative position of packet with index $i+1$ (resp. 1) in another system. This is because for different systems, the schedule are only different at the first time slot. Therefore, for the system with more service, the relative position of packet $i+1$ (resp. 1) will only lead the packet with the same index in another system, but it will not pass the packet with the index i (resp. W) in another system. Note that the additional $-H$ term when we compare packet 1 and W comes from the fact that at time zero the difference of packet 1 relative position and packet W relative position is at most $H-1$.

Lemma 29: Consider system Y_k, Y_w. For any time t and packet i, if $\vec{z}_k \succcurlyeq \vec{z}_w$, then $RP_w(i,t) \geq RP_k(i+1,t)$, $i = 1 \cdots W-1$, and $RP_w(W,t) \geq RP_k(1,t) - H$.

Proof: We prove this lemma by induction on time t. At time 0,

$$\begin{aligned} RP_k(1,0) - RP_w(W,0) &= P_k(1,0) - P_w(W,0) \\ &\leq H - 1. \end{aligned} \tag{36}$$

For $i \neq W$, since system Y_k and Y_w have the same initial packet placements, we have that $RP_w(i,0) = RP_k(i,0)$. Furhtermore, by Lemma 24, $RP_k(i,0) \geq RP_k(i+1,0)$. Thus,

$$RP_w(i,0) \geq RP_k(i+1,0). \tag{37}$$

At time 1, four systems have different schedule, but we can still prove the inequalities. To see this, note that by (26) and (36),

$$\begin{aligned} RP_k(1,1) &\leq RP_k(1,0)+1 \leq RP_w(W,0)+H \\ &\leq RP_w(W,1)+H. \end{aligned}$$

For $i \neq W$ at time 1, if $RP_k(i+1,1) = RP_k(i+1,0)$, by (26) and (37),

$$RP_w(i,1) \geq RP_w(i,0) \geq RP_k(i+1,0) = RP_k(i+1,1).$$

If

$$RP_k(i+1,1) = RP_k(i+1,0)+1, \tag{38}$$

we know that, for system Y_k, packet $i+1$ must be at the front of queue $P_k(i+1,0)$. Hence, packet i is not at the same queue. Because packet i is ahead of packet $i+1$, and they are not in the same queue at the end of time 0, $P_k(i,0) > P_k(i+1,0)$. Therefore, we have that

$$RP_k(i,0) > RP_k(i+1,0). \tag{39}$$

Moreover, because the two systems have the same packet initial placement, $RP_w(i,0) = RP_k(i,0)$. Combined with (39), we have that

$$RP_w(i,0) > RP_k(i+1,0). \tag{40}$$

It then follows from (26), (40), and (38) that

$$RP_w(i,1) \geq RP_w(i,0) \geq RP_k(i+1,0)+1 = RP_k(i+1,1).$$

Now, suppose the inequalities hold at time t, i.e.,

$$\begin{cases} RP_w(i,t) \geq RP_k(i+1,t), i \neq W, \\ RP_w(W,t) \geq RP_k(1,t) - H. \end{cases} \tag{41}$$

We would like to show that they hold at time $t+1$. This would be proved in three steps.

Step 1: Prove $RP_w(W,t+1) \geq RP_k(1,t+1) - H$. From (41), we only need to consider the case when

$$\begin{cases} RP_w(W,t) = RP_k(1,t) - H, \\ RP_k(1,t+1) = RP_k(1,t)+1. \end{cases} \tag{42}$$

In all the other cases, $RP_w(W,t+1) \geq RP_k(1,t+1) - H$ because the relative position increases by at most 1 at one time slot, and the relative position is a non-decreasing function of time. Since $RP_w(W,t) = RP_k(1,t) - H$, by Lemma 26 (ii), we know that

$$P_w(W,t) = P_k(1,t). \tag{43}$$

Since $RP_k(1,t+1) = RP_k(1,t)+1$, for system Y_k packet 1 must be at the front of queue $P_k(1,t)$, and queue $P_k(1,t)$ is served at time $t+1$. By (43) and the fact that two systems have the same schedule after time 1, if, for system Y_w packet W is also at the front of queue $P_w(W,t)$, we then have $RP_w(W,t+1) \geq RP_k(1,t+1) - H$. Suppose not, then for system Y_w packet $W-1$ is at queue $P_w(W,t)$ because the next packet in front of packet W is packet $W-1$. This implies that packet $W-1$ is ahead of packet W, and $P_w(W-1,t) = P_w(W,t)$.

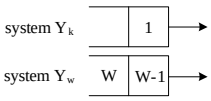


Fig. 8. For system Y_w, packet $W-1$ may or may not be served at time $t+1$.

By Lemma 22 (i), we know that $R_w(W-1,t) = R_w(W,t)$. Thus,

$$RP_w(W-1,t) = RP_w(W,t). \tag{44}$$

We then have from (44), (41), Lemma 25, and (42) that

$$\begin{aligned} RP_w(W,t) &= RP_w(W-1,t) \\ &\geq RP_k(W,t) \geq RP_k(1,t) - H = RP_w(W,t). \end{aligned}$$

Therefore, $RP_k(W,t) = RP_k(1,t) - H$. From Lemma 26 (ii), we have that $R_k(W,t) = R_k(1,t) - 1$ and $P_k(W,t) = P_k(1,t)$. It then follows from Lemma 22 (ii) that, for system Y_k, packet W must be ahead of packet 1. Since $P_k(W,t) = P_k(1,t)$, packet 1 will not be served at time $t+1$. This contradicts with (42).

Step 2: Prove $RP_w(1,t+1) \geq RP_k(2,t+1)$. With similar approach of Step 1, we only need to consider the case when

$$\begin{cases} RP_w(1,t) = RP_k(2,t), \\ RP_k(2,t+1) = RP_k(2,t)+1, \end{cases} \tag{45}$$

and we have that

$$R_w(1,t) = R_k(2,t) \text{ and } P_w(1,t) = P_k(2,t). \tag{46}$$

Furthermore, for system Y_k, packet 2 is at the front of queue $P_k(2,t)$ and served at time $t+1$. For system Y_w, packet W is ahead of packet 1, and $P_w(1,t) = P_w(W,t)$.

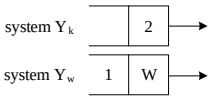


Fig. 9. For system Y_w, packet W may or may not be served at time $t+1$.

From Lemma 22 (ii), we know that $R_w(W,t) = R_w(1,t) - 1$. Thus,

$$RP_w(1,t) = RP_w(W,t) + H. \tag{47}$$

We then have from (47), (41), Lemma 24, and (45) that

$$\begin{aligned} RP_w(1,t) &= RP_w(W,t) + H \\ &\geq RP_k(1,t) \geq RP_k(2,t) = RP_w(1,t). \end{aligned}$$

Therefore, $RP_k(1,t) = RP_k(2,t)$. From Lemma 26 (ii), we have that $R_k(1,t) = R_k(2,t)$ and $P_k(1,t) = P_k(2,t)$. It then follows from Lemma 22 (i) that, for system Y_k, packet 1 must be ahead of packet 2. Since $P_k(1,t) = P_k(2,t)$, packet 2 can not be served at time $t+1$. This contradicts with (45).

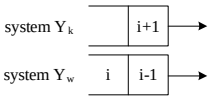
Step 3: Prove $RP_w(i,t+1) \geq RP_k(i+1,t+1), 1 < i < W$. With similar approach of Step 1, we only need to consider the case when

$$\begin{cases} RP_w(i,t) = RP_k(i+1,t), \\ RP_k(i+1,t+1) = RP_k(i+1,t)+1, \end{cases} \tag{48}$$

and we have that

$$RP_w(i,t) = RP_k(i+1,t) \text{ and } P_w(i,t) = P_k(i+1,t). \quad (49)$$

Also, for system Y_k, packet $i+1$ is at the front of queue $P_k(i+1,t)$ and served at time $t+1$. For system Y_w, packet $i-1$ is ahead of packet i, and $P_w(i-1,t) = P_w(i,t)$.



From Lemma 22 (i), we know that $R_w(i-1,t) = R_w(i,t)$. Thus,

$$RP_w(i,t) = RP_w(i-1,t). \quad (50)$$

We then have from (50), (41), Lemma 24 and (48) that

$$\begin{aligned} RP_w(i,t) &= RP_w(i-1,t) \\ &\geq RP_k(i,t) \geq RP_k(i+1,t) = RP_w(i,t). \end{aligned}$$

Therefore, $RP_k(i,t) = RP_k(i+1,t)$. From Lemma 26 (i), we have that $R_k(i,t) = R(i+1,t)$ and $P_k(i,t) = P_k(i+1,t)$. It then follows from Lemma 22 (i) that, for system Y_k, packet i must be ahead of packet $i+1$. Since $P_k(i,t) = P_k(i+1,t)$, packet $i+1$ can not be served at time $t+1$. This contradicts with (48). ■

Lemma 30: Consider systems Y_k, Y_w. For any time t, if $\vec{z}_k \succcurlyeq \vec{z}_w$, then $T_w(t) \geq T_k(t) - 1$.

Proof: At the beginning of time slot $t+1$, suppose that packet i is the next outgoing packets at the last queue for system Y_w, which means that it is ahead of other packets in the system Y_w. Therefore,

$$T_w(t) = R_w(i,t)W + (i-1) \quad (51)$$

Consider the following two cases.

Case 1: $i \neq W$. From Lemma 29, we know that $RP_w(i,t) \geq RP_k(i+1,t)$. We then have from Lemma 26 (i) that

$$R_w(i,t) \geq R_k(i+1,t). \quad (52)$$

By (51), (52), and (25), we conclude that

$$T_w(t) = R_w(i,t)W + (i-1) \geq R_k(i+1,t)W + i - 1 \geq T_k(t) - 1.$$

Case 2: $i = W$. From Lemma 29, we know that $RP_w(W,t) \geq RP_k(1,t) - H$. We then have from Lemma 26 (i) that

$$R_w(W,t) \geq R_k(1,t) - 1. \quad (53)$$

By (51), (53), and (25), we conclude that

$$T_w(t) = R_w(W,t)W + (W-1) \geq R_k(1,t)W - 1 \geq T_k(t) - 1.$$

■

Lemma 31: For any time t and any packet i, $RP_2(i,t) \geq \min[RP_3(i,t), RP_4(i,t)]$.

Proof: We prove this lemma by induction on time t. At time 0, for system Y_2, Y_3, and Y_4, they have the same initial packet placement. Furthermore, we know that $R_2(i,0) = R_3(i,0) = R_4(i,0) = 0$. We then have that $RP_2(i,0) = RP_3(i,0) = RP_4(i,0)$.

At time 1, we know that $\vec{z}_2 = \vec{z}_3 \wedge \vec{z}_4$. If a link is scheduled by $\vec{z}_2$, and a packet i is served, then, for system Y_2, the relative position of packet i will increase by 1. Since, for system Y_3 and Y_4, the relative position of packet i will increase at most 1. We have that $RP_2(i,1) \geq \min[RP_3(i,1), RP_4(i,1)]$. If a link ℓ is not scheduled by $\vec{z}_2$, either $\vec{z}_3$ or $\vec{z}_4$ will not schedule link ℓ. Suppose that packet i is at the front of queue ℓ. If $RP_4(i,1) = RP_4(i,0) + 1$, then $RP_3(i,1) = RP_3(i,0)$. Similarly, if $RP_3(i,1) = RP_3(i,0) + 1$, then $RP_4(i,1) = RP_4(i,0)$. This implies that $\min[RP_3(i,1), RP_4(i,1)] = \min[RP_3(i,0), RP_4(i,0)]$, and the inequality will hold at time 1.

After time 1, system Y_2, Y_3, and Y_4 will have the same schedules at every time slot. Assume that the inequality holds at time t, we want to prove that it holds at time $t+1$. Notice that if $RP_2(i,t+1) = RP_2(i,t) + 1$ then the inequality will hold trivially at time $t+1$. Therefore, we only consider the case when

$$RP_2(i,t+1) = RP_2(i,t). \quad (54)$$

Assume that packet i is at queue ℓ for system Y_2.

Consider the following three cases.

Case 1: $i = 1$ and link ℓ is scheduled at time $t+1$. Since $RP_2(1,t+1) = RP_2(1,t)$, packet 1 is not scheduled by link ℓ. Therefore, packet W is ahead of packet 1, and $P_2(W,t) = P_2(1,t)$. We then have from Lemma 22 (ii) that $R_2(W,t) = R_2(1,t) - 1$. Thus,

$$RP_2(1,t) = RP_2(W,t) + H. \quad (55)$$

From the induction hypothesis, $RP_2(W,t) \geq \min[RP_3(W,t), RP_4(W,t)]$. Without loss of generality, assume that $\min[RP_3(W,t), RP_4(W,t)] = RP_3(W,t)$. Therefore,

$$RP_2(W,t) \geq RP_3(W,t). \quad (56)$$

From (55), (56), Lemma 25, and Lemma 27, we have that

$$\begin{aligned} RP_2(1,t) &= RP_2(W,t) + H \geq RP_3(W,t) + H \\ &\geq RP_3(1,t) \geq RP_2(1,t). \end{aligned} \quad (57)$$

Hence, $RP_3(W,t) + H = RP_3(1,t)$. From Lemma 26 (ii), we know that $R_3(W,t) = R_3(1,t) - 1$ and $P_3(W,t) = P_3(1,t)$. It then follows from Lemma 22 (ii) that, for system Y_3, packet W is ahead of packet 1. Since $P_3(W,t) = P_3(1,t)$, packet 1 can not be served at time $t+1$. Therefore,

$$RP_3(1,t+1) = RP_3(1,t). \quad (58)$$

From (54), (57), and (58), we conclude that

$$\begin{aligned} RP_2(1,t+1) &= RP_2(1,t) = RP_3(1,t) = RP_3(1,t+1) \\ &\geq \min[RP_3(1,t+1), RP_4(1,t+1)]. \end{aligned}$$

Case 2: $i \neq 1$ and link ℓ is scheduled at time $t+1$. Since $RP_2(i,t+1) = RP_2(i,t)$, packet i is not scheduled by link ℓ. Therefore, packet $i-1$ is ahead of packet i, and $P_2(i-1,t) =$

$P_2(i,t)$. We then have from Lemma 22 (i) that $R_2(i-1,t) = R_2(i,t)$. Thus,

$$RP_2(i,t) = RP_2(i-1,t). \tag{59}$$

From the induction hypothesis, $RP_2(i-1,t) \geq \min[RP_3(i-1,t), RP_4(i-1,t)]$. Without loss of generality, assume that $\min[RP_3(i-1,t), RP_4(i-1,t)] = RP_3(i-1,t)$. Therefore,

$$RP_2(i-1,t) \geq RP_3(i-1,t). \tag{60}$$

From (59), (60), Lemma 24, and Lemma 27, we have that

$$\begin{array}{rcl} RP_2(i,t) & = & RP_2(i-1,t) \geq RP_3(i-1,t) \\ & \geq & RP_3(i,t) \geq RP_2(i,t). \end{array} \tag{61}$$

Hence, $RP_3(i-1,t) = RP_3(i,t)$. From Lemma 26 (ii), we know that $R_3(i-1,t) = R_3(i,t)$ and $P_3(i-1,t) = P_3(i,t)$. It then follows from Lemma 22 (i) that, for system Y_3, packet $i-1$ is ahead of packet 1. Since $P_3(i-1,t) = P_3(i,t)$, packet i can not be served at time $t+1$. Therefore,

$$RP_3(i,t+1) = RP_3(i,t). \tag{62}$$

From (54), (61), and (62), we conclude that

$$\begin{array}{rcl} RP_2(i,t+1) & = & RP_2(i,t) = RP_3(i,t) = RP_3(i,t+1) \\ & \geq & \min[RP_3(i,t+1), RP_4(i,t+1)]. \end{array}$$

Case 3: link ℓ is not scheduled at time $t+1$. From the induction hypothesis, we know that $RP_2(i,t) \geq \min[RP_3(i,t), RP_4(i,t)]$. Without loss of generality, assume that $\min[RP_3(i,t), RP_4(i,t)] = RP_3(i,t)$. Therefore, $RP_2(i,t) \geq RP_3(i,t)$. From Lemma 27, we have that $RP_3(i,t) \geq RP_2(i,t)$. Thus,

$$RP_3(i,t) = RP_2(i,t). \tag{63}$$

We then have from Lemma 26 (ii) that

$$R_3(i,t) = R_2(i,t) \text{ and } P_3(i,t) = P_2(i,t). \tag{64}$$

Since link ℓ is not scheduled, and system Y_2 and Y_3 have the same schedule after time 1, we have from (64) that, for system Y_3, the corresponding queue of packet i will also not be scheduled at time $t+1$. Hence, for systems Y_2 and Y_3, the relative positions of packet i will not increase at time $t+1$, i.e.,

$$RP_2(i,t+1) = RP_2(i,t), \text{ and } RP_3(i,t) = RP_3(i,t+1) \tag{65}$$

From (65) and (63), we conclude that

$$\begin{array}{rcl} RP_2(i,t+1) & = & RP_2(i,t) = RP_3(i,t) = RP_3(i,t+1) \\ & \geq & \min[RP_3(i,t+1), RP_4(i,t+1)]. \end{array}$$

■

Lemma 32: For any time t, $2T_2(t)+1 \geq T_3(t)+T_4(t)$.

Proof: At the beginning of time slot $t+1$, suppose that packet i is the next outgoing packets at the last queue for system Y_2, which means that it is ahead of other packets in the system Y_2. Therefore,

$$T_2(t) = R_2(i,t)W + i - 1. \tag{66}$$

Without loss of generality, assume that $\min[RP_3(i,t), RP_4(i,t)] = RP_3(i,t)$. From Lemma 31, we know that $RP_2(i,t) \geq RP_3(i,t)$. We then have from Lemma 26 (i) that

$$R_2(i,t) \geq R_3(i,t). \tag{67}$$

From (66), (67), and (25), we have that

$$T_2(t) = R_2(i,t)W + i - 1 \geq R_3(i,t)W + i - 1 \geq T_3(t).$$

Furthermore, we have from Lemma 30 that $T_2(t)+1 \geq T_4(t)$. Thus, we conclude that

$$2T_2(t) + 1 \geq T_3(t) + T_4(t).$$

■

Proof of Lemma 10: We prove this lemma by contradiction. Suppose that at time t,

$$T_1(t) + T_2(t) < T_3(t) + T_4(t). \tag{68}$$

From Lemma 32, we know that

$$T_3(t) + T_4(t) \leq 2T_2(t) + 1.$$

Thus,

$$T_1(t) < T_2(t) + 1.$$

From Lemma 28, we have that $T_1(t) \geq T_2(t)$, and it implies that $T_1(t) = T_2(t)$. From Lemma 28, we also know that $T_1(t) \geq T_3(t) \geq T_2(t)$. Hence, $T_3(t) = T_2(t)$. Similarly, we can show that $T_4(t) = T_2(t)$. We then conclude that

$$2T_2(t) = T_1(t) + T_2(t) = T_3(t) + T_4(t).$$

This contradicts with (68). ■

APPENDIX B
PROOF OF THEOREM 13

The proof of this theorem is very similar to the proof of Theorem 5 with only a small difference. Recall the probability distribution of $\vec{S}^{(3)}(\cdot)$ and $\vec{S}^{(4)}(t)$ are

$$P\{\vec{S}^{(3)}(t) = \vec{e}_\ell\} = \mu'_\ell, P\{\vec{S}^{(3)}(t) = \vec{x}_0\} = 1 - \sum_\ell \mu'_\ell,$$

and

$$P\{\vec{S}^{(4)}(t) = \vec{e}_\ell\} = \mu', P\{\vec{S}^{(4)}(t) = \vec{x}_0\} = 1 - \sum_\ell \mu'.$$

Notice that $\mu'_\ell \geq \mu'$. To prove this theorem, we also prove the counter part of Theorem 11 first.

Theorem 33:

$$\begin{array}{rcl} & & E[f(\vec{S}^{(4)}(1), \cdots, \vec{S}^{(4)}(k-1), \vec{S}^{(3)}(k), \cdots, \vec{S}^{(3)}(t))] \\ & \geq & E[f(\vec{S}^{(4)}(1), \cdots, \vec{S}^{(4)}(k), \vec{S}^{(3)}(k+1), \cdots, \vec{S}^{(3)}(t))] \end{array}$$

, for any $k = 1, 2, \ldots, t$.

Using similar technique of Theorem 12 and Theorem 5, we can then show that

$$E[f(\vec{S}^{(3)}(1), \cdots, \vec{S}^{(3)}(t))] \geq E[f(\vec{S}^{(4)}(1), \cdots, \vec{S}^{(4)}(t))],$$

and $T(\vec{S}^{(3)}(\cdot)) \geq T(\vec{S}^{(4)}(\cdot))$.

Proof of Theorem 33: To ease the presentation of the proof, define $\vec{S}^{(i)}(t_1,t_2)$ be the sequence $\vec{S}^{(i)}(t_1),\cdots,\vec{S}^{(i)}(t_2)$ for system $\vec{S}^{(i)}(\cdot)$. We first show the following inequality.

$$E[f(\vec{S}^{(4)}(1,k-1),\vec{S}^{(3)}(k,t))|\vec{S}^{(4)}(1,k-1),\vec{S}^{(3)}(k+1,t)] \geq E[f(\vec{S}^{(4)}(1,k),\vec{S}^{(3)}(k+1,t))|\vec{S}^{(4)}(1,k-1),\vec{S}^{(3)}(k+1,t)] \quad (69)$$

If (69) holds, then Theorem 33 can be proved by conditional expectation. To show (69), assume that $\vec{S}^{(4)}(1,k-1)$ equal to a sequence of fixed schedules $\vec{y}(1,k-1)$, and $\vec{S}^{(3)}(k+1,t)$ equal to a sequence of fixed schedules $\vec{y}(k+1,t)$. Denote $\phi(\vec{z})=f(\vec{y}(1,k-1),\vec{z},\vec{y}(k+1,t))$. We have that

$$\begin{aligned}
&E[f(\vec{S}^{(4)}(1,k-1),\vec{S}^{(3)}(k,t))|\vec{S}^{(4)}(1,k-1),\vec{S}^{(3)}(k+1,t)] \\
&- E[f(\vec{S}^{(4)}(1,k),\vec{S}^{(3)}(k+1,t))|\vec{S}^{(4)}(1,k-1),\vec{S}^{(3)}(k+1,t)] \\
&= \sum_{\ell}\phi(\vec{e}_\ell)\mu'_\ell + \phi(\vec{x}_0)\left(1-\sum_{\ell}\mu'_\ell\right) \\
&\quad - \sum_{\ell}\phi(\vec{e}_\ell)\mu' - \phi(\vec{x}_0)\left(1-\sum_{\ell}\mu\right) \\
&= \sum_{\ell}(\mu'_\ell-\mu')(\phi(\vec{e}_\ell)-\phi(\vec{x}_0)) \geq 0.
\end{aligned}$$

The last inequality is a application of Lemma 28. Notice that $\vec{e}_\ell \succcurlyeq \vec{x}_0$ and $\vec{y}(1,k-1)$ will generate an arbitrary initial packet placement. Let $\phi(\vec{e}_\ell)$ and $\phi(\vec{x}_0)$ be the throughput of system Y_1 and Y_2. Then we can apply Lemma 28 to have the last inequality. ∎

APPENDIX C
COUNTER EXAMPLE

In this appendix, We will give a example to show that f is not a supermodular function with respect to the whole sequence of schedules. Consider a 3-hop tandem network with 2 packets. Let $T_i(t)$, $i=1,\cdots,4$, be the total number of departing packets from queue H for system Y_i at the end of time t. Further, let $\vec{x}_i(t)$, $i=1,\cdots,4$, be the schedule for system Y_i at time t. The four deterministic systems are given as follows.

System Y_1:

time	1	2	3	4	5
state	$(2,0,0)$	$(1,1,0)$	$(0,1,1)$	$(1,0,1)$	$(1,1,0)$
$\vec{x}_1(t)$	$(1,1,1)$	$(1,1,1)$	$(1,1,1)$	$(1,1,1)$	$(1,1,1)$
$T_1(t)$	0	0	1	2	2

System Y_3:

time	1	2	3	4	5
state	$(2,0,0)$	$(1,1,0)$	$(1,0,1)$	$(1,1,0)$	$(1,0,1)$
$\vec{x}_3(t)$	$(1,0,1)$	$(0,1,0)$	$(1,0,1)$	$(0,1,0)$	$(1,0,1)$
$T_3(t)$	0	0	1	1	2

System Y_4:

time	1	2	3	4	5
state	$(2,0,0)$	$(2,0,0)$	$(1,1,0)$	$(1,0,1)$	$(1,1,0)$
$\vec{x}_4(t)$	$(0,1,0)$	$(1,0,1)$	$(0,1,0)$	$(1,0,1)$	$(0,1,0)$
$T_4(t)$	0	0	0	1	1

System Y_2:

time	1	2	3	4	5
state	$(2,0,0)$	$(2,0,0)$	$(2,0,0)$	$(2,0,0)$	$(2,0,0)$
$\vec{x}_2(t)$	$(0,0,0)$	$(0,0,0)$	$(0,0,0)$	$(0,0,0)$	$(0,0,0)$
$T_2(t)$	0	0	0	0	0

From the above example, we know that $T_1(5)=2$, $T_2(5)=0$, $T_3(5)=2$, and $T_4(5)=1$. Hence, $T_1(5)+T_2(5)<T_3(5)+T_4(5)$. Since at any time t $\vec{x}_1(t)=\vec{x}_3(t)\vee\vec{x}_4(t)$, and $\vec{x}_2(t)=\vec{x}_3(t)\wedge\vec{x}_4(t)$, we conclude that f is not a supermodular function with respect to the whole sequece of schedule vectors.

REFERENCES

[1] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource Allocation and Cross-Layer Control in Wireless Networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[2] X. Lin, N. B. Shroff, and R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, Aug. 2006.

[3] E. Modiano, D. Shah, and G. Zussman, "Maximizing Throughput in Wireless Networks via Gossiping," in *ACM SIGMETRICS*, 2006.

[4] S. Sanghavi, L. Bui, and R. Srikant, "Distributed Link Scheduling with Constant Overhead," in *ACM SIGMETRICS*, 2007.

[5] X. Lin and N. B. Shroff, "The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks," *IEEE/ACM Trans. on Networking*, vol. 14, no. 2, pp. 302–315, 2006.

[6] X. Lin and S. Rasool, "Constant-Time Distributed Scheduling Policies for Ad Hoc Wireless Networks," *IEEE Trans. on Automatic Control*, vol. 54, no. 2, pp. 231–242, February 2009.

[7] C. Joo and N. B. Shroff, "Performance of Random Access Scheduling Schemes in Multi-hop Wireless Networks," in *IEEE INFOCOM*, 2007.

[8] P. Charporkar, K. Kar, and S. Sarkar, "Throughput Guarantees Through Maximal Scheduling in Wireless Networks," *IEEE Trans. on Inf. Theory*, vol. 54, no. 2, pp. 572–594, Feb. 2008.

[9] C. Joo, X. Lin, and N. B. Shroff, "Understanding the Capacity Region of the Greedy Maximal Scheduling Algorithm in Multi-hop Wireless Networks," in *IEEE INFOCOM*, 2008.

[10] A. Gupta, X. Lin, and R. Srikant, "Low-Complexity Distributed Scheduling Algorithms for Wireless Networks," *IEEE/ACM Trans. on Networking*, December 2009.

[11] L. Bui, R. Srikant, and A. L. Stolyar, "Novel Architectures and Algorithms for Delay Reduction in Back-pressure Scheduling and Routing," in *IEEE INFOCOM Mini-Conference*, 2009.

[12] P.-K. Huang and X. Lin, " The End-to-End Delay Performance of A Class of Wireless Scheduling Algorithms," in *Allerton Conference on Communication, Control, and Computing, invited talk*, 2010.

[13] L. Huang and M. J. Neely, "Delay Reduction via Lagrange Multipliers in Stochastic Network Optimization," in *WiOpt*, 2009.

[14] M. J. Neely, "Delay-Based Network Utility Maximization," in *IEEE INFOCOM*, 2010.

[15] ——, "Delay Analysis for Maximal Scheduling in Wireless Networks with Bursty Traffic," in *IEEE INFOCOM*, 2008.

[16] L. B. Le, K. Jagannathan, and E. Modiano, "Delay Analysis of Maximum Weight Scheduling in Wireless Ad Hoc Networks," in *IEEE CISS*, Baltimore, MD, March 2009.

[17] G. R. Gupta and N. B. Shroff, "Delay Analysis for Multi-hop Wireless Networks," in *IEEE INFOCOM*, 2009.

[18] L. Huang and M. J. Neely, "Delay Efficient Scheduling Via Redundant Constraints in Multihop Networks," in *WiOpt*, 2010.

[19] L. B. Le, E. Modiano, and N. B. Shroff, "Optimal Control of Wireless Networks with Finite Buffers," in *IEEE INFOCOM*, 2010.

[20] P. Jayachandran and M. Andrews, "Minimizing End-to-End Delay in Wireless Networks Using a Coordinated EDF Schedule," in *IEEE INFOCOM*, 2010.

[21] M. Xie and M. Haenggi, "Towards an End-to-End Delay Analysis of Wireless Multihop Networks," *Elsevier Ad Hoc Networks*, vol. 7, pp. 849–861, July 2009.

[22] S. Jagabathula and D. Shah, "Optimal Delay Scheduling in Networks with Arbitrary Constraints," in *ACM SIGMETRICS*, June 2008.

[23] B. Hajek and G. Sasaki, "Link Scheduling in Polynomial Time," *IEEE Trans. on Inf. Theory*, vol. 34, no. 5, pp. 910–917, September 1988.
[24] S. Sarkar and L. Tassiulas, "End-to-end Bandwidth Guarantees Through Fair Local Spectrum Share in Wireless Ad-hoc Networks," in *Proc. of the IEEE CDC*, Maui, Hawaii, December 2003.
[25] F. P. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.
[26] A. Stolyar, "Large number of queues in tandem: Scaling properties under back-pressure algorithm," *Bell Labs Technical Memo*, October 2009.
[27] S. H. Low and D. E. Lapsley, "Optimization Flow Control–I: Basic Algorithm and Convergence," *IEEE/ACM Trans. on Networking*, vol. 7, no. 6, pp. 861–874, December 1999.
[28] S. Shakkottai and R. Srikant, *Network Optimization and Control*. Foundations and Trends in Networking, 2007.
[29] P. Bremaud, *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues*. New York: Springer-Verlag, 1999.
[30] S. M. Ross, *Stochastic Processes*, 2nd ed. New York: John Wiley & Son, 1996.
[31] M. Shaked and J. G. Shanthikumar, *Stochastic Orders and Their Applications*. Academic Press, 1994.
[32] A. Muller and D. Stoyan, *Comparison Methods for Stochastic Models and Risks*. John Wiley & Son, 2002.
[33] P. Billingsley, *Probability and Measure*, 3rd ed. New York: John Wiley & Son, 1995.
[34] M. Schwartz, *Telecommunication Networks: Protocols, Modeling, and Analysis*. Addison Wesley, 1987.
[35] S. Athuraliya and S. H. Low, "Optimization Flow Control–II: Implementation," *Technical report, University of Melbourne, http://netlab.caltech.edu/publications.php*, 2000.